Low complexity bandwidth guaranteed routing algorithms using path holding time

Cao Thai Phuong Thanh
Saigon University
Ho Chi Minh City, Vietnam
Email: ctpthanh@sgu.edu.vn

Ha Hai Nam, Tran Cong Hung
Post & Telecommunications Institute of Technology
Vietnam
Email: namhh@ptit.edu.vn, conghung@ptithcm.edu.vn

***Abstract*—This paper introduces new bandwidth guaranteed routing algorithms using remaining path holding time. The idea is to combine residual bandwidths with future available bandwidths which are calculated based on holding time of routing paths to select next routes. The proposed algorithms have low complexity because only link bandwidths are considered and no critical computation is needed. They are tested against other popular traffic engineering (TE) routing algorithms, i.e. Minimum Hop Algorithm (MHA), Minimum Interference Routing Algorithm (MIRA), and Random Race based Algorithm for TE (RRATE). Experimental results indicate that the proposed algorithms not only accept more number of routing requests but also achieve lower computation time than all the others.**

***Keywords*-bandwidth guaranteed routing; traffic engineering; path holding time;**

I. Introduction

Today network service providers want to optimize their network resources while still satisfy as much quality of service demand as possible. Therefore, traffic engineering (TE) routing algorithms that select routes for the dual purposes of network optimization and constraint satisfaction have become important. Besides the basic Minimum Hop Algorithm (MHA) [1], many dynamic routing algorithms including Minimum Interference Routing Algorithm (MIRA) [2] and Random Race based Algorithm for TE (RRATE) [3] have been proposed and successfully improve the routing performance. The improvement results from considering states of networks such as link residual bandwidths, ingress-egress pair information when selecting routes. Additionally, link criticality is defined as a measure of link interference with potential paths and critical values are then used to avoid link congestion. However, the calculation of critical values increases the algorithm complexities that results in the increase of time needed to handle routing requests.

This paper introduces a novel idea to use holding time, the duration in which bandwidth resource is reserved along the paths, to obtain future available bandwidths, then combined with residual bandwidths to form a simple but effective values for route selection. Hence, the proposed algorithm is named Bandwidth Guaranteed routing algorithm using Holding Times (BGHT). The rest of the paper is organized as follows. In section 2, after problem definitions and notations, related works are briefly introduced. Section 3 proposes two variations of the algorithm: BGHT1 and BGHT2 in details. Experimental results in term of request accepted ratio and average computing time are described in section 4. Finally, section 5 gives conclusion.

II. Problem definition & related works

A. Problem definition

A network topology with n nodes and m links is considered. Each link has its own capacity and residual bandwidth at a given time. Traffic requests, which require certain bandwidths from ingress nodes to egress nodes, are sequentially handled by a routing algorithm. Each traffic demand if accepted will hold a part of bandwidth resource along its path for a period of time called the holding time. Moreover, network information such as topology, residual bandwidth, ingress egress pairs is assumed available for routing algorithms but future routing requests are now known. Table I lists mathematical notations used to describe the problem.

Table I: Problem notations

Symbol	Description
$G(N,L)$	A direct graph presents the network topology $N(\|N\|=n)$ is a set of nodes $L(\|L\|=m)$ is a set of links
$c(l)$	Capacity bandwidth of link l
$r(l)$	Residual bandwidth of link l
$d(s,d,b,h)$	A traffic request from ingress node s to egress node d. It requires b bandwidth units in h time units.
p_{sd}	A routing path from s to d

The goal of routing algorithms is to accept as many routing requests as possible given that established paths must satisfy certain bandwidth requirements.

$$\begin{cases} \text{Maximize number of satisfied requests} \\ \text{subject to} \\ \text{Find } p_{sd} \text{ for } d(s,d,b,h)/\forall l \in p_{sd} : r(l) \geq b \end{cases} \tag{1}$$

Since ingress-egress pairs have commodity integral flows (i.e. common links as well as sequences of links) and routing requests are not known in prior, the problem of maximization of accepted requests is NP-hard [4]. Generally, TE routing algorithms weigh links (by different methods) then apply the shortest path algorithm such as Dijkstra to select the least weighted path for a request. Table II generalizes steps of such algorithms.

Table II: General TE routing algorithm steps [5]

Input	A network graph $G(N, L)$ with sets of links and residual bandwidths A traffic demand $d(s, d, b, h)$
Output	A satisfied bandwidth path from s to d, p_{sd}, toward the optimal goal in (1) Or no route satisfying the request
General algorithm	1. Calculate link weights $w(l)$ 2. Temporarily remove links that have residual bandwidth less than b 3. Find the least weight path p_{sd}

B. *Related works*

The simplest routing solution is Minimum Hop Algorithm (MHA), which selects the shortest path between the source and destination nodes. Because links of the shortest paths are always used, they are quickly congested whereas other links are underutilized. Consequently, further routing requests cannot be satisfied due to network congestion. To address this issue, the Minimum Interference Routing Algorithm (MIRA) [2] defines link criticality by maxflow-mincut characteristics. Specifically, when a routing request arrives, MIRA identifies mincut set for each ingress egress pair. Links that belong to those mincut sets are called critical because if they are used, the maxflows of corresponding pairs are decreased. The more critical the links are, the more weight they are assigned. Then, Dijkstra algorithm determines the least weight route which is considered as the route having minimum interference with future requests. Evaluations confirm that MIRA accepts much more requests than MHA. In contrast, the computing time of MIRA is also orders of magnitude greater than one of MHA due to the maxflow-mincut calculation.

In Random Race based Algorithm for Traffic Engineering (RRATE) [3], a machine learning technique called random race is applied in order to reduce the computation time. Generally, k pre-selected paths of an ingress egress pair 'race' together in the aspect of the number of satisfied requests. After such learning phase, routes of the corresponding pair are selected among those k paths based on their racing values instead of continuing to compute link weights. Therefore, RRATE requires less time than MIRA while still accepts high number of requests. A thorough study of the above algorithms as well as other bandwidth guaranteed TE routing solutions can be found in [5].

III. Proposed algorithms

In order to maximize the number of satisfied demands, TE routing algorithms attempt to balance the network load via dynamic link weights. Despite various weight calculations, residual bandwidths which reflect the network availability for further requests always constitute an important part of the weight. The proposed BGHT algorithms calculate weights of links based on not only their residual bandwidths but also future available ones. The latter values are obtained using the path holding time.

Specifically, when a request r_i arrives at a time t_i, a period of time Δt_i is determined. Based on remaining holding times of existing paths, an amount of bandwidth that will be released after the Δt_i period (i.e. at the time $(t_i + \Delta t_i)$) is computed for each link. Such to-be-released bandwidths are then combined with the residual bandwidths to form link weights as follows:

$$w(l) = \frac{1}{r(l) + relBw(l)} \tag{2}$$

$relBw(l)$:bandwidth released from link l after Δt_i

Two ways of determining Δt_i lead to the BGHT1 and BGHT2 algorithms. In BGHT1, Δt_i is determined as an interval between the current request r_i and the next request r_{i+1}. Because there is no prior knowledge of future demands, triangular distribution [6] is employed to estimate when the request r_{i+1} arrives. Particularly, once a request arrives, the min, the max, and the average arrival intervals are computed, then Δt_i is obtained by a statistical generator following the triangular distribution. According to such Δt_i estimation, BGHT1 greedily selects paths that have as wide bandwidth as possible when the next request r_{i+1} happens.

Considering that if the arrival time is short but the holding time is relatively long then all links may not release any bandwidth after the Δt_i period of BGHT1 (i.e. $\forall l, relBw(l) = 0$). Therefore, the BGHT2's Δt_i is determined as the minimum remaining holding time of existing paths. It means, in BGHT2 weight calculation, there are always links that have future bandwidth values $relBw(l)$ greater than zero. In other words, BGHT2 calculates available bandwidths after having a path that releases bandwidths. Whereas BGHT1 estimates available bandwidths when the next request is expected to arrive. Table III summarizes steps of the BGHT algorithms

Table III: The BGHT algorithm steps

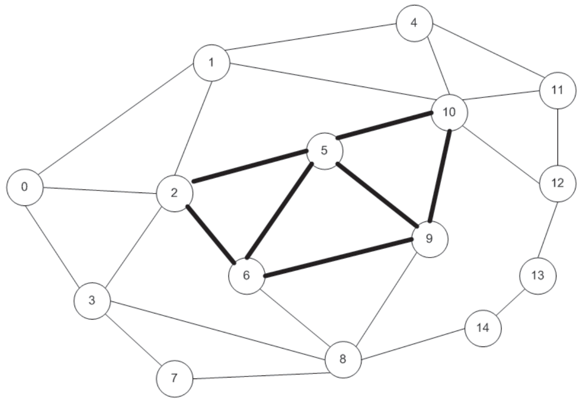
Input	A network graph $G(N, L)$ A traffic request $d(s, d, b, h)$
Output	A selected path p_{sd} Or no path
BGHT algorithms	*For each $d(s, d, b, h)$:* 1. Determine Δt_i *BGHT1*: compute the min, the max, and the average intervals of the demands then generate Δt_i using the triangular distribution. *BGHT2*: Δt_i is equal to the minimum remaining holding time of the existing paths 2. For all links l in L, calculate $w(l)$ using equation (2) 3. Temporarily remove links whose $r(l) < b$ 4. Apply Dijkstra to find the least weight path from s to d If found: return the path p_{sd} and update necessary information. Else: reject the request.

The use of BGHT algorithms has two advantages. Firstly, the network load is well-prepared for future demands because both the residual bandwidths and the to-be-released ones are considered. Secondly, the algorithm responding time is quick due to no calculation of criticality. Experiments in the next section evaluate performance of BGHT against other popular TE routing algorithms.

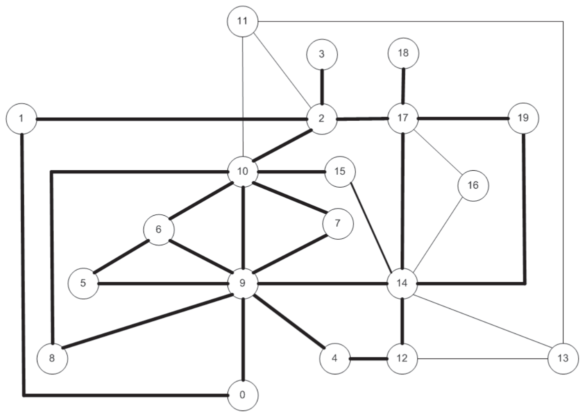
IV. Experimental results

A. Evaluation setup

All the above algorithms including MHA, MIRA, RRATE, BGHT1, and BGHT2 are implemented in Visual C#. Two different network topologies are simulated: one (figure 1(a)) is adapted from previous TE routing works such as [2], [7], [8], and the other (figure 1(b)) is inherited from the real CESNET MPLS topology [9]. Both networks' links are bidirectional and have two types of capacity. The higher capacity in MIRA topology (the thicker links in the figure) is 4800 and the lower ones (the thinner links) is 1200. Meanwhile, those values in CESTNET topology are 10000 and 1000 bandwidth units.



(a) MIRA topology



(b) CESNET topology

Figure 1: Simulation network topologies

For each topology, various test sets of 2000 routing requests per set are evaluated. Those demands arrive randomly according to the Poisson distribution of mean λ requests per time unit, and their holding times are distributed by the Exponential mean μ time units. The source and destination nodes are arbitrarily selected between four ingress egress pairs (0, 12), (4, 8), (3, 1), and (4, 14) of MIRA topology, and the bandwidths are requested in 10, 20, 30, and 40 units. Meanwhile, for CESNET topology, those ingress egress pairs and demanded bandwidths are respectively (0, 18), (1, 11), (3, 16), (4, 7), (5, 13), (6, 19), (15, 0), (19, 8) and 40, 80, 120, 160 units.

B. Accepted ratio and computing time results

Performance of routing algorithms is evaluated by two metrics. Firstly, accepted ratios (percentage of accepted requests) are compared. Obviously, the higher the ratio is, the better an algorithm performs. Secondly, average of computing time is considered. The computation time of each request is counted when it arrives until it is accepted or rejected and should be minimized.

Table IV shows detailed results with following algorithms' parameters.

- For MIRA topology, the Poisson arrival time $\lambda = 40$ and the Exponential holding time $\mu = 10$
- For CESNET topology, $\lambda = 80$ and $\mu = 30$.
- For MIRA algorithm [2], the ingress egress parameters are all equal $\alpha = 1$ (the same value as other TE routing experiments [2], [3], [5])
- For RRATE algorithm [3], the moderation parameters $k_1 = k_2 = 0.5$, the number of pre-selected path $k = 30$, and the racing threshold $N = 15$. Those parameter values are chosen for they give the best results after trying different values.

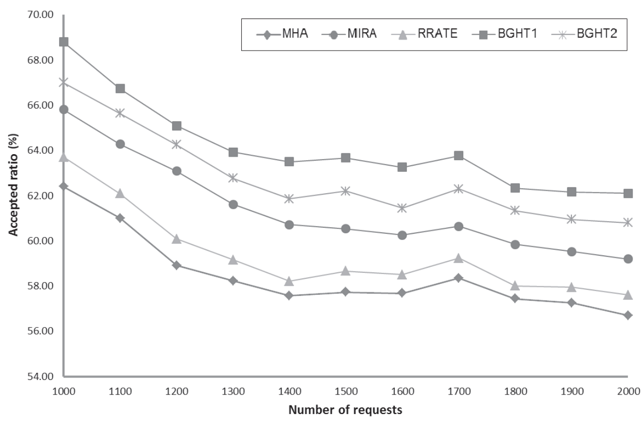
Table IV: Results of accepted ratio (in percent) - computing time (in miliseconds) subject to number of requests (NoR)

(a) Results on MIRA network

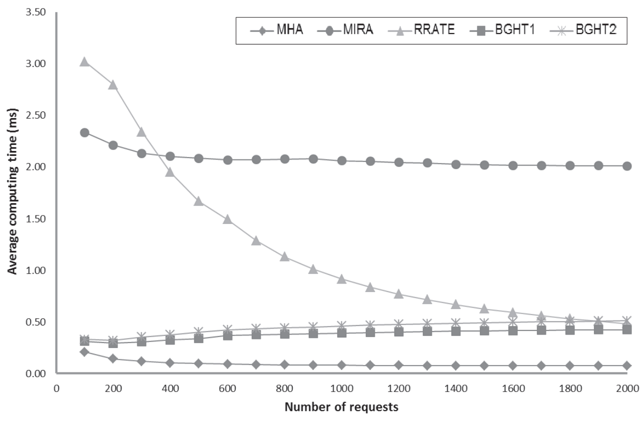
NoR	**MHA**	**MIRA**	**RRATE**	**BGHT1**	**BGHT2**
100	100- 0.21	100- 2.34	100- 3.02	100- 0.31	100- 0.33
500	70.60- 0.10	74.40- 2.08	72.20- 1.67	75.80- 0.34	76.40- 0.40
1000	62.40- 0.08	65.80- 2.06	63.70- 0.91	68.80- 0.39	67.00- 0.46
1500	57.73- 0.08	60.53- 2.02	58.67- 0.63	63.67- 0.41	62.20- 0.49
2000	56.70- 0.08	59.20- 2.01	57.60- 0.48	62.10- 0.42	60.80- 0.51

(b) Results on CESNET network

NoR	**MHA**	**MIRA**	**RRATE**	**BGHT1**	**BGHT2**
100	100- 0.20	100- 8.44	100- 9.90	100- 0.33	100- 0.33
500	83.00- 0.11	74.00- 8.08	73.80- 8.71	73.80- 0.37	73.80- 0.39
1000	63.30- 0.10	63.60- 8.11	63.00- 6.50	66.40- 0.50	65.00- 0.57
1500	52.73- 0.10	53.13- 7.68	53.40- 5.32	54.93- 0.60	53.93- 0.70
2000	47.25- 0.09	47.20- 7.33	47.70- 4.64	48.80- 0.65	47.80- 0.78

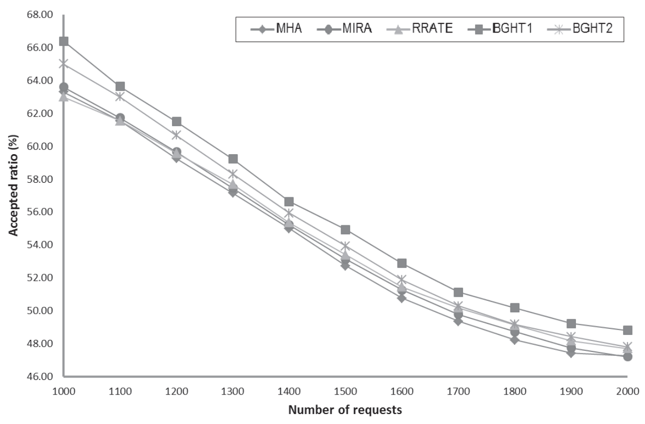


(a) Accepted ratios

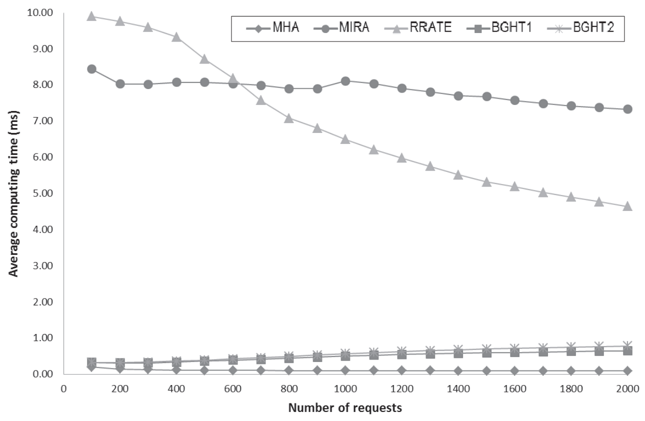


(b) Avarage computing times

Figure 2: Compare of results on the MIRA network



(a) Accepted ratios



(b) Avarage computing times

Figure 3: Compare of results on the CESNET network

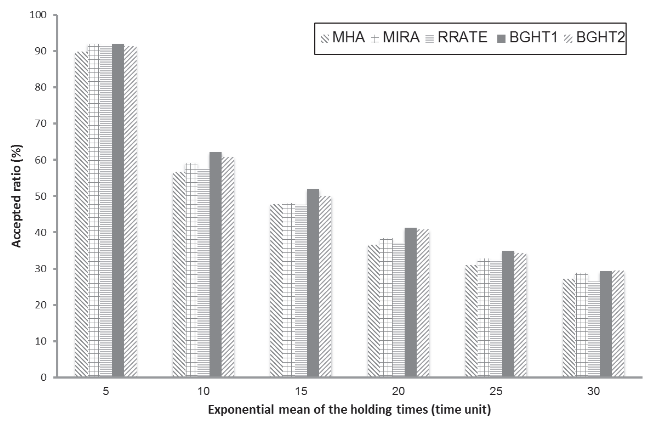
Particularly, figure 2(a) depicts that MHA routes the least number of requests on MIRA topology. The usage of MIRA's interference noticeably improves the MHA's ratio from 56.70% to 59.20% but at the cost of more than 20 times higher computing time. RRATE also has better accepted ratio than MHA and effectively reduces responding time of MIRA. Figure 2(b) clearly shows that the required time for about 300 first requests of RRATE is quite superior but after this learning stage it is drastically decreased. Nevertheless, both of BGHT algorithms gain better performances than the others. Especially, BGHT1 achieves a 2.9% improvement on the accepted ratio over MIRA, 4.5% over RRATE, and even 5.4% over MHA. Additionally, the average computing time of BGHT1 is not only much better than that of MIRA but also lower and more stable than RRATE's (figure 2(b)).

Effectiveness of the BGHT algorithms is also illustrated by experiments on CESNET topology in which the BGHT2 closely accepts more requests than MHA, MIRA, and RRATE while the BGHT1 has a noticeably better accepted ratio than BGHT2's (figure 3(a)). Because CESNET is inherited from a real network [9], its topology is optimized for the shortest path routing. Therefore, MHA performs well on the CESNET; and the improvements of BGHT algorithms (1.55% and 0.55% higher of BGHT1 and BGHT2 respectively) are meaningful although the absolute differences are small. On the other hand, figure 3(b) also presents that the responding times of BGHTs are significantly lower than MIRA and RRATE.

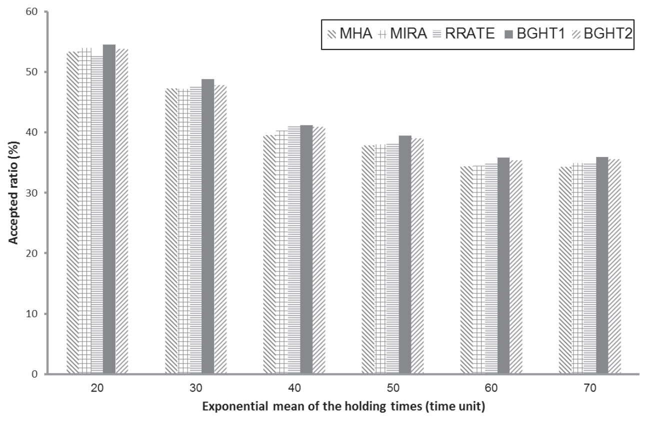
Furthermore, it is worthy to mention that the above details are example results of many conducted tests. Various experiments are repeated with different generated requests and / or algorithms' parameters. All outcomes indicate that the BGHT algorithms, especially the BGHT1, outperform the others.

C. *Effect of the holding times*

This sub section investigates how the holding times affect performance of the routing algorithms, Intuitively, the longer the routes hold bandwidth resource, the less requests are acceptable. Figure 4 plots the accepted ratios of all the algorithms on various values of holding times. Specifically,



(a) Results on the MIRA topology



(b) Results on the CESNET topology

Figure 4: Plot of the accepted ratios versus the Exponential means of requests' holding times (μ values)

test sets of 2000 demands are randomly generated. Their arrival rates are distributed by the fixed Poisson $\lambda = 40$ demands per time unit for the MIRA network and $\lambda = 80$ for the CESNET one, while the holding times are allocated by different Exponential μ time units.

The experimental outcomes indicate that the proposed BGHT algorithms allow more requests than the others in all cases of holding time μ values. For example, when $\mu = 15$ the BGHT1 has the highest percentage of 51.95% and the BGHT2 has the 2nd highest value of 50.05% on MIRA topology (figure 4(a)). Another example in figure 4(b) shows the case of $\mu = 60$ on CESNET, BGHT1 and BGHT2 respectively gain 35.85% and 35.4% comparing to near 35% accepted ratios of the other algorithms.

Additionally, BGHT1 performs well for either short or long holding times. It is because BGHT1 effectively select the widest links for the next requests. In the other words, network resource is dynamically prepared for every next routing demands. Meanwhile, BGHT2 always consider future bandwidths into link weights. However, if the minimum holding time is long then BGHT2 might misselect the heavy load links. In contrast, if more than one paths are going to be released in the short future period but BGHT2 only count the soonest one, it also leads to negative effect. Consequently, BGHT2 is not as efficient as BGHT1 although it has better performance than the existing algorithms.

V. Conclusion

This paper introduces a novel idea to use holding times for bandwidth guaranteed TE routing algorithms. The proposed BGHT algorithms simply but effectively weigh links as proportional to available bandwidths after the Δt_i period (counting from the current request time t_i). The BGHT1 greedily determines the Δt_i as an interval between the current and the next demands while the BGHT2 identifies the Δt_i as the minimum of the resting holding times. Thorough experiments show that the BGHT algorithms perform considerably better than the other popular ones with respect to both accepted ratios and average computing times. Furthermore, the greedy selection of BGHT1 results in its impressive performance.

References

[1] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[2] K. Kar, M. Kodialam, and T. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2566–2579, 2000.

[3] B. J. Oommen, S. Misra, and O. Granmo, "Routing Bandwidth-Guaranteed paths in MPLS traffic engineering: A multiple race track learning approach," *IEEE Transactions on Computers*, vol. 56, no. 7, pp. 959–976, 2007.

[4] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Proceeding SFCS '75 Proceedings of the 16th Annual Symposium on Foundations of Computer Science*. IEEE, 1975, pp. 184–193.

[5] C. T. P. Thanh and T. C. Hung, "A study of bandwidth guaranteed routing algorithms for traffic engineering," in *Advanced Methods for Computational Collective Intelligence*, ser. Studies in Computational Intelligence, N. T. Nguyen, B. Trawiński, R. Katarzyniak, and G.-S. Jo, Eds. Springer Berlin Heidelberg, 2013, vol. 457, pp. 313–322.

[6] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Triangular Distribution*. John Wiley & Sons, Inc., 2010, pp. 189–190.

[7] R. Boutaba, W. Szeto, and Y. Iraqi, "DORA: efficient routing for MPLS traffic engineering," *Journal of Network and Systems Management*, vol. 10, pp. 309–325, 2002, 10.1023/A:1019810526535.

[8] A. Alidadi, M. Mahdavi, and M. R. Hashmi, “A new low-complexity QoS routing algorithm for MPLS traffic engineering,” in *2009 IEEE 9th Malaysia International Conference on Communications (MICC)*, Kuala Lumpur, Malaysia, 2009, pp. 205–210.

[9] Václav Novák, Petr Adamec, Pavel Šmrha, and Josef Verich, “CESNET2 IP/MPLS backbone network design and deployment in 2008,” 2008. [Online]. Available: http://www.cesnet.cz/doc/techzpravy/2008/CESNET2-ip-mpls-backbone-in-2008/