BGP-aware IGP Link Weight Optimization in Presence of Route Reflectors

Simon Balon and Guy Leduc
Research Unit in Networking (RUN) - Université de Liège (ULg)
{Simon.Balon,Guy.Leduc}@ulg.ac.be

***Abstract*—The first generation of IGP Link Weight Optimizers (LWOs) was based on presumably invariant intra-domain traffic matrices only, ignoring the fact that updating link weights had a side effect on these traffic matrices due to hot-potato routing, thus resulting in suboptimal link weight settings, and sometimes to very bad performance.**

The second generation of IGP LWOs, referred to as BGP-aware LWOs, has been able to optimize link weights while taking hot-potato effects into account. However, these tools relied on the complete visibility assumption fulfilled by e.g. a full-mesh iBGP configuration.

This paper proposes a third generation LWO, still BGP-aware, but also able to work with iBGP configurations based on route reflectors, which usually hide some reachability information from routers. This partial visibility may cause various problems, including path deflections (i.e., the actual egress router is not the expected one), which may in turn create forwarding loops.

Our LWO embeds a BGP routing solver which can always predict the actual egress router, even when route reflectors are used. It can also forbid solutions leading to path deflection. Its efficiency is evaluated on a real dataset, and compared to other LWOs.

***Index Terms*—Traffic Engineering, Hot-potato Routing, BGP, IGP, OSPF, Path Deflection**

I. Introduction & Motivation

Several methods have been proposed to engineer the traffic inside a network running a link state Interior Gateway Protocol (IGP) like OSPF or ISIS. These algorithms try to find the best possible set of link weights so that shortest paths based on these link weights lead to a good load balance inside the network. [6] is one such algorithm. Typically these optimizers consider that the intradomain traffic matrix is invariant with respect to the link weights. But this assumption is not fully respected in current networks. Indeed for one part of the traffic (which may be significant), the egress point inside the autonomous system (AS) is chosen according to the BGP (Border Gateway Protocol) hot-potato rule, which takes into account the IGP cost to the possible egresses, which is actually based on the link weights.

S. Balon is a Research Fellow of the Belgian National Fund for the Scientific Research (FNRS) and also partially funded by the EU under the ANA FET project (FP6-IST-27489).

Some methods have been proposed to integrate this BGP hot-potato rule in the optimizer ([2], [3], [10]). But we will see that these methods implicitly suppose that every router in the AS is aware of every route announce toward every destination. This is always correct when the iBGP configuration is a full-mesh, but not when route reflectors are used.

Indeed in iBGP full-mesh configurations, each route announcement received by any router on an eBGP session is retransmitted on iBGP sessions to every other BGP router in the AS. So every router is aware of all the available routes for every destination and it can choose its *global* best route in the whole set of available routes.

The problem of the iBGP full-mesh is that it requires $\frac{n.(n-1)}{2}$ iBGP sessions in an AS composed of n BGP routers. This may be prohibitive in large ASes. To solve this scalability problem, network operators usually install route reflectors, which reduces the number of iBGP sessions ([4]). But it is known that route reflectors can introduce anomalies that are due to partial route visibility. This happens because clients receive only the *best* route from their route reflector. So route reflector clients do not have access to the whole set of available routes. Moreover the route reflector's best route may differ from the client *global* best route (i.e. the route that it would have chosen, had it seen every available route). This can lead to non-optimal hot-potato egress choice. Another anomaly that can be created by route reflectors is the forwarding deflection. This happens when a router selects its best route and on the forwarding path to the egress point corresponding to this best route, there is a router that selects another best route and thus another egress point. In this case we say that the traffic is deflected. We will see an example of network configuration inducing a path deflection in section III.

The biggest problem with deflections is that these can introduce forwarding loops ([7]):

- Intra-AS loops due to the combination of multiple deflections in a particular way.
- Inter-AS loops which can appear when incorrect ASPATH information is transmitted by routers for which the traffic has been deflected. As the ASPATH information is not correct, the BGP loop detection mechanism may not work properly.

In this paper we first want to study the impact of

partial visibility on LWOs that make a wrong assumption of complete visibility. This impact can range from a non-optimal traffic engineering solution found by the optimizer to the more dangerous introduction of path deflections in the AS or even forwarding loops. This motivated us to develop a BGP-aware LWO able to take account of partial visibility in presence of Route Reflectors, which allows to detect and forbid path deflections possibly leading to forwarding loops.

The paper is structured as follows. Section II presents related works. Section III analyses in detail the different iBGP configurations that are conflicting with traditional LWOs. Section IV uses simulations to evaluate the impact of partial visibility on LWOs. To this end we run a LWO that wrongly considers complete visibility on a network whose iBGP configuration contains a route reflector. Then section V presents a new LWO embedding a BGP simulator to predict the actual egress points and detect path deflections. We evaluate the performance of this new LWO in section VI with simulations based on a real dataset. Finally section VII concludes this paper.

II. Related Work

Section II-A presents BGP Route Reflector related works while section II-B presents LWOs related works.

A. Route Reflection

BGP suffers from several problems ([5]). In particular, route reflectors can introduce anomalies that include path deflection or forwarding loops. These problems have been quite extensively studied (in [7] for example). In this paper we analyze the impact of these route reflector problems on Link Weight Optimizers. We will see that the partial visibility introduced by route reflectors can have a big impact on the performance of LWOs. But a bigger issue is that LWOs can also introduce path deflections and forwarding loops in a network containing route reflectors. One first simple way to solve these problems would be to ensure complete visibility, but this is really not a trivial task. In [7] they show that determination of iBGP configuration correctness is NP-hard[1]. However they provide sufficient conditions on network configurations that guarantee correctness. These sufficient conditions can help but do not solve all problems. First the sufficient conditions only guarantee correctness for one set of link weights. So if an LWO is run on an AS for which the sufficient conditions hold, it may not be the case anymore after the optimization. Furthermore, in [5] we can read that *"First, these sufficient conditions are very strong: they imply that every edge that is on a shortest path to an exit point must have a corresponding iBGP session. Second, the conditions require that redundant route reflectors must be located close to the primary to have a similar view of the best routes, introducing undesirable fate sharing. Finally, we have recently discovered IGP topologies for which this constraint is not satisfiable"*.

In [12] they propose a new method to build an iBGP topology made of route reflectors which can guarantee the *complete visibility* property[2]. The *complete visibility* property is defined in [12] as: *The dissemination of information amongst the routers should be "complete" in the sense that, for every external destination, each router picks the same route that it would have picked had it seen the best route from every other BGP router in the AS*. This technique is a great work and good step toward good route reflector configuration in networks. If that technique is used to design the hierarchy of route reflectors, there is no problem with previously proposed LWOs, as they guarantee that the egress point used by any router is the egress that would be chosen in the iBGP full-mesh case, and this remains true for any set of link weights. But it is not clear whether the technique of [12] can be used in practice to design all the route reflector configurations. Indeed the algorithm is really not flexible: it proposes one and only one solution which is correct but absolutely not tunable. What happens if the proposed solution still contains too many iBGP sessions? The number of iBGP sessions is reduced by a factor between 2.5 and 5 depending on the network compared to an iBGP full-mesh. But it is not clear that it is sufficient for every network. Another potential problem is the number of route reflectors and the number of levels in the route reflector hierarchy which seems quite high in the generated configurations. This may be considered too complex by some network operators. Moreover this solution does not provide redundancy as each router may receive only one route and the iBGP organization must potentially be changed after each internal link failure, which makes this solution unusable in practice.

B. LWOs

Some methods have been proposed to integrate the BGP hot-potato rule in the optimizer. For example in [2], they recompute the intradomain traffic matrix from the interdomain traffic matrix[3] at each step of the optimizer, choosing for each ingress node the nearest next-hop for each destination prefix. This solution is not correct when the complete visibility property is not respected, as in this case it is not always the nearest next-hop which is used, but the nearest *available* next-hop, which may be very different.

In [3] and [10] the proposed method consists in adding some virtual nodes to the intradomain topology to model the reachability of prefixes via multiple egress points. These solutions are based on the fact that the traffic

[1]In that paper they define a correct iBGP configuration to be one that is anomaly-free for every possible set of routes sent by neighboring ASes. They focus on anomalies that can cause the protocol to diverge, and those that can cause path deflections.

[2]The proposed method also guarantees *loop-free forwarding* and *robustness to IGP failures*.

[3]We consider that an inter-domain traffic matrix is a matrix representing traffic from ingress routers to IP prefixes.

will follow the shortest path based on the link weights from an ingress node to the virtual node modeling the destination prefix, which respect both the IGP based on shortest paths and BGP's hot-potato rule. In this case it is possible to reuse existing Link Weight Optimizers with efficient heuristics on the extended topology. But again this method requires the complete visibility property. If that property is not respected, the chosen next-hop may not be the one which is on the shortest path between the ingress node and the virtual node modeling the destination prefix, as this next-hop may not be available at this ingress node. This means that from an algorithmic point of view it is not sufficient to compute a shortest path to infer the egress node chosen by BGP. A more complex model of the BGP decision process is required.

III. The problem of link weights optimizers and route-reflectors

We will illustrate the problems of link weights optimizers and route-reflectors on the topology of figure 1. Suppose that R_1, R_2, R_3 and R_4 are part of the AS we want to engineer, N_1 and N_2 are part of a neighboring AS and can both reach IP prefix P_1, N_1 has an eBGP session with R_3 and N_2 with R_4. Suppose also that a BGP route advertisement message concerning prefix P_1 is sent on both eBGP sessions ($N_1 \rightarrow R_3$ and $N_2 \rightarrow R_4$) with the same BGP attributes. We will mainly consider the following two configurations of iBGP session:

- There is a full-mesh of iBGP sessions; *or*
- R_2 is the route reflector for R_1, R_3 and R_4.

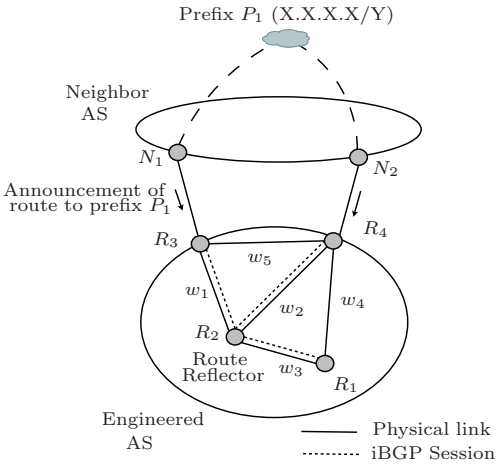

Fig. 1. Toy Example

First, consider the full-mesh iBGP configuration with the link weight setting of figure 2a. R_1, R_2, R_3 and R_4 receive the two available routes on their {e,i}BGP sessions. R_3 chooses the route with N_1 as next-hop while R_4 chooses the route with N_2 as next-hop, both respecting the rule that enforces routers to prefer eBGP-learned routes to iBGP-learned ones. R_1 and R_2 have received these two routes on iBGP sessions, so these routers use the hot-potato rule to choose their best route. R_2 chooses the route with N_1 or N_2 as next-hop depending on the IGP distance between R_2 and respectively R_3 and R_4. For the same reason R_1 chooses between the two available routes depending on the IGP distance from R_1 to R_3 and R_4. For the particular link weights setting of figure 2a, R_2 chooses the route with N_1 as next-hop while R_1 chooses the route with N_2 as next-hop. We clearly see that adding a virtual node corresponding to prefix P_1 and two virtual links from N_1 to P_1 and from N_2 to P_1 allows an algorithm to consider that the next-hop chosen by R_2 (resp. R_1) is on the shortest path from R_2 (resp. R_1) to P_1, provided that these two virtual links and the two interdomain links have a weight of 0. This is the idea elaborated in [3].

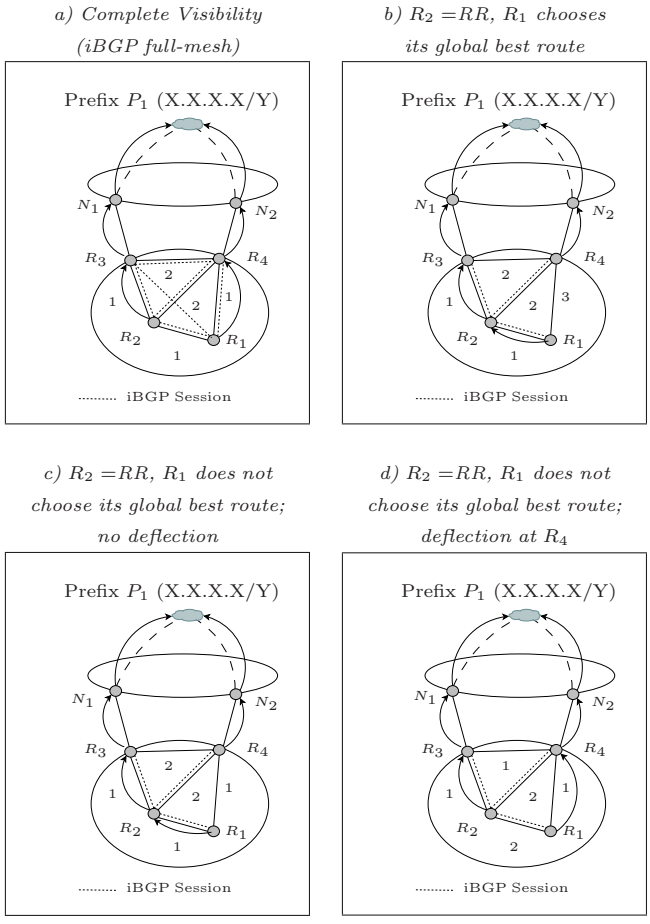

Fig. 2. Different iBGP configurations and link weights setting

Now we will analyze the same network where R_2 is the route reflector for R_1, R_3 and R_4. Figures 2b, 2c and 2d do all use these iBGP sessions. The only differences are on the link weights setting. According to the BGP protocol the best routes will be determined as follows. R_3 and R_4 will send their routes to the route reflector R_2 which will run its BGP decision process and thus select one of these depending on the IGP distance from R_2 to R_3 and R_4

respecting the hot-potato rule. Now R_2 will forward this (and only this) best route to R_1 which will choose the same route as only this one is available at this ingress router. We clearly see that R_1 will use the same egress point as R_2 even though it is not necessarily the nearest egress point for R_1. In that case we say that R_1 has a partial visibility if its *globally* best egress is not visible to it.

Consider the link weights setting of figure 2b. In that case R_2 chooses R_3 as egress node, and so does R_1 as only this route is available. In this case, R_1 chooses its *globally* best route as the distance from R_1 to R_3 is 2 while the distance from R_1 to R_4 is 3. This means that with these link weights we have complete visibility as defined in section II (i.e. each router selects the route which is its *globally* best route).

Now consider the link weights setting of figure 2c. In that case R_2 still chooses R_3 as egress node, and so does R_1 as only this route is available, although its *globally* best route is via R_4 as the distance from R_1 to R_3 is 2 while the distance from R_1 to R_4 is 1. Anyway in this case there is no path deflection as the shortest path from R_1 to R_3 is $R_1 \rightarrow R_2 \rightarrow R_3$, which does not cross R_4.

Finally consider the link weights setting of figure 2d. In that case R_2 still chooses R_3 as egress node, and so does R_1 as only this route is available. It is still not its *globally* best route as the distance from R_1 to R_3 is 2 while the distance from R_1 to R_4 is 1. But now there is a path deflection. Indeed the shortest path from R_1 to R_3 is $R_1 \rightarrow R_4 \rightarrow R_3$. So R_1 forwards traffic to R_4 thinking that R_4 will forward it to R_3 while R_4 will actually send it directly to N_2. In that case we say that the traffic is deflected by R_4. So the real path of the traffic will be $R_1 \rightarrow R_4 \rightarrow N_2$ and not $R_1 \rightarrow R_4 \rightarrow R_3 \rightarrow N_1$.

Note that the combination of multiple path deflections can lead to forwarding loops inside the AS. We refer to [7] for an example of such situation.

The consequences of the presence of route reflectors on the performance of LWOs are the following:

- LWOs that consider complete visibility may lead to non-optimal traffic engineering solutions, as the link utilizations that are predicted by the optimizers do not reflect the ones that will be observed in the network, due to errors in the egress prediction. For example, in case of figure 2c, the optimizer will consider the routes of figure 2a;
- LWOs that consider complete visibility may introduce path deflections. Indeed one *deflection-free* configuration (e.g. figure 2b) can be transformed into a BGP configuration *containing deflection* (e.g. figure 2d) simply by changing the set of link weights.

To summarize, we have to consider different cases of route reflector configurations, from the safer to the most dangerous:

1 The configurations with complete visibility for every possible link weights setting. For example configurations generated using the technique described in [12] or simple iBGP full-mesh configurations.
2 The configurations with complete visibility for the present link weights setting, without guarantee that all possible link weights settings will preserve the complete visibility property (e.g. figure 2b).
3 The configurations without complete visibility for the present link weights setting, but for which no deflection occurs (e.g. figure 2c).
4 The configurations without complete visibility for the present link weights setting, for which simple deflections occur between egresses toward the **same** neighbor AS[4] (e.g. figure 2d).
5 The configurations without complete visibility for the present link weights setting, for which simple deflections occur between egresses toward **different** neighbor ASes (e.g. figure 2d, but with N_1 and N_2 in two different ASes).
6 The configurations without complete visibility for the present link weights setting, for which multiple deflections occur and form a forwarding loop inside the AS (see [7] for such an example).

It can be noted that we make a difference between cases 4 and 5 because simple deflections between two egress routers that connect to the same neighbor AS (and which have the same corresponding ASPATHs) are not potentially the cause of inter-AS loops. Indeed in that case there is no erroneous ASPATH information transmitted to neighboring ASes. We think that this kind of deflection could be allowed by a network operator, as it is not dangerous. Moreover we will see in section VI that allowing this kind of deflection may let the optimizer find a better TE solution in the extended search space.

Case 1 happens when the algorithm of [12] is used. If run on the topology of figure 1, it would probably produce the following iBGP configuration: R_2 and R_4 are both route reflectors for R_1 and R_3. It is not possible to find a topology with only one route reflector that would produce the complete visibility property for every possible link weights setting.

In case 1 we are sure to avoid the problems due to partial visibility and so BGP-aware LWOs (like [2], [3], [10]) can be used. In all the other cases, these optimizers can fail because of potential partial visibility and path deflections. This can be simply explained by the fact that changing the link weights setting can drive from any configuration in the set {2,3,4,5,6} to any other configuration in the set {2,3,4,5,6}. So even if we start a link weight optimization on a configuration of type 2, the resulting link weights can lead to a configuration from type 2 to 6 in the worst case.

The important point is that if the iBGP configuration of a network is not of type 1, it is safer to run a LWO

[4]In fact we should say for which the corresponding ASPATH is the same in both routes, which is generally the case when both routes are received from the same neighbor AS.

that can deal with partial visibility and avoid (or at least minimize the number of) deflections in the network. The iBGP configuration with optimized link weights setting should ideally be of type {1,2,3,4}, while type {5,6} should be avoided. Indeed type {1,2,3,4} guarantee that *no forwarding loops* will be created.

IV. Evaluating the impact of partial visibility on Link Weights Optimizers

In this section we evaluate the impact of partial visibility on the traffic engineering quality of solutions found by LWOs that make a wrong assumption of complete visibility.

To this end we will test a LWO that considers complete visibility on a topology that contains a route reflector giving partial visibility only. We use the LWO presented in [3] which is available in TOTEM ([1]). We will refer to this LWO as *BGP-CV-LWO* as it correctly takes the BGP hot-potato rule into account when the complete visibility (CV) property is respected. The data used in the simulations of this section are real data of a multi-gigabit operational network that spreads over the European continent and is composed of about 25 nodes and 40 bidirectional intradomain links. Link capacities range from 155Mbps to 10Gbps. It is a transit network that has two providers connected with about 10 interdomain links, has other peer ASes connected with about 15 shared-cost links, and has more than 25 customer ASes, which are mainly single-homed. The total traffic exchanged is about 10 Gbps on average.

We have used the technique exposed in [3] to build our model from BGP dumps and netflow data. We had access to about one month of traces (one month of year 2005), one BGP dump per day and one sampled netflow file for each ingress router. With these data we have generated 2,512 aggregated interdomain traffic matrices (each matrix is an average over 15 minutes). Some of these induce a low load on the network while some induce a high load. This whole set of traffic matrices is representative of the traffic on the studied network. The average number of prefixes is 160,973 of which 97.2% (156,407) are hot-potato (i.e. multi-egress) prefixes. If we now take traffic into account, we have measured that these 97.2% amount to 35.6% of the traffic on average.

A. Simulation description

The actual iBGP configuration of the network we have studied is an iBGP full-mesh. We have designed an hypothetical simple BGP route reflector hierarchy to simulate what happens with partial visibility. The route reflector hierarchy we consider is the following: (this hierarchy is inspired from [11]): one router which has a *central* position in the topology is chosen as the route reflector and all the other routers of the network are clients of this route reflector[5].

For each traffic matrix, we have run *BGP-CV-LWO* which considers hot-potato traffic, but with complete visibility. Values labelled *"predicted"* denote the link loads predicted by this algorithm which assumes optimal hot-potato egress selection. Values labelled *"resulting"* represent the actual link loads resulting from the BGP behavior with possible non-optimal egress selection due to partial visibility causing non best route choice for some routers. We have used the C-BGP simulator ([9]) to compute the actual outcome of the BGP decision process in this situation.

B. Simulation results

Figure 3 presents the CDFs (cumulative distribution functions) of the maximal link utilization for all the traffic matrices (TMs). We can see that the optimizer predicted that about 75% of the TMs will lead to a maximal utilization below 40% while it is in practice only true for about 45% of the TMs. Note that a maximal link utilization over 100% means that one link is overloaded in the network. The mean error of the optimizer concerning the maximal link utilization is about 7.8%. In the worst case, the maximal utilization is greater than 140% while the optimizer predicted less than 65%. The *BGP-CV-LWO* that was very efficient and precise with complete visibility provides very poor results in this route reflector configuration. If we compare these results with the results of [3] we can say that with this route reflector configuration the *BGP-CV-LWO* behaves as badly as a completely BGP-blind LWO. These results show that a more precise model of the BGP decision process is really needed in the link weights optimizer.

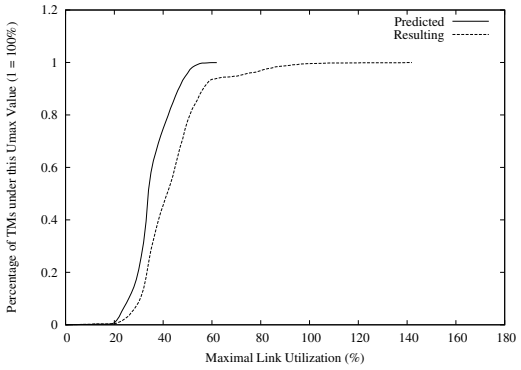

Fig. 3. CDFs of *"predicted"* versus *"resulting"* U_{max} over all TMs for *BGP-CV-LWO* on the topology which contains a route reflector

V. A generic BGP-aware Link Weights Optimizer

We now present *BGP-LWO* a new LWO which can consider any iBGP configuration. As a result it can correctly

[5]By choosing the router which has the "central" position we mean that we have chosen the router whose sum of its distance (in terms of number of hops) to all other routers is minimal.

predict the actual egress node for each prefix. To our knowledge, this is the first link weight computation algorithm that correctly models hot-potato reroutings in the presence of Route-Reflectors (leading to partial visibility) and that detects path deflections.

A. Description of the algorithm

The algorithm is based on the simulated annealing meta-heuristic[6]. We have developed this algorithm from scratch as it was too dangerous to modify an existing LWO (for example the LWO of the TOTEM toolbox [1]) to consider Route Reflectors, as it would have required a complete rewriting of the core of the software. We think that such a deep modification would be too error-prone to be considered. This is why we decided to write a new LWO based on a well-known heuristic. In section VI-A we validate the quality of our algorithm which seems to provide quite good solutions. We know that more sophisticated and optimized algorithms have been proposed for LWO, which could improve the computational efficiency of the resulting software. Anyway this is not the goal of this study. The goal of this study is to prove that it is possible to design an LWO which correctly predicts link utilizations in presence of route reflectors in the iBGP configuration, in addition to avoiding dangerous path deflections. Future works could improve the computational efficiency of this algorithm.

We use a modified version of the objective function of [6]. We just add a penalty for each deflection that happens in the network[7]. This penalty should drive the optimizer to solutions that minimize the number of path deflections. Both parts of the objective function should be minimized. The value of the penalty associated with each deflection can be tuned to find a trade-off between both objectives. The network operator should test different values for the penalty and choose one of the resulting solutions. If the network operator absolutely wants to avoid path deflections, the optimizer should return a deflection-free solution if the penalty is set high enough (if such a solution exists, of course). When multiple deflections are combined to form a forwarding loop, the value of the objective function is set to ∞. This will lead the optimizer to discard all these solutions.

B. Obtaining the required data

The optimizer requires some BGP and traffic data. We consider two disjoint categories of destination IP prefixes. The *single-egress prefixes* are those prefixes for which the BGP next-hop is chosen by one of the first 4 BGP criteria (local-pref, ASPATH length, origin number, MED). The *hot-potato prefixes* are all the other prefixes. For each of them there is at least one router in the domain that has used the hot-potato criterion to select the next-hop. Note that only *hot-potato prefixes* are potentially subject to partial visibility and path deflections.

An important point is the aggregation of *hot-potato prefixes*. Indeed the *hot-potato prefixes* can be grouped into clusters of prefixes which have the same set of possible egress nodes. Inside such a cluster all the prefixes will have the same egress point (be it deflected or not), and this is the only point which is important for us. On the studied dataset this has reduced the number of prefixes to give to the simulator from 156,407 *hot-potato prefixes* to 26 *aggregated hot-potato prefixes*.

C. Computing the egress node using C-BGP

Our algorithm uses the C-BGP simulator ([9]) to compute the egress node for each aggregate of *hot-potato* prefixes. Running a C-BGP simulation gives us the best route from every router inside the AS toward every cluster of hot-potato prefix. Thus we are able to check whether path deflections occur by combining the shortest path information with these best routes. We also check whether forwarding loops have been introduced due to multiple path deflections.

Let us note that the problem is computationally tractable because we do not run the simulator with all the prefixes, but only with *aggregated hot-potato prefixes*.

VI. Evaluation of the proposed optimizer

In this section we will run the new LWO presented in section V on the dataset presented in section IV[8]. We will refer to our new optimizer which embbeds the C-BGP simulator as *BGP-LWO* as it can correctly model the BGP behavior in every iBGP configuration, which is different from *BGP-CV-LWO*, which had been designed with complete visibility in mind.

We first want to validate the quality of the solutions found by the new algorithm *BGP-LWO*. To this end in section VI-A we compare it to a state of the art LWO. This comparison is performed in an iBGP full-mesh configuration which does not favor one LWO over the other. In section VI-B we compare both algorithms on a configuration with one route reflector. This will allow us to evaluate the TE performance gain of *BGP-LWO* over *BGP-CV-LWO* which wrongly assumes complete visibility. Note that for a fair comparison we have not included a path deflection penalty in *BGP-LWO* in the simulations of sections VI-A and VI-B as of course *BGP-CV-LWO* does not consider path deflections either. Then in section VI-C we include a penalty for path deflections in the objective function. These simulations test the ability of *BGP-LWO* to avoid path deflections. Finally section VI-D evaluates the TE performance gain that can be realized when the network operator allows *non-dangerous* path deflections.

[6]Simulated annealing is a well-known meta-heuristic for combinatorial problems. It has been first introduced by Kirkpatrick et al. in [8].

[7]We count one penalty for each deflection from one node to one aggregated prefix. We explain in section V-C how deflections are predicted.

[8]The values of the different parameters of the simulated annealing heuristic used in these simulations are the following: $T_0 = 100000$, $L = 50$, $\alpha = 0.9$, $\epsilon = 2$, $K = 3$.

A. Quality of the new optimizer

First we want to validate the quality of the new algorithm. This point is examined by running our new LWO (*BGP-LWO*) and the LWO of [3] (*BGP-CV-LWO*) on the whole dataset considering an iBGP full-mesh. *BGP-CV-LWO* is based on the heuristics of [6] which we consider as the state of the art. Both algorithms should provide good results on this dataset with this iBGP configuration. Figure 4 presents this comparison. Values are sorted according to *BGP-CV-LWO*. Note that values of maximal link utilization under 33.3% are not very important in this comparison as the objective function used is linear under this value. This means that there is no significant penalty associated with highly utilized links used in this utilization range ([0;33,3%]). So that the optimizer will not really try to reduce a maximal link utilization under 33.3%. We can see on the figure that both optimizers provide solutions of similar quality. This means that the solutions found by *BGP-LWO* are quite good, at least on that dataset.

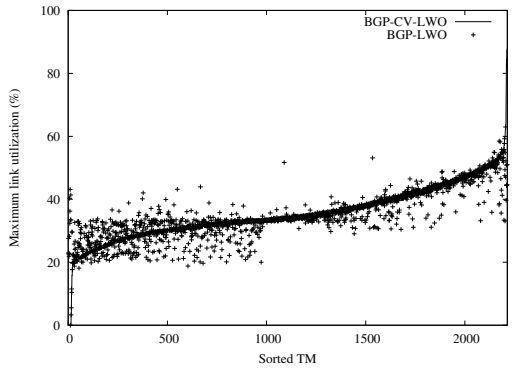

Fig. 4. Evaluation of the quality of the solutions found by *BGP-LWO* compared to *BGP-CV-LWO* in an iBGP full-mesh configuration

B. Actual prediction of the egress point

Now we evaluate the quality of our new LWO (*BGP-LWO*) for what it has been designed for, which is when the iBGP topology is not a full-mesh. In that situation it should perform better than other LWOs which have not been designed to work in these situations. We have run our *BGP-LWO* on the full dataset supposing that there is a route reflector in the AS (the iBGP configuration presented in section IV). We compare these results with the results of *BGP-CV-LWO* (the optimizer of [3]). Figure 5 presents the CDF of maximal link utilizations. We can see that *BGP-LWO* solves the problems presented in section IV and performs quite well. This means that it correctly predicts the egress points that will be chosen by the routers. This is quite logical as it was designed for, but these simulations confirm our expectations.

The mean reduction of maximal link utilization obtained by *BGP-LWO* is about 7.6% but can be as high as reducing the maximal link utilization from more than 140% to about 90%.

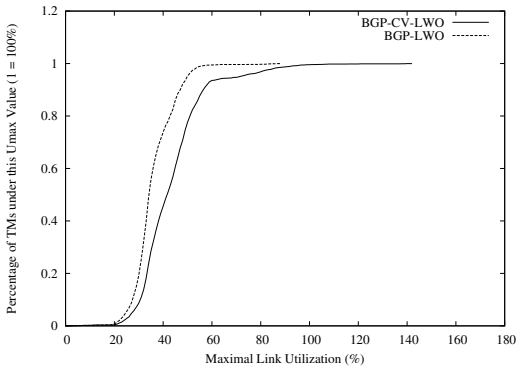

Fig. 5. CDFs of U_{max} over all TMs for *BGP-CV-LWO* and the *BGP-LWO* on the topology which contains a route reflector

C. Minimizing the number of path deflections

As explained in section V, *BGP-LWO* detects and forbids solutions that lead to intra-AS forwarding loops due to the combination of multiple path deflections. A network operator should also avoid (or at least minimize) simple deflections[9] inside its network as these could lead to inter-AS forwarding loops in the worst case.

To this end the methodology that we have adopted is the following. The optimizer is allowed to consider solutions with simple deflection during the execution of the algorithm. So it is easy to find an initial solution and to propose a move. But we have included in the objective function a penalty for each deflection. So during the execution of the algorithm the number of path deflections should decrease. If the penalty for each deflection is high enough the optimizer will first try to minimize the number of path deflections. If the number of path deflections is 0 at the end of such a simulation we know that a solution without path deflection exists. If it is not the case, we know the lower bound concerning the number of path deflections[10]. And if the solution that completely avoids path deflections is too bad concerning TE objectives, the network operator can choose (at his/her own risks) to allow some path deflections to improve the solution with respect to TE objectives, by tuning the value of the path deflection penalty.

We will see that on this dataset we are able to completely avoid path deflections while keeping good TE performance.

An execution of *BGP-LWO* without penalty associated with path deflections is shown on the left column of figure 6. These graphs present the evolution of different components of the objective function during the execution of the algorithm (from the first iteration -the leftmost point- to

[9]Here we define simple deflections as deflections that are not part of an Intra-AS forwarding loop.

[10]This is not absolutely true as the simulated annealing heuristic does not guarantee to find the **best** solution but only a **good** solution. So it may be possible that this execution gives a value which is not the **global** lower bound.

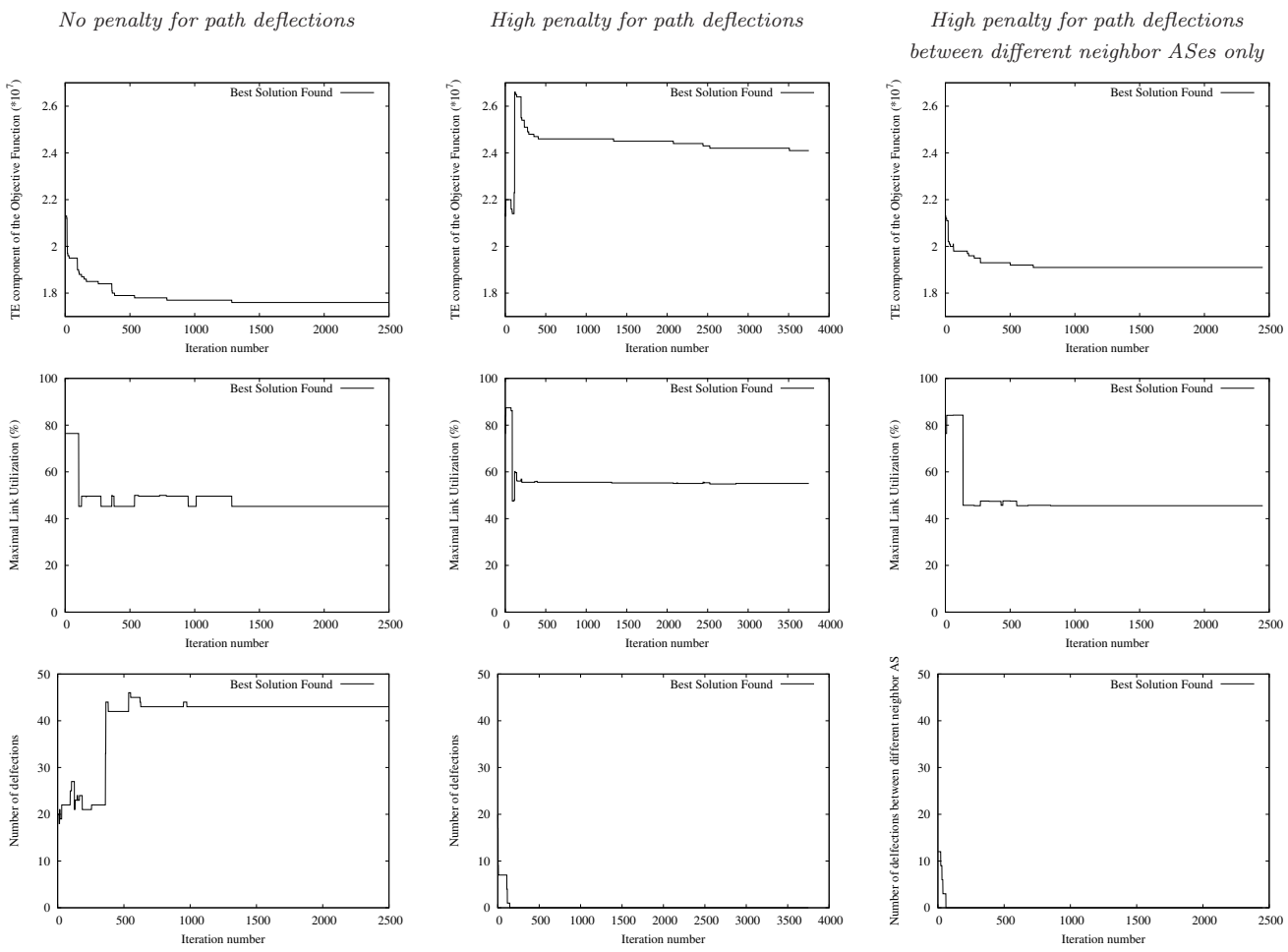

Fig. 6. Evolution of the Best Solution during the execution of the algorithm for one traffic matrix

the last one -the rightmost point-) for the best solution found so far. We see that the simulation stops after 2500 iterations. The first graph shows the value of the TE part of the objective function, which is supposed to reflect the quality of the load balance (the lower the value the lower the link utilizations). This is quite abstract as this value has no direct physical meaning. This is why we also present the maximal link utilization (the second graph), while this is not directly minimized by the optimizer. This value gives a physical idea of the TE quality of the solution. The third and last graph presents the number of path deflections. We see that the algorithm stops on a solution which induces 43 path deflections. We can conclude from this simulation that the best solution for TE induces a quite high number of path deflections.

We have also run *BGP-LWO* on the same traffic matrix while giving a very high penalty to each path deflection. The results of this execution should let us know if it is possible to find a solution (i.e. a set of link weights) inducing no deflection at all. Indeed the number of path deflections at the end of this simulation will give the lower bound concerning path deflections. The execution of the algorithm with these parameters is presented on the middle column of figure 6. The good news is that the algorithm rapidly finds a solution with no deflection (after about 200 iterations). During these first 200 iterations, the TE component of the objective function was not considered by the optimizer (second order of magnitude in the objective function) and the value of this component has been increased from about $2.2 * 10^7$ to $2.65 * 10^7$. After the 200^{th} iteration, as it is not possible anymore to decrease the number of path deflection (0 is obviously a lowest bound), the algorithm tries to minimize the TE component of the objective function while keeping the number of deflections to 0, which is less easy than when there were no constraint on the number of deflections. As a result, at the end of the simulation, the number of deflections for the best solution found is 0, but the corresponding values of TE component and maximal link utilization are quite significantly higher than in the case with no penalty for path deflections. This is quite logical as requiring no deflection limits the search space of the optimizer. It has to find the best TE solution from the set of solutions inducing no deflection while the preceding

	TE Objective Function ($*10^7$)	Maximal Utilization	Nb Deflections
No penalty	1.76	45.24%	43
	1.91	45.53%	9
	1.98	45.54%	3
High penalty	2.41	55.05%	0

TABLE I
TRADE-OFF BETWEEN LOAD BALANCE AND NUMBER OF DEFLECTIONS

execution of the algorithm could find the best TE solution from the complete set of solutions, no matter how many deflections would be induced.

Varying the value of the penalty associated with each deflection provides us trade-off solutions between a good TE state and a minimum number of path deflections. These solutions are found in table I. The first line (No penalty) is the solution found at the end of the execution of *BGP-LWO* for the left column of figure 6 while the last line (High penalty) is the solution found at the end of the execution of the *BGP-LWO* for the middle column of figure 6. We see that allowing a low number of deflections (3) allows the optimizer to improve the TE quality of the solution (from $2.41*10^7$ to $1.98*10^7$ for the TE component of the objective function or from 55.05% to 45.54% for the maximal link utilization).

D. Allowing deflections between two routes having the same ASPATH

As we have seen in section III, simple deflections between egress routers toward the same neighbor AS (if the two corresponding routes have the same ASPATH) is not a problem as this cannot potentially lead to forwarding loops. So we could take advantage of this point and avoid only path deflections between different ASes. This will put less constraints on the optimizer, which could allow it to find a better solution for TE. This is confirmed by the execution of the algorithm which is presented on the right column of figure 6. We can see that the optimizer finds in less iterations (2450 instead of 3750) a better solution ($1.91 * 10^7$ instead of $2.41 * 10^7$ for the TE component of the objective function and 45.52% instead of 55.05% for the maximal link utilization) while inducing no *dangerous* path deflections. This means that in some cases it may be interesting to be flexible with *non-dangerous* path deflections. Avoiding all path deflections may be too restrictive.

VII. Conclusion

Link Weight Optimizers try and minimize a traffic engineering objective function based on link utilizations. Therefore, for a precise optimization they need accurate estimations of all link utilizations resulting from the application of the optimized weights in the AS.

The first generation of LWOs was imprecise because these algorithms were based on the assumption that the intradomain traffic matrix was invariant when link weights were changed. This optimization thus neglected the effect of BGP's hot-potato rule, which may modify the intradomain traffic matrix, resulting in wrong estimations of actual link utilizations.

The second generation of LWOs did take BGP's hot-potato rule into account, but did not consider cases where routers have only partial BGP visibility due to route reflectors. The consequence is that these LWOs may wrongly predict the egress node for some traffic, and they do not take path deflection into account. The consequence is again an incorrect estimation of link utilizations.

Considering that complete visibility cannot always be affordable in every AS, we have studied in detail the impact of partial visibility on LWOs. We have shown that if route reflectors are present in the AS and lead to partial visibility, previously proposed BGP-aware LWO methods may behave poorly. Furthermore, these LWOs may also introduce path deflections, which in turn may lead to forwarding loops.

We have developed a new LWO algorithm, embedding the C-BGP simulator in its routing model, that always computes the correct link utilizations in any possible iBGP configuration. An additional asset of our proposed LWO is its ability to avoid path deflections when possible, or otherwise to minimize their number. In any case our LWO always avoids the creation of intra-AS forwarding loops due to multiple path deflections. The efficiency of our LWO with respect to other LWOs has been assessed on a real dataset from an operational network.

References

[1] http://totem.info.ucl.ac.be/.

[2] S. Agarwal, A. Nucci, and S. Bhattacharyya. Measuring the Shared Fate of IGP Engineering and Interdomain Traffic. In *Proc. IEEE ICNP*, November 2005.

[3] S. Balon and G. Leduc. Combined Intra- and Inter-domain Traffic Engineering using Hot-Potato Aware Link Weights Optimization. In *Proc. ACM SIGMETRICS*, June 2008.

[4] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. *RFC 2796*, April 2000.

[5] N. Feamster, H. Balakrishnan, and J. Rexford. Some Foundational Problems in Interdomain Routing. In *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[6] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proceedings of INFOCOM*, pages 519–528, 2000.

[7] T. G. Griffin and G. Wilfong. On the correctness of ibgp configuration. In *Proc. of ACM SIGCOMM*, pages 17–29, 2002.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):pp. 671–680, 1983.

[9] B. Quoitin and S. Uhlig. Modeling the routing of an Autonomous System with C-BGP. *IEEE Network*, 19(6), 2005.

[10] J. Rexford. *Handbook of Optimization in Telecommunications*, chapter Route optimization in IP networks. Springer Science + Business Media, February 2006.

[11] V. Van den Schrieck. Conception d'un langage flexible de définition de politiques de routage BGP. Master's thesis, Université Catholique de Louvain (UCL), Belgium, June 2005.

[12] M. Vutukuru, P. Valiant, S. Kopparty, and H. Balakrishnan. How to Construct a Correct and Scalable iBGP Configuration. In *Proceedings of INFOCOM*, Barcelona, Spain, April 2006.