Simulated Annealing Based Bandwidth Reservation for QoS Routing

Baohua Zhang[a], Changcheng Huang[b], Michael Devetsikiotis[c]

a School of Mathematics and Statistics, Carleton University, Ottawa, Canada
bhzhang@sce.carleton.ca
b Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
huang@sce.carleton.ca
c Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA
mdevets@ncsu.edu

Abstract— **Numerous routing schemes have been reported to improve network performance over the years. Multi-path routing belongs to one of them and MPLS is an excellent platform for such routing. In this paper, the Shortest Distance Path Based Simulated Annealing (SDPSA) algorithm for finding optimal bandwidth reservation solutions for multi-path routing is developed to improve network performances. The algorithm, which employs the annealing method, is based on previous solutions to find the current sub-optimal solution for multi-path routing. Multiple objectives including balancing traffic load and minimizing network resource consumption are taken into consideration. Finally, the proposed algorithm is applied to a randomly generated network and the NSFNET network. The performance values are compared to a well-known multi-path routing algorithm-HSTwp. The simulation and comparison results show that the proposed SDPSA algorithm is feasible and efficient for the optimization of multi-path IP routing.**

I. INTRODUCTION

Congestion control is one of the major problems of network optimization. Many congestion control algorithms reduce the traffic rate at the edge of the network based on feedbacks of the online traffic measurements. There are others trying to solve the congestion by shifting the traffic to alternative paths [1][2]. Due to traffic shaping at the edge, traffic demands in the core network are usually stable over a given time period [3]. Given the topology of a network and capacity of the links, some online measurement approaches such as effective bandwidth approximation, can be used to estimate the available bandwidth over different links [4][5].

To avoid uneven network utilization, numerous approaches with various objectives in network optimization have been proposed for efficiently utilizing the network resources such as [6][7][8][9][10]; Fortz's scheme adjusted link weights to optimize the network utilization [6]; Wang's scheme minimized the maximum utilization over links to improve the ability of accepting more future traffic [7]; Kodialam's scheme tried to establish a path with less interference by increasing the cost of critical links [8]; In reference [9], constrained multi-path routing scheme was formulated, and a heuristic algorithm called HSTwp was given by Lee et al. to calculate the constrained multiple paths and their load splitting ratios. By splitting the traffic demands among multiple paths, it is possible to improve the overall network utilization through better load balancing schemes. However, splitting the traffic demands among multiple paths also raises new problems, for example, how to determine the number of paths and how to assign the volume of traffic to the selected paths in accommodating various traffic demands; how to adaptively accommodate the dynamic traffic demands by using multi-path routing algorithms, etc. Such issues require the candidate multi-path routing algorithms to be simple and scalable.

In this paper, we introduce an algorithm called SDPSA to find sub-optimal multi-path routing solutions for dynamic traffic demand. SDPSA algorithm proposed here is a simple adaptive routing algorithm which allows efficient load balancing and network resource consumption. Our objective is to find a set of paths that can accommodate traffic demands with as little consumption of network resources as possible. At the same time, network load balancing is considered to reduce network congestions. Through simulated annealing, the summation of distances of selected path set (SPS) is minimized. Starting from the previous solution as initial annealing point, the SDPSA algorithm is more efficient in path set re-selection as traffic demands change dynamically.

The rest of the paper is organized as follows: section II formulates the problem leading to the development of the SDPSA algorithm and describes the objectives in detail. Some definitions and notations are also given. In section III, after a brief description of the background of simulated annealing, the SDPSA algorithm is developed. In section IV, the proposed algorithm is applied to two network examples and the simulation results are analyzed and compared with the well-known multi-path routing algorithm, HSTwp. The last section summarizes our work and identifies some directions for future work.

II. PROBLEM FORMULATION

A backbone network can be modeled as a bi-directional graph G (V, E). Nodes or vertices $\{V\}$ represent switches, routers, or hosts; edges $\{E\}$ represent communication edges, $N=|V|$ and $M=|E|$ denote the number of vertices and edges respectively. For each edge in E, there will be two links for both traffic directions, they are expressed as (i,j) and (j,i). Throughout this paper, (i, j) will be used to denote a link from node i to node j and r_{xy} is used to denote a path from ingress node x to egress node y unless specified otherwise.

A. Some Definitions

Traffic demand matrix: For each link (i, j) in E, where i and j are network nodes, let C_{ij} be the capacity of the link. Let $T_d(\tau)$ represent the set of all traffic demands among all pairs of ingress and egress routers over a MPLS network at time τ. The traffic demand matrix can be written as the matrix $T_d(\tau)$ in (1); $t_{i,j}(\tau)$ stands for traffic demands from node i to node j and $t_{i,j}(\tau)$ is piecewise constant in each time period τ. $(0 \le i, j \le N-1)$. τ stands for each time period. For example, time period τ_1 stands for time T_0 to T_1 and τ_2 stands for time T_1 to T_2, and so on.

We assume here that there is a good enough online measurement algorithm that can predict the future traffic relatively accurately. Sometimes the traffic predictions are not so accurate especially when predicting the future traffic in relatively small time period is needed. In our example, we use the aggregation of effective bandwidths from the edge of the network. This way, we can obtain a relatively accurate traffic matrix $T_d(\tau)$ indicating the future traffic demands in the next time period after τ. The time period can be measured in hours or days. Within each of these time periods, the traffic demand matrix is fixed. This assumption is reasonable according to the previously reported work [10][11]. As mentioned in [11], the traffic profile can be taken from the service level agreements, created by rule-of-thumb or any other mechanism suitable to the network operator.

$$T_d(\tau) = \begin{bmatrix} 0 & t_{0,1}(\tau) & \cdots & t_{0,N-2}(\tau) & t_{0,N-1}(\tau) \\ t_{1,0}(\tau) & 0 & \cdots & t_{1,N-2}(\tau) & t_{1,N-1}(\tau) \\ \cdots & \cdots & 0 & \cdots & \cdots \\ t_{N-2,0}(\tau) & t_{N-2,1}(\tau) & \cdots & 0 & t_{N-2,N-1}(\tau) \\ t_{N-1,0}(\tau) & t_{N-1,1}(\tau) & \cdots & t_{N-1,N-2}(\tau) & 0 \end{bmatrix} \quad (1)$$

Candidate Path Set (CPS): All paths between an ingress and egress pair are candidate paths for this ingress-egress pair. A Candidate Path Set (CPS) is the set of all paths for all ingress-egress pairs in a given topology.

Selected Path Set (SPS): a SPS is a subset of *CPS,* which can satisfy the traffic demand matrix $T_d(\tau)$. Therefore it is a function of time period τ. In the following, we will omit τ for simplicity.

Norm of a *SPS*: the number of paths in a s*elected path set* S is its norm, denoted by $|S|$. With the given traffic demands T_d, there is a total of $N*(N-1)$ pairs of nodes. For each pair of nodes (x, y) which has nonzero traffic demands, there is a certain number of paths, denoted by $|S|_{x,y}$, in SPS to accommodate the traffic demands related to node pair (x, y).

Neighbor of a SPS: the set of all SPS with the same number of paths for each ingress-egress pair as SPS S is called the neighbor of S . Thus, $N(S)$ has the same norm $|S|$.

Complete Solution Set, C^{sps}: all possible solutions related to all SPS in accommodating given traffic demands are called the complete solution set C^{sps}. C^{sps} is the whole solution space for the incoming traffic demands $T_d(\tau)$.

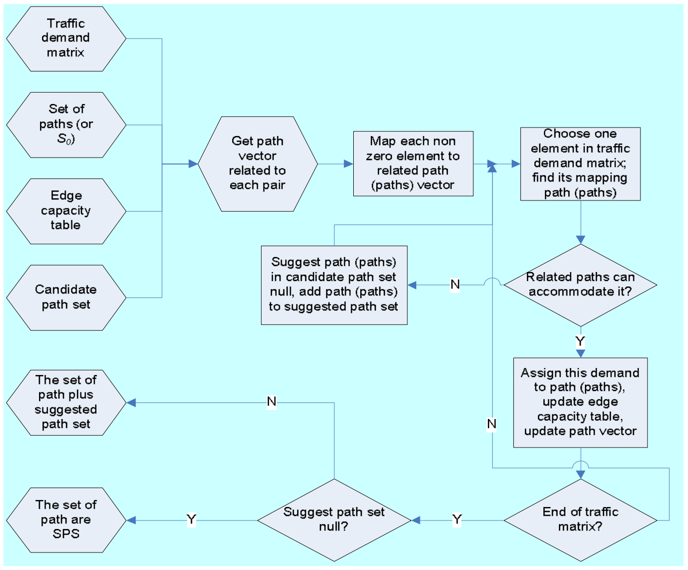
Link Capacity Table: This table contains the identity information of each link in the network. It also contains the nodes incident to each link, and the original capacity of each link. Moreover, the dynamically updated residual capacity for each link is also included.

Path vector, $P(r^i_{xy})$: *path vector* contains several pieces of information: the identification of links in the path, the identification of the path (i) among node pair x and y, the capacity of the path i.e. the minimum of the capacities of all links in the path, and the amount of traffic assigned to this path. For example, $P(r^i_{xy})=(l_0,l_1,l_2,\ldots,l_{M-1}, C^i_{xy}, S_{r^i_{xy}})$ stands for a path vector related to path r from node x to node y. For each ingress-egress pair x and y, there might be many paths. We will use the notation i to identify the different paths related to ingress-egress pair x and y (i=1,2,3,…). C^i_{xy} is defined as the capacity of the path which is the minimal capacity of all links in the path, and $S_{r^i_{xy}}$ is the traffic assigned to path r^i_{xy}. We will use bandwidth as units for both C^i_{xy} and $S_{r^i_{xy}}$ in the following sections.[1] All nodes are sorted and each node is identified with an i.d. number from 0 to N-1. Supposing a link exists from node i to node j ($0< i < j < N-1$), l_i is defined as (2).

$$l_i = \begin{cases} 1, & (i,j) \in r_{xy} \\ 0, & (i,j) \notin r_{xy} \\ -1, & (j,i) \in r_{xy} \end{cases} \quad (2)$$

SPS generator function: The *SPS generator function* is used for generating a SPS. The input includes traffic demand matrix, traffic incident matrix, path vector, link capacity table, and a subset of a related CPS to accommodate the traffic demand matrix. The SPS generator function verifies if the subset is a *selected path set.* The output includes the *SPS* and its *norm* when the given subset is *SPS;* otherwise it provides an additional suggested path set Fig. 1.

[1] Generally, we will use r_{xy} to denote a path between ingress-egress nodes x and y when we do not need to identify all the paths among these two nodes.



Traffic demand matrix
Set of paths (or S_0)
Edge capacity table
Candidate path set
Get path vector related to each pair
Map each non zero element to related path (paths) vector
Choose one element in traffic demand matrix; find its mapping path (paths)
Related paths can accommodate it?
N
Suggest path (paths) in candidate path set null, add path (paths) to suggested path set
Y
Assign this demand to path (paths), update edge capacity table, update path vector
N
End of traffic matrix?
Y
Suggest path set null?
N
The set of path plus suggested path set
Y
The set of path are SPS

Fig. 1: SPS generator function

B. Formulation

Our objective is to achieve minimum overall resource consumption and balance the load simultaneously. We define the distance of a path to be the sum of the inverses of the available bandwidths on all links along the path. This definition tries to strike a balance between link hop counts and available bandwidth.

Supposing U_{ij} is the unused bandwidth of link (i, j) where (i, j) is a link in path r_{xy}. If we use S_{ij} to denote the total traffic amount assigned to link (i, j), then $U_{ij} = C_{ij} - S_{ij}$

$$S_{ij} = \sum_{(i,j)\in r_{xy}} S_{r_{xy}} \quad where\ x \neq y;\ x, y \in \{V\} \qquad (3)$$

Formula (3) shows the traffic assigned to link (i, j) equals the sum of the entire traffic assigned to each path in *SPS* traverses link (i, j). V stands for the node set. We define the distance of a path r_{xy} as in (4)

$$D(r_{xy}) = \sum_{(i,j)\in r_{xy}} 1/U_{ij} \qquad (4)$$

Given a topology $G\ (V, E)$ with N nodes and related capacities for links, there is a total of $N*(N-1)$ pairs of nodes. With given traffic demand, the objective function is formulated as follows:

$$\underset{S\in C^{sps}}{Min} \sum_{r_{xy}\in S} D(r_{xy}) \quad where\ x \neq y;\ x, y \in \{V\} \qquad (5)$$

Subject To:

1. $S_{ij} \leq C_{ij} \quad where\ (i, j) \in r_{xy}$

2. $\sum_{r_{xy}\in S} S_{r_{xy}} \geq t_{xy} \quad where\ x \neq y;\ x, y \in \{V\}$

Where r_{xy} stands for the paths between nodes x and y; $t_{x,y}(\tau)$ is the traffic demands between nodes x and y. ($t_{x,y}(\tau)$ is an element in *traffic demand matrix* T_d (τ)) S_{ij} and $S_{r_{xy}}$ are the traffic assigned to link (i, j) and path r_{xy} respectively. While the total traffic assigned to link (i, j), S_{ij}, is equal to the sum of the traffic assigned to all paths r_{xy} that traverse link (i, j), as shown in (3).

The reason why we chose to minimize the sum of distances of all selected paths between all ingress-egress pairs is the following: the distance of each path is a combination of considering both load balance and network resource consumption. The sum of the distances of all selected paths between all ingress-egress node pairs is a combined consideration of two objectives: load balancing and resource consumption of the whole network.

To summarize, the paths in the optimized Selected Path Set should have as low hop counts as possible, at the same time, we should aim to balance the load over all paths in order to keep the available bandwidth of each link as much as possible

The calculation of (5) is time consuming because the available bandwidth along all links changes dynamically. Furthermore, the change of a path selection in one ingress and egress router pair might lead to changes among a few other pairs and finally make the calculation even more complex. To solve the problem within a reasonable time frame, the next section will deal with the simulated annealing technique used for developing an algorithm.

III. SDPSA ALGORITHM

A. Simulated Annealing

The SDPSA algorithm discussed in this paper is based on the so-called annealing process. The term *annealing* is an analogy of an optimization technique to the cooling process of a liquid or solid. In an annealing process a metal is allowed to be cooled more slowly than usual and its interior structure will be more consistently; therefore, it will have much stronger crystal structures than their counterparts with faster cooling. Since Kirkpatrick *et al* suggested the simulated annealing technique in 1983 [12], it has been widely applied to optimization in areas such as the Very Large Scale Integration (VLSI) design, and the Traveling Salesman Problem (TSP) which is an NP hard problem.

The simulated annealing scheme for finding the optimal solution can be applied to solving the multi-path routing optimization problem in section II. In order to find a Selected Path Set satisfying our objective function in (5), we need to take a further look at the distance formula (4). In formula (4), each element on the right-hand side is an indication of the utilization of each link. The summation part indicates the total hops in each path. In summary, the paths in the optimized selected path set should have as little hop count as possible, and at the same time, balance the load over all paths to keep as much available bandwidth as possible on each link.

In order to accommodate the traffic demands, the paths of some node pairs (or all node pairs) in the network (depending on the traffic demand matrix) will be selected and the traffic will be assigned to each path according to its link capacity. For each particular node pair in the network, when the number of paths increases in the Selected Path Set, the link utilization in each path will decrease. On the other hand, the link utilization will increase if fewer paths are selected for the traffic demand related to this particular node pair.

B. SDPSA Algorithm

As we mentioned previously, the optimization problem discussed in this paper is very similar to an annealing process, and thus simulated annealing optimization is applied. As discussed above, the traffic metrics do not change in a certain time period regardless of the duration of this period. In a given time period τ_1, we assume that the sub-optimized solution is available with a given path set which can accommodate related traffic demands with minimized objective function. As the time moves to the next period, τ_2, the traffic demand metrics change, and consequently, some or all of the paths related to individual ingress and egress pairs should be changed in order to reach new optimal solution. As shown in Fig. 2, our newly developed SDPSA algorithm does take advantage of the previous optimal solution to achieve the current objective.

```
Select any candidate subset related to traffic demand matrix and
get an initial solution $S_0$ from S Generator
Repeat
    Repeat
      Randomly select S in $N(S_0)$
      let $\delta=F(S)-F(S_0)$
      if $\delta < 0$ then $S_0=S$
      else generate random x uniformly in range (0,1);
      if $x< \exp(-\delta*|S|)$ then $S_0=S$;
    Until iteration-count = next representative
      Add edges to $S_0$ based on edge capacity table
Until all edges reach the stopping condition
$S_0$ is approximately to the optimal solution.
```

Fig. 2: SDPSA algorithm

There are two loops in the SDPSA algorithm, and the inner loop searches for a better solution among those SPSs with the same norm. The *SPS* with the same norm is the analogy of the temperature in simulated annealing. The SPS (noted as *S*) with the same norm here stands for the number of paths that have been selected according to each ingress and egress pair ($|S|_{x,y}$) are equal.

$$|S| = \sum_{r_{x,y}\in S} |S|_{x,y} \quad \text{Where } x \neq y;\ x, y \in \{V\} \qquad (6)$$

Parameter δ is defined as the difference between the currently selected sample of *distance of S* (7) and its neighbor's *distance of* S_0 (8).

$$F(S) = \sum_{\substack{r_{xy}\in S \\ S\in N(S_0)}} D(r_{xy}) \qquad (7)$$

$$F(S_0) = \sum_{r_{xy}\in S_0} D(r_{xy}) \qquad (8)$$

Here *S* is also a selected path set. Whenever a better solution is found, i.e. $\delta < 0$, the currently best solution will be updated; otherwise, a random number *x* between (0, 1) is generated. If $x < \exp(-\delta*|N-S_0|)$, the currently best solution will still be updated. The iteration continues until the maximum number of repetitions is reached.

The outer loop increases the number of paths similar to the decrease of temperature in the simulated annealing process until the stopping condition is met. Indeed, the solution of the last time period can be used as the initial solution of the current time period. As shown in Fig. 2, if previous solution, say S_0, can accommodate current traffic demands, the algorithm will simply take S_0 as the current solution.

As can be seen from the algorithm in Fig. 2, the SDPSA algorithm allows probabilistic acceptance of non-improving moves. The acceptance probability depends on the control parameter, $|S|$, and the amount of costs increases, δ. The acceptance probability is high for a small δ and $|S|$. As $|S|$ increases, the acceptance probability will decrease accordingly.

C. Complexity Analysis

The HSTwp algorithm contains three parts: preprocessing the given graph, finding multiple paths, and calculating load-splitting ratios. The complexity analysis of the HSTwp algorithm is as following: firstly, in the graph conversion problem, the computation complexity is bounded by $O(|M+N|)$ where *M* is the number of links and *N* is the number of nodes in the graph. Secondly, in the *K* widest paths problem, the best known bound for the ordered set is $O(N^3)$. Finally, the algorithm for splitting the traffic demand into *K* paths is bound by $O(|K\log K|)$, and calculating the load splitting ratio for each path is bound by $O(|2K|)$ where *K* is the number of paths selected. Therefore, the time complexity for the HSTwp algorithm is bound by $O(M+N^3+|K\log K|)$ [9].

The SDPSA algorithm also consists of three parts. Firstly, in preparing the *candidate path set* for each node pair, the computation complexity is $O(N!)$. However, finding *K* paths between each pair of nodes in the network to be the *candidate path set* is the alternative solution for finding the *candidate path set*. The complexity of finding *K* paths for a pair of nodes is bounded by $O(N^3)$ [13]. There is a total of $N*(N-1)$ pairs where *N* is the number of nodes. Therefore the total computation complexity for this computing *K* paths for each pair of nodes in the network as our *candidate path set* is $O(N^5)$. This part needs to be calculated offline for one time only unless the topology changes. Second, in the initial *selected path set*, the worst case for calculating a selected path set, with *S* generator function, is $O(N^2*KM\log N)$ where *M* is the number of links, *N* is the number of nodes, and *K* is the number of paths for any pair of nodes in a given topology. Finally, for the simulated annealing process, we simply use previous *SPS* if the previous solution can accommodate the current future traffic. Otherwise, *SPS* will be re-calculated and the time complexity is bounded by $O(N^2*KMlogN)$. The total number of paths in *candidate path set* is $K*N*(N-1)$. If we increase the norm by one each time, the time complexity for the simulated annealing process is bound by $O(N^4*K^2MlogN)$. In a more coarse way, we can increase the norm of *SPS* by at most N^2 links each time until number of paths related to each pair reaches the limit *K*. The complexity for the simulated annealing process is then reduced to $O(N^2*K^2MlogN)$ [14].

IV. SIMULATION RESULTS

In this section, the performance of the SDPSA algorithm is compared with the HSTwp algorithm in a random generated network topology (case I), and in the NSF network (case II).

A. Simulation Analysis for Case I

The performance of the SDPSA algorithm is investigated by applying the algorithm to a randomly generated topology example. The topology shown in Fig. 3 is generated by the BRITE topology generator [15], which can generate flat topologies. Our topology example has six nodes and twenty links. Link capacities are obtained from the BRITE package by using a uniform distribution over the range of {300 1000} bandwidth units. The traffic is bi-directional and the traffic demands are asymmetric as that of the common cases in [9]. The traffic demands are taken from randomization of an *initial static traffic demands matrix* of $\{t_{05}=600, t_{14}=500, t_{21}=400, t_{54}=300\}$.

By applying the well-known heuristic HSTwp algorithm reported in [9] and our SDPSA algorithm to the scenario above, two groups of different multi-path routing solutions are generated through various time periods. In the following, we will analyze the simulation results to show that the SDPSA algorithm has a superior performance over the HSTwp algorithm in terms of overall network resource consumption.

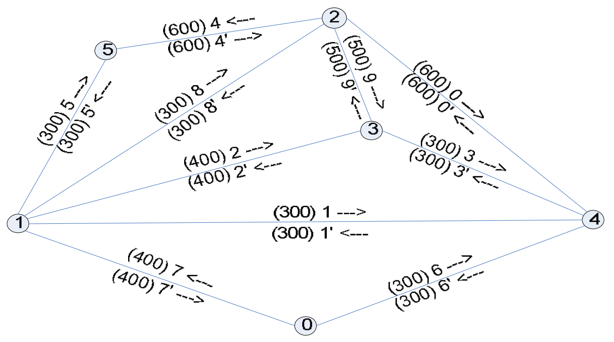


Fig. 3: Simulation topology

Table 1 shows that the SDPSA algorithm consistently saves overall network resources. The total network resource consumption is calculated by adding the product of all the bandwidth assigned to each of the selected paths and hop counts in each path. On average, by using SDPSA, 12.9% resource was saved compared to HSTwp algorithm. The maximum resource savings can reach up to 26.6%, which is more than twice of the average resource saved. This implies that the algorithm meets our objective of reducing network resource consumption. By using our algorithm, the network can satisfy dynamic traffic demands more efficiently.

Table 1: TOTAL RESOURCES CONSUMED FOR CASE I

Time period	Resources used in SDPSA (R_{SDPSA})	Resources used in HSTwp (R_{HSTwp})	Resources gain ((R_{HSTwp}-R_{SDPSA})/R_{SDPSA})
1	3841	4532	0.180
2	3500	3911	0.117
3	3206	3542	0.105
4	3560	3911	0.099
5	3570	3870	0.084
6	3989	4215	0.057
7	2440	3061	0.255
8	3355	4247	0.266
9	4394	5081	0.156
10	3948	4077	0.033
Average	3580.3	4044.7	0.129

B. Simulation Analysis for Case II

Different from scenario I which is a randomly generated simpler network topology, the NSFNET (Fig. 4) is a well known practical core network example. The performance of our SDPSA algorithm is analyzed using this real backbone network topology. Again, the HSTwp algorithm is performed on the same topology for comparison. Similar to scenario I, the link capacities are obtained by using a uniform distribution over the range of {300 1000} Megabytes bandwidth units. The traffic demands are generated from randomization of an *initial static traffic demand matrix* of $\{t_{1,8}=400, t_{2,6}=700, t_{11,13}=500\}$ and we use megabyte bandwidth units as the traffic demands.

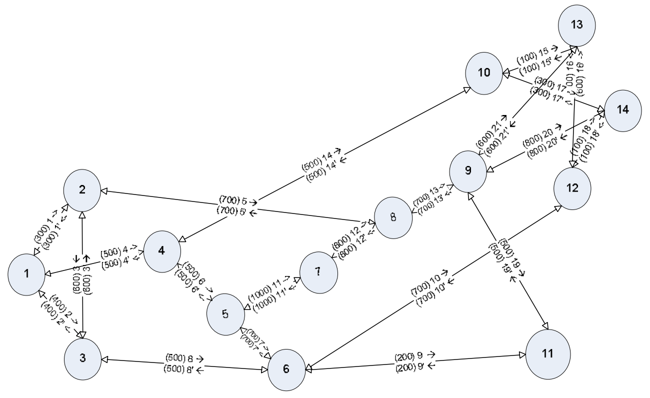


Fig. 4: NSFNET Topology with bi-directional links

Table 2 shows the SDPSA algorithm consistently saves overall network resources over the HSTwp algorithm on average, by using the SDPSA algorithm, 7.3% less network resources were consumed comparing to the HSTwp algorithm. The maximum resource saving can reach up to 10%. The overall resource gain for the SDPSA algorithm is less than the gain in scenario I. One of the factors that affect the resource gain is that there exists less disjoint paths for node pairs in NSFNET topology. Nevertheless, our SDPA algorithm does successfully achieve the objective of reducing the total network resource consumption. As a result, by using the SDPSA algorithm, the network can accommodate more traffic demands.

Table 2: TOTAL RESOURCES CONSUMED FOR CASE II

Time period	Resources used in SDPSA (R_{SDPSA})	Resources used in HSTwp (R_{HSTwp})	Resources gain ((R_{HSTwp}- R_{SDPSA})/R_{SDPSA})
1	4153	4540	0.0932
2	4185	4546	0.0863
3	3608	3916	0.0854
4	3784	4111	0.0860
5	4632	4866	0.0510
6	5158	5371	0.0420
7	3386	3662	0.0820
8	3576	3938	0.1000
9	3148	3454	0.0974
10	5500	5550	0.0091
Average	4113	4395.4	0.0732

V. CONCLUSIONS

In this paper we present a Shortest Distance Path based Simulated Annealing (SDPSA) algorithm for the multi-path routing and the splittable multi-commodity flow problem for dynamic traffic engineering in backbone networks. The algorithm finds a set of paths with the objective of minimizing the overall network resource consumption and achieving network load balancing at the same time. Moreover, the algorithm takes advantage of the previous sub-optimal solution by using it as the initial point for subsequent annealing process. As a result, the SDPSA algorithm is more efficient in getting new sub-optimal solutions. The SDPSA algorithm is applied to a randomly generated network topology and to the NSFNET topology. After comparing the proposed algorithm with the well-known HSTwp algorithm, the simulation results and comparison analysis show that the SDPSA algorithm has a superior performance over the HSTwp algorithm in terms of network resource consumption.

By using the multi-path routing with bandwidth reservation over backbone networks, the SDPSA algorithm gives a sub-optimal solution regarding the dynamic traffic demands. The simulated annealing method used in the algorithm takes advantage of previous multi-path routing information as its initial point for the current period. This is an efficient way to obtain multi-path routing solutions with dynamic traffic demands.

It is necessary for future research to uncover the applicability of the SDPSA algorithm to different applications such as VoIP and VPN. Improving the algorithm, for example, in regards to the alternative stopping conditions in the algorithm could be possible direction for future research.

ACKNOWLEDGMENT

The authors would like to express their appreciations to all people in Optical Networks Laboratory in Carleton University especially Ph.D. candidate HongQing Zeng for his valuable comments and stimulating discussions.

REFERENCES

[1] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering", INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE,Vol.3, 2001,pp.1300-1309.

[2] S.D. Patek, R. Venkateswaran, J. Liebeherr, "Enhancing Aggregate QoS Through Alternate Routing", Global Telecommunications Conference, 2000. GLOBECOM '00.IEEE Volume 1, 27 Nov.-1 Dec. 2000 Page(s):611 - 615 vol.1

[3] K. Papagiannaki, N. Taft, Z. Zhang & C. Diot, "Long-Term Forecasting of Internet Backbone Traffic-Observations and Initial Models", Conf. Computer Communications, IEEE INFOCOM, 2003

[4] F. Kelly, "Notes on Effective Bandwidths," Stochastic Networks: Theory and Applications, ClarendonPress, Oxford, 1996.

[5] A.W. Berger, Y Kogan, "Dimensioning Bandwidth for Elastic Traffic in High Speed Data Networks," Networking, IEEE/ACM Transactions on, Oct. 2000.

[6] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," IEEE Journal on Selected Areas in Communications (JSAC), 20(4):756—766, May 2002.

[7] Y. Wang, and Z. Wang, "Explicit Routing Algorithms for Internet Traffic Engineering", in Proc. IEEE ICCCN, 1999

[8] M. Kodialam and T.V. Lakshman, "Minimum Interference Routing with Applications to MPLS Traffic Engineering", Proceedings of IEEE INFORCOM 2000

[9] Y. Lee, Y. Seok, and Y. Choi, "Traffic Engineering with Constrained Multi-path Routing in MPLS Networks", IEICE TRANS. COMMUN., 2002

[10] Y. Yang and C.-H. Lung, "Traffic Forecast in QoS Routing", Proc. of the 22nd Queens Biennial Symposium on Communications (QBSC), Queens University, Kingston, ON, May 2004, pp. 277-279.

[11] S. Suri, M. Waldvogel, D. Bauer & P. R. Warkhede, "Profile Based Routing and Traffic Engineering", Computer Communications, vol. 26, 2003.

[12] J.C. Spall, " Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control", Wiley, Hoboken, NJ, 2003

[13] J.Y. Yen, "Finding the K Shortest Loop-less Paths in a Network", Management Science 17:712-716, 1971

[14] B. Zhang, "SDP Based Simulated Annealing on Bandwidth Reservation with Multi-Path Routing", M.Sc. thesis, School of Mathematics and Statistics, Ottawa-Carleton Institute of Mathematics and Statistics, Carleton University, 2005

[15] A. Medina, A.Lakhina, I. Matta, and J. Byers. "Brite: An Approach to Universal Topology Generation," Proc. MASCOTS Aug, 2001