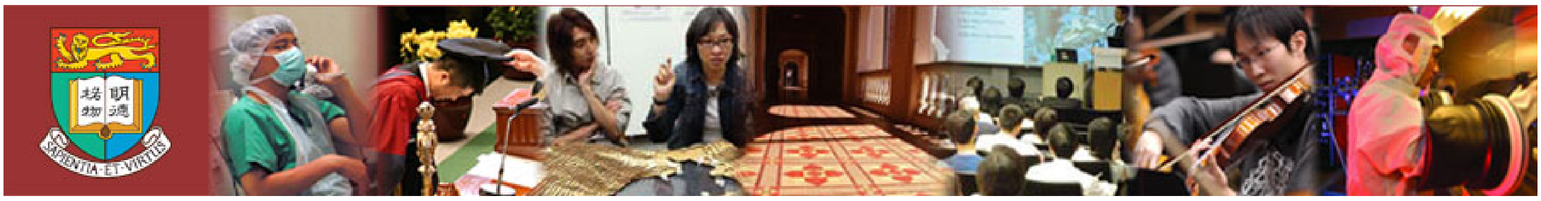
View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by HKU Scholars Hub

The HKU Scholars Hub The University of Hong Kong 香港大學學術庫



Title	BGP ingress-to-egress route configuration in a capacity-constrained AS
Author(s)	Chim, TW; Yeung, KL; Lui, KS
Citation	2005 Asia-Pacific Conference On Communications, 2005, v. 2005, p. 386-390
Issued Date	2005
URL	http://hdl.handle.net/10722/57271
Rights	©2005 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

2005 Asia-Pacific Conference on Communications, Perth, Western Australia, 3 - 5 October 2005.

BGP Ingress-to-Egress Route Configuration in a Capacity-constrained AS

Tat Wing Chim, Kwan L. Yeung and King-Shan Lui

Dept. of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Tel: (852) 2857-8493; Fax: (852) 2559-8738
E-mail: {twchim, kyeung, kslui}@eee.hku.hk

Abstract — **The BGP ingress-to-egress route configuration problem is to find a set of paths in an ISP to carry the transit flows, such that the amount of network resources consumed is minimized without violating the bandwidth constraint on all network links. To solve the problem, we first formulate it using Integer Linear Programming (ILP). Due to the high complexity involved in ILP, a heuristic algorithm, called MPPF, is then proposed. MPPF is designed based on the idea that heavily-loaded destination prefixes should be given higher priority to select less expensive edge links and routes. Simulation results show that MPPF requires less network resources and edge link capacity than an alternative heuristic called BTF.**

Keywords — BGP, Border Router Advertisements, Most Popular Prefix First

I. INTRODUCTION

A major responsibility of an Internet Service Provider (ISP) is to provide transit service for its neighbors. Traffic goes into and out of an ISP through a set of border routers which are managed by the ISP and are connected to its neighbors via a set of peering edge links. The problem of BGP ingress-to-egress route configuration is to determine a set of ingress-to-egress paths within the ISP's network to carry the transit traffic such that the network resources consumed is minimized.

To fully understand the mechanisms available for configuring ingress-to-egress routes, knowledge of Border Gateway Protocol (BGP) [1] is essential. In short, BGP is a path vector protocol under which routing decisions can be made based on policy. BGP divides the Internet into a collection of Autonomous Systems (ASes). An AS is defined as a set of routers under a single technical administration, e.g. an ISP. ASes exchange routing reachability information through external BGP peering sessions. A border BGP router receives route advertisements from either external peers (in neighboring ASes) or internal peers (in local AS). Each advertisement contains a destination prefix, an IP address of the next-hop, a multi-exit discriminator (MED) and a list of ASes leading to the destination prefix. Some or all of these received routes will be included into its own routing table and advertised to its peers based on a set of decision criteria [1].

When an AS propagates an advertisement to an external peer, the MED field can be used to indicate the preference of accepting incoming traffic at a particular ingress edge (thus ingress router) and the smaller the MED value, the higher the preference. So properly setting the MED field can help to balance the network load. In general, no matter where the transit traffic arrives, as long as it has the same destination prefix, it will be sent to the same selected egress link.

Related work on optimizing the performance of an ISP can be found in some recent literatures. In [2] and [3], the problem of intra-domain path selection is studied and several OSPF weight assignment algorithms are proposed for setting up peer-to-peer intra-domain paths. In [4], a border router and intra-domain link placement problem is examined. The solution is useful at the development phase of an ISP. In [5], the problem of BGP route configuration is investigated under the assumption that an ISP cannot control the ingress points of the transit traffic and all ingress edge links and intra-domain links have unlimited bandwidth. This problem is proved to be NP-hard. Their proposed heuristic algorithm is based on the Integer Linear Programming (ILP) formulation and is quite complicated. In [6], a time-efficient algorithm based on the idea of most popular prefix first is designed.

In this paper, we extend the model for BGP egress selection [5][6] to take advantages of the MED option in BGP. This allows a local AS to recommend certain ingress edge/router to its neighbors for accepting their incoming transit traffic. The resources consumed by the local AS can be further reduced. Since the assumption of infinite capacity on intra-domain links [5][6] may not be realistic, links with finite bandwidth are considered in this paper. A new heuristic algorithm called MPPF is proposed. It is based on the similar idea of [6] that heavily-loaded destination prefixes should be given higher priority to select less expensive ingress and egress edge links and paths. This allows more traffic flows to go through less expensive paths which can potentially minimize the network resources consumed.

In the next section, the route configuration problem is formally defined. In Section III, we show that the problem can be solved using ILP. In Section IV, the heuristic algorithm MPPF is presented and its performance is evaluated in Section V. Finally, we conclude the paper in Section VI.

II. PROBLEM FORMULATION

Assume that all intra-domain links and edge links connecting to other ASes have finite capacities and are bidirectional. The capacity allocated to each direction is pre-determined and dedicated. Let $G = (V, E)$ denote the intra-domain topology where $V = \{1 .. N\}$ is the set of routers, and E is the set of intra-domain links. Among N routers, X of them are border routers. The set of edge links is given by $B = \{b_1, \ldots, b_l\}$ and $R(b_i)$ returns the router associated with edge link b_i.

Multiple edge links may be connected to the same border router. A neighboring AS may be connected to the local AS through a direct edge link, or indirectly via other ASes. Each neighbor may be connected to the local AS through multiple edge links. For simplicity, we assume that the prefixes received by the AS are non-overlapping. We further assume that route advertisements for any prefix are advertised to all

This work is supported by Hong Kong Research Grant Council Earmarked Grant HKU 7048/02E.

connecting neighbors so that neighbors are able to choose which ingress points to use (based on the MED value in the advertisements).

Given a set of neighbors $A = \{A_1, \ldots, A_H\}$. For each neighbor A_h, let $In(h)$ denote the set of edge links through which A_h may send in the transit traffic. External BGP peering sessions at the border routers receive advertisements for network prefixes across the edge links. Let $P = \{P_1, \ldots, P_K\}$ denote the set of prefix advertisements received across all edge links. For each prefix P_k, let $Out(k)$ denote the set of edge links at which an advertisement for P_k has been received.

Each edge link b_i has an ingress capacity constraint $C_i^{ingress}$ and an egress capacity constraint C_i^{egress}. The amount of residual bandwidth on ingress and egress links is denoted by $w_i^{ingress}$ and w_i^{egress} respectively. Let l denote a direct link in E and let d_l, C_l and w_l denote the cost per unit traffic, capacity constraint and current residual capacity of that link respectively. Also let $p = [a, b]$ denote a path connecting routers a and b in V and let d_p denote the cost per unit traffic along p, i.e.

$$d_p = \sum_{l \in p} d_l \tag{1}$$

Problem Statement

Let $t(h, k)$ be the volume of the traffic flow from neighbor A_h to destination prefix P_k. Further denote f as the ingress-to-egress route configuration function. For the flow from A_h to P_k, $f(h, k)$ returns an ingress edge $igr(h, k) \in In(h)$, an egress edge $egr(h, k) \in Out(k)$, and a complete intra-domain path from the router associated with $igr(h, k)$ to the router associated with $egr(h, k)$. For convenience, we write $f(h, k) = <igr(h, k), egr(h, k), p(h, k)>$ where $p(h, k) = [R(igr(h, k)), R(egr(h, k))]$.

Our goal is to minimize the total amount of network resources consumed for carrying the transit traffic

$$\sum_{h=1}^{H}\sum_{k=1}^{K} t(h,k)\cdot d_{p(h,k)} \tag{2}$$

and $f(h, k) = <igr(h, k), egr(h, k), p(h, k)>$ satisfies the following constraints:

- No violation on ingress edge link capacity, i.e. $\sum_{h,k:igr(h,k)=l} t(h,k) \le C_l^{ingress}\ \forall l \in B$;
- No violation on egress edge link capacity, i.e. $\sum_{h,k:egr(h,k)=l} t(h,k) \le C_l^{egress}\ \forall l \in B$;
- No violation on intra-domain link capacity, i.e. $\sum_{h,k:l \in p(h,k)} t(h,k) \le C_l\ \forall l$;
- The same egress edge link is assigned to all transit traffic flows going to the same destination prefix, i.e., for each prefix P_k, $egr(h, k) = l\ \forall h$ for some l.

III. Integer Linear Programming Formulation

To facilitate our problem formulation, we introduce a flat network topology $G' = (V', E')$. Details of $G' = (V', E')$ are defined below:

- $V' = V \cup \{(N+h) \mid A_h \in A\} \cup \{(N+H+k) \mid P_k \in P\}$
- $E' = E \cup \{(N+h, R(e)) \mid A_h \in A, e \in In(h)\} \cup \{(R(e), N+H+k) \mid P_k \in P, e \in Out(k)\}$ $\forall (i,j) \in E'$
- $c_{(i,j)} = \begin{cases} C_l, & \forall l \in E \\ C_e^{ingress}, & N+1 \le i \le N+H, e \in In(i-N), j = R(e) \\ C_e^{egress}, & e \in Out(j-N-H), i = R(e), N+H+1 \le j \le N+H+K \end{cases}$ $\forall (i,j) \in E'$
- $d_{(i,j)} = \begin{cases} d_l, & \forall l \in E \\ 0, & \text{otherwise} \end{cases}$

That is, nodes 1 to N represent the set of routers in the local AS, nodes $N + 1$ to $N + H$ represent the set of neighboring ASes (A_1 to A_H), nodes $N + H + 1$ to $N + H + K$ represent the set of destination prefix networks (networks having prefixes P_1 to P_K). Note that the cost for the ingress and egress edge links are all set to zero so that they will not affect our intra-AS cost calculations. Let X_{ij}^{hk} represent the percentage of the traffic from neighbor A_h to destination prefix P_k that flows across link (i, j) in E'. The ILP formulation for the ingress-to-egress route configuration problem is given below.

ILP Formulation

$$\min\left(\sum_{h=1}^{H}\sum_{k=1}^{K}\sum_{(i,j)\in E'} t(h,k)\cdot d_{(i,j)}\cdot X_{ij}^{hk}\right) \tag{3}$$

s.t.

$$\sum_{j:(i,j)\in E'} X_{ij}^{hk} - \sum_{j:(j,i)\in E'} X_{ji}^{hk} = 0, A_h \in A, P_k \in P, i \ne N+h \text{ and } i \ne N+H+k \tag{4}$$

$$\sum_{j:(i,j)\in E'} X_{ij}^{hk} - \sum_{j:(j,i)\in E'} X_{ji}^{hk} = 1, A_h \in A, P_k \in P, i = N+h \tag{5}$$

$$X_{ij}^{hk} = 0 \text{ or } 1, \forall (i,j) \in E' \tag{6}$$

$$\sum_{h=1}^{H}\sum_{k=1}^{K} t(h,k)\cdot X_{ij}^{hk} \le c_{(i,j)} \tag{7}$$

$$X_{i(N+H+k)}^{h_1 k} = X_{i(N+H+k)}^{h_2 k}, A_{h_1}, A_{h_2} \in A, P_k \in P, i \in [1, N] \tag{8}$$

The objective function (3) is to minimize the resources consumed. (4) and (5) are flow conservation constraints. (4) states that the amount of traffic flowing into a node has to be equal to the amount of traffic flowing out of the node for non-source and destination nodes. (5) states that the net flow out of a source node is 1. (6) restricts the X_{ij}^{hk} variables to be either 0 or 1. (7) is the link capacity utilization constraint. (8) ensures that the same egress edge link is assigned to all transit traffic flows going to the same prefix. Noted that this constrain also differentiates the formulation from those for the well-known multicommodity unsplittable flow problem [7].

Though the ingress-to-egress route configuration problem can be solved by ILP, the process is too complicated for any realistic sized problems.

IV. Heuristic Algorithm – MPPF

A. MPPF Algorithm

Let p_k be the total amount of traffic destined to prefix P_k. Our proposed algorithm aims at giving the prefix with the largest amount of traffic destined to it, i.e. $\max_k\{p_k\}$, the highest route selection priority. The idea is that if no priority is given to the prefix with the largest p_k value, it is very likely that the most desirable egress link leading to this prefix together with the corresponding intra-domain route would have been occupied by others. The potential extra cost of carrying this traffic on alternative egress links together with the corresponding intra-domain route would be very high. Since the route configuration priority is based on p_k, we call our algorithm Most Popular Prefix First (MPPF).

The pseudo code of MPPF is listed in Fig. 1. Traffic flows $t(t_1, t_2)$ are first sorted in non-increasing order of their volumes to form an ordered list **T** (Step 4). Then prefixes (P_k) are sorted in non-increasing order of prefix traffic volume p_k to form an ordered list **K** (Step 6). For each prefix in list **K** (Step 8), we determine the egress edge (from the set of egress edges which have advertisements for that prefix) which leads to the minimum incurred cost (Steps 8.1 to 8.3). This is done by considering the egress links one by one. During the consideration of each egress link, we compute the corresponding minimum incurred cost by calling the function *Cost_Prefix*() in Fig. 2. When a minimum egress link is found, the same selection process as in function *Cost_Prefix*() is used to perform the actual assignments/configurations (Step 8.4). Note that all these procedures are designed to give priority to prefixes with larger traffic volumes.

```
                              MPPF Algorithm
Input Parameters
G; t(h, k) for all h, k;  d_l for all l
Output Parameter
f

1.    For all i, set $w_i^{ingress} = C_i^{ingress}$   /* Initialize ingress residual capacities. */
2.    For all i, set $w_i^{egress} = C_i^{egress}$   /* Initialize egress residual capacities. */
3.    For all l, set $w_l = C_l$              /* Initialize link residual capacities. */
4.    Sort $(t_1, t_2)$ in non-increasing order of $t(t_1, t_2)$ to form an ordered list T.
5.    Compute $p_k = \sum_{h=1}^{H} t(h,k) \; \forall k \in [1, K]$.
6.    Sort k in non-increasing order of $p_k$ to form an ordered list K.
7.    For all (h, k), set f(h, k) = null.   /* Initialize assignments */
8.    For each k in the ordered list K {
8.1        MinEgress = 0          /* MinEgress = null, i.e. 0 by default. */
8.2        MinEgressCost = ∞  /* MinEgressCost = ∞ by default. */
8.3        For each egress edge $b_j$ in Out(k)
               if (Cost_Prefix (G, k, j, $w^{ingress}$, w, T) < MinEgressCost)
               & $(w_j^{egress} - p_k \ge 0)$ {
                   MinEgress = j
                   MinEgressCost = Cost_Prefix (G, k, j, $w^{ingress}$, C, T)
               }
8.4        For each $t(t_1, k)$ in the ordered list T {
               MinIngress = 0
               MinIngressCost = ∞
               MinPath = null
                For each ingress edge $b_i$ in $In(t_1)$
                   if $(w_i^{ingress} - t(t_1,k) \ge 0)$ & $(i \ne MinEgress)$ {
                           Shortest_Path (G, R(i),
                           R(MinEgress), $t(t_1, k)$, w, δ, Path)
                           If $(t(t_1, k) * d_{Path} < MinIngressCost)$ {
                                   MinIngress = i
                                   MinIngressCost
                                   $= t(t_1, k) * d_{Path}$
                                   MinPath = Path
                           }
                   }
           }
           $f(t_1,k) = < MinIngress, MinEgress, MinPath >$
           $w_{MinIngress}^{ingress} = w_{MinIngress}^{ingress} - t(t_1,k)$
           $w_{MinEgress}^{egress} = w_{MinEgress}^{egress} - t(t_1,k)$
           $w_l = w_l - t(t_1,k) \quad \forall l \in MinPath$
      }
```

Fig. 1 Main algorithm in the MPPF algorithm

Function *Cost_Prefix*() in Fig. 2 is for computing the minimum cost for forwarding all traffic flows for prefix P_k to egress edge b_j. It imports the sorted lists **T** and examines traffic flows going to prefix P_k one by one in non-increasing order of their traffic volumes (Step 4). Each flow is tentatively assigned to a minimum possible ingress link together with an ingress-to-egress path (Step 4.4) making use of the function *Shortest_Path*() (summarized in Fig. 4 for completeness). Note that the assignments at this stage are tentative in the sense that the actual ingress residual capacities ($w^{ingress}$ and w) will not be updated. This is because we still need to compare the costs of using other egress links. So two running vectors $vw^{ingress}$ and vw are used in Step 2 and Step 3 for storing the tentative residual capacities.

```
Function Cost_Prefix (G, k, j, $w^{ingress}$, w, T)
Input Parameters
G:              local AS topology
k:              prefix $P_k$ under consideration
j:              egress edge $b_j$ under consideration
$w^{ingress}$:  residual capacity of all ingress edges
w:              residual capacity on all links
T:              a sorted list that contains traffic flows t(h, k) in non-increasing
                order
Output Parameter
Minimum cost for forwarding all traffic for prefix $P_k$ using egress edge $b_j$

1.     MinCost = 0                        /* Initial MinCost to 0. */
2.     For all i, set $vw_i^{ingress} = w_i^{ingress}$  /* Store virtual ingress capacities. */
3.     For all l, set $vw_l = w_l$               /* Store virtual link capacities. */
4.     For each $t(t_1, k)$ in the ordered list T {
4.1         MinIngress = 0          /* MinIngress = null, i.e. 0 by default. */
4.2         MinIngressCost = ∞   /* MinIngressCost = ∞ by default. */
4.3         MinPath = null           /* MinPath = null by default. */
4.4         For each ingress edge $b_i$ in $In(t_1)$
                    if $(vw_i^{ingress} - t(t_1,k) \ge 0)$ & $(i \ne j)$ {
                          Shortest_Path (G, R(i), R(j), $t(t_1, k)$, vw, Path)
                          If $(t(t_1, k) * d_{Path} < MinIngressCost)$ {
                                   MinIngress = i
                                   MinIngressCost = $t(t_1, k) * d_{Path}$
                                   MinPath = Path
                          }
                    }
            MinCost = MinCost + MinIngressCost
            $vw_{MinIngress}^{ingress} = vw_{MinIngress}^{ingress} - (t_1,k)$
            $vw_l = vw_l - t(t_1,k) \quad \forall l \in MinPath$
       }
       return MinCost
```

Fig. 2 Function *Cost_Prefix*() in the MPPF algorithm

```
Function Shortest_Path (G, Src, Dest, Vol, w, Path)
Parameters
G:       local AS topology
Src:     source node
Dest:    destination node
Vol:     volume of the traffic instance under consideration
w:       an array that stores residual capacities of all links
Path:    the shortest wide path found

1.    VG = (VV, VE) = G = (V, E)
2.    $vw_l = w_l - Vol \quad \forall l \in VE$
3.    Prune out link l from VG if it has insufficient resources, i.e. $vw_l < 0$
      $\forall l \in VE$
4.    Find the shortest path Path connecting Src and Dest using Dijkstra's
      algorithm on VG.
```

Fig. 3 Function *Shortest_ Path*() in the MPPF algorithm

B. Time Complexity

The sorting in Step 4 of Fig. 1 requires a worst case time complexity of $O(HK\log(HK))$, where H is the number of neighbors and K is the number of prefixes. The sorting in Step

6 requires a worst case time complexity of $O(K \log K)$. Step 8 is the most expensive step and it invokes function *Cost_Prefix*() many times.

Function *Cost_Prefix*() in turn goes through at most HK traffic flows and for each traffic flow, we have to go through I ingress edges and for each ingress edge, function *Shortest_Path*(), which is based on Dijkstra's Algorithm, is invoked. Let N be the number of nodes in the topology, the worst case time complexity of Dijkstra's Algorithm is known to be $O(N \log N + |E|) \sim O(N \log N)$ since $|E| \sim N$ in a BGP environment. As a result, the time complexity of function *Cost_Prefix*() is $O(HKIN \log N)$.

As Step 8.3 iterates function *Cost_Prefix*() I times, its time complexity is $O(HKI^2N \log N)$ and it is the most expensive substep in Step 8. Therefore, the running time of MPPF is $O(HK^2I^2N \log N)$.

C. Implementation using MPLS

Current OSPF intra-domain routing protocol is shortest-path based. However, due to capacity constraints on internal links, the paths returned by MPPF may not be the shortest. To allow a flow following a non-shortest path, the technique of OSPF weight assignment [2][3] or the technique of Multi-Protocol Label Switching (MPLS) [8] can be used. Since not all paths can be specified by setting OSPF weights [9], MPLS is preferred. With MPLS, each path found by the MPPF algorithm is specified by a logical *label-switched path*. Upon receiving a transit traffic flow, a border router checks its BGP routing table to determine which egress edge (thus egress border router) should be used. Then it forwards the flow to the selected egress border router using a predetermined label-switched path.

As a final remark, MPPF algorithm has assumed that neighboring ASes will always honor the ingress edge recommendations made by the local AS. In reality, this may not be the case, possibly due to the conflict of interests among neighboring ASes. We will study the impact of such factors in the future.

V. Performance Evaluations

A. Network Model

Consider a local AS with $N - X$ internal and X border routers, H neighboring ASes and K destination prefixes for transit flows. The network topology for simulation is generated as follows:

- A topology containing N nodes with cost per unit traffic on the direct link between any two nodes in the range {1…10} is generated by BRITE [10]; X nodes are picked up randomly to become border routers. Assume $d_{(i,j)} = d_{(j,i)}$ for any two routers r_i and r_j.
- The multihoming degree of each border router is randomly selected from 1 to 3. Each border router is then associated with the corresponding number of edge links. All edge links are uniquely numbered to form the edge set.
- The size of set $In(h)$ for each neighbor A_h is randomly selected from 2 to 3. The elements of $In(h)$ are randomly selected from the edge set.
- For each prefix P_k, the size of $Out(k)$ is randomly selected from 2 to 5. The elements of $Out(k)$ are again randomly selected from the edge set.

B. Traffic Model

Assume that every neighboring AS has some traffic destined for every destination prefix. So there are altogether $H \times K$ traffic instances/flows, which forms an $H \times K$ traffic matrix. Each entry of the matrix represents the traffic volume of a flow. Its value is uniformly distributed between real numbers 0 and 20. Looping is not allowed. So if a neighboring AS has forwarded an advertisement for prefix P_k to the local AS, this AS cannot inject traffic for prefix P_k into the local AS.

Two sets of simulations are conducted based on the parameters used in [5]. In particular, simulation results based on ($N = 100$, $X = 25$, $H = 12$, $K = 35$) are plotted in Fig. 4 and Fig. 5. Simulation results based on ($N = 100$, $X = 30$, $H = 17$, $K = 110$) are shown in Fig. 6 and Fig. 7. Each point of simulation results is obtained by taking the average of 20 independent experiments each with a randomly generated BRITE network topology and traffic matrix.

C. MPPF vs other algorithms

Besides MPPF algorithm, the following algorithms are implemented for comparison:

- BTF (Biggest Traffic First): BTF is an alternative heuristic that might be used by an ISP [5]. In BTF, the set of traffic instances are sorted in non-increasing order. For each traffic instance $t(h, k)$, an attempt is made to search for the best shortest path with sufficient capacity connecting the sets $In(h)$ and $Out(k)$. The ingress and egress links are returned automatically. If prefix P_k has already been assigned to a particular egress link and which still has sufficient egress capacity, the search is limited to paths going to the same egress link only. Otherwise, the search is not limited by any egress link.
- MPPF_NI (Most Popular Prefix First with No Ingress Recommendation). MPPF_NI is the same as MPPF except that ingress edges of all transit traffic flows are determined by neighboring ASes and the local AS does not make any recommendations.
- ICS (Infinite Capacity Solution): This is obtained by assuming all ingress capacities, egress capacities and internal link capacities are infinite. It serves as a lower bound.

Note that the algorithms proposed in [5] cannot be extended to cover the ingress-to-egress route configuration problem. In [5], the ingress edges of the traffic flows are assumed to be known in advance and so traffic flows going to the same prefix can be aggregated into a job as in the generalized assignment problem [11]. Also the path used is assumed to be always the shortest path and intra-domain link capacities are assumed to be infinity. However, in our ingress-to-egress route configuration problem, all these assumptions do not hold.

Assume the initial unidirectional capacity of all ingress links and all egress links are fixed to *ExtBW*. Also assume that the initial unidirectional capacity of all intra-domain links are fixed to *IntBW*. The performance of using two different values of *IntBW* is shown in Fig. 5. Fig. 4 and Fig. 6 show the percentage of traffic sent against *ExtBW*. Fig. 5 and Fig. 7 show the solution cost, which is normalized by the solution cost obtained using ICS algorithm, against *ExtBW*.

From Fig. 4 and Fig. 6, we can see that MPPF requires lower edge capacity than BTF to forward 100% of traffic. In particular, when edge capacity is not enough (< 600 in Fig. 4 and < 1200 in Fig. 6), our MPPF algorithm can forward up to

9.14% and 25% more traffic than BTF algorithm in the two different parameter settings respectively.

From Fig. 5 and Fig. 7, we can see that the normalized solution cost for BTF is worse than that for MPPF. In fact, the normalized solution cost for BTF cannot converge to that of ICS. We can see that the normalized solution cost for BTF is about 10% higher than that of MPPF. For MPPF_NI, since there is no ingress edge recommendation, its normalized solution cost is 35% - 40% higher than MPPF.

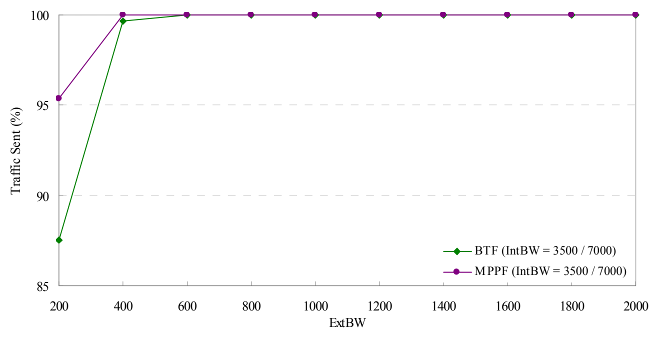


Fig. 4 Percentage of traffic sent ($N = 100$, $X = 25$, $H = 12$ and $K = 35$)

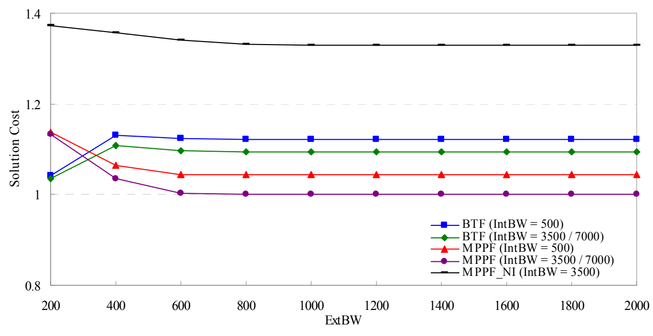


Fig. 5 Normalized solution cost ($N = 100$, $X = 25$, $H = 12$ and $K = 35$)

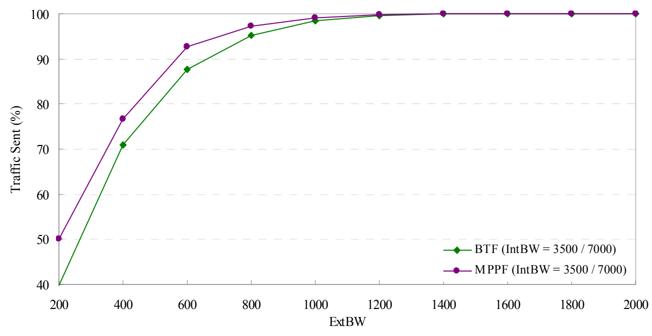


Fig. 6 Percentage of traffic sent ($N = 100$, $X = 30$, $H = 17$ and $K = 110$)

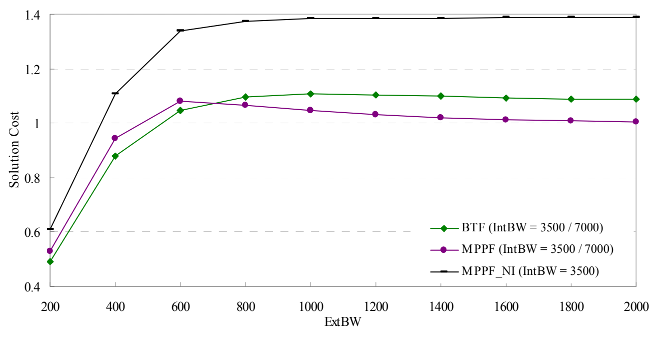


Fig. 7 Normalized solution cost ($N = 100$, $X = 30$, $H = 17$ and $K = 110$)

VI. CONCLUSIONS

The BGP ingress-to-egress route configuration problem is to find a set of paths in an AS/ISP to carry the transit flows, such that the amount of network resources consumed is minimized without violating the bandwidth constraint on all network links. To solve the problem, we first formulated it using Integer Linear Programming (ILP). Due to the high complexity involved in ILP, a heuristic algorithm, called MPPF, was then proposed. MPPF was designed based on the idea that heavily-loaded destination prefixes should be given higher priority to select less expensive edge links and routes.

REFERENCES

[1] Y. Rekhter and T. Li, “A border gateway protocol 4,” Internet-Draft (RFC1771), February 1998.

[2] Y. Wang, Z. Wang and L. Zhang, “Internet Traffic Engineering without Full Mesh Overlaying,” *Proceedings of IEEE INFOCOM*, vol. 1, pp. 565 – 571, April 2001.

[3] B. Fortz and M. Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights,” *Proceedings of IEEE INFOCOM*, vol. 2, pp. 519 – 528, March 2000.

[4] F. Lam, W. C. Lau and V. O. K. Li, “Inter-Domain Router Placement and Traffic Engineering,” *Proceedings of IEEE ICC*, vol. 4, pp. 2443 - 2448, May 2002.

[5] T. C. Bressoud, R. Rastogi and M. A. Smith, “Optimal Configuration for BGP Route Selection,” *Proceedings of IEEE INFOCOM*, vol. 2, pp. 916 – 926, April 2003.

[6] T. W. Chim and K. L. Yeung, “Time-Efficient Algorithms for BGP Route Configuration,” *Proceedings of IEEE ICC*, June 2004.

[7] C. Chekuri, S. Khanna and F. B. Shepherd, “The All-or-Nothing Multicommodity Flow Problem,” *Proceedings of ACM STOC*, pp. 156 – 165, June 2004.

[8] E. Rosen, A. Viswanathan and R. Callon, “Multiprotocol Label Switching Architecture,” Internet-Draft (RFC3031), January 2001.

[9] J. L. Sobrinho, “Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, August 2002.

[10] http://www.cs.bu.edu/brite/

[11] D. B. Shymoys and E. Tardos, “An approximation algorithm for the generalized assinment problem,” *Mathematical Programming A*, vol. 62, pp. 461 – 474. 1993.