Construct QoS End-to-end Virtual Path With Lagrangean Relaxation Method

Chang-Jiu Chen, Ming-Su Liu
National Chiao Tung University, Taiwan,ROC
{cjchen,msliu}@csie.nctu.edu.tw

Abstract- *In this paper, we have improved a QoS routing problem. We give an approach to minimize the congested link utilization while to satisfy individual connection's packet delay. We use a Lagrangean Relaxation based approach augmented with an efficient primal heuristic algorithm, called Lagrangean Relaxation Heuristic (LRH). With the aid of generated Lagrangean multipliers and lower bound indexes, the primal heuristic algorithm of LRH achieves a near-optimal upper-bound solution. A performance study delineated that the performance trade-off between accuracy and convergence speed can be manipulated via adjusting the Unimproved Count (UC) parameter in the algorithm. We have drawn comparisons of accuracy and computation time between LRH and the Linear Programming Relaxation (LPR)-based method, under three networks named* ***NSFNET****,* ***PACBELL****, and* ***GTE*** *and three random networks. Experimental results demonstrated that the LRH is superior to the other approach, namely the LPR method, in both accuracy and computational time complexity, particularly for larger size networks*

Keywords: QoS, Routing Problem, Lagrangean Relaxation, LRH, LPR

1. Introduction

To ensure reliable and high-quality network services, routing and capacity assignment policies should be carefully designed. Traditional quasi-static routing algorithms attempt to optimize a certain aggregate measure, e.g. to minimize the average end-to-end packet delay [1, 2]. However, this kind of performance measures may not be consistent with the service objectives and may result in fairness problems.

The routing problem in virtual circuit networks has been a traditional research topic in computer networks and has attracted even more attention since the emergence of the Asynchronous Transfer Mode (ATM) technology. However, most previous researches on virtual circuit routing considers the objective function of minimizing the average end-to-end packet delay [1, 3, 4], which address a system-optimization perspective without taking individual users into account. And also these researches do not consider the later connections, that is these current established connections may cause big load for connections that request to establish virtual circuits later. Cheng and Lin[5] took a user-optimization approach and considered a fairness issue by minimizing the maximum individual end-to-end packet delay in virtual network, but they didn't consider the system's perspective. In this paper, we attempt to jointly consider both system and user perspectives, and keep maximum tolerance to the later connections.

2. Problem Model and Formulation

We will describe our problem, and model it in this section. Furthermore, we also formulate our problem into *non-linear integer programming* form.

Problem Description

We construct the network into load balance model subject to end-to-end packet delay constraints for each individual user. This model has two advantages.

1. This model can reduce packets delay implicitly.
2. This model reserves the maximum flexibility to the later connections.

The problem has also been shown to be NP-complete which means no polynomial time algorithm for it unless P=NP. For the sake of obtaining sub-optimal solutions, Lagrangean relaxation is applied to the formulation to decompose the problem into several tractable subproblems in next section. The candidate path set does not need to be prepared in advance and the best paths are generated while solving the subproblems in our approach. A heuristic algorithm based on the solving procedure of the Lagrangean relaxation will be developed to obtain a primal feasible solution in the next two sections.

Network Model and Definition

A virtual circuit communications network is modeled as a graph where the processors are represented by nodes and the communication channels are represented by arcs. Let $V = \{1, 2, 3, ..., \mathrm{N}\}$ be the set of nodes in the graph and let L denote the set of communication links in the network. Let W be the set of origin-destination (O-D) pairs (commodities) in the network. For each O-D pair $w \in W$, the arrival of new traffic is modeled as a Poisson process with rate r_w (packet/sec). To reduce the problem's complexity, we assume that each O-D pair w, the overall traffic is transmitted over one path in the set P_w. For each link $l \in L$, the capacity is C_l packets/sec.

For each O-D pair $w \in W$, let x_p be 1 when $p \in P_w$ is used to transmit packets for O-D pair w and 0 otherwise. In a virtual circuit network, all of the packets in a session are transmitted over exactly one path from the origin to the destination. Thus $\sum_{p\in P_w} x_p = 1$. For each path p and link $l \in L$, let δ_{pl} denote the indicator function which is 1 if link l is on path p and 0 otherwise. Then, the aggregate flow over link l, denote as g_l, is $\sum_{p\in P_w}\sum_{w\in W} x_p r_w \delta_{pl}$.

In the network, there is a buffer for each outbound link. Using Kleinrock's independence assumption [6], the arrival of packets to each buffer is a Poisson process where the rate is the aggregate flow over the outbound link. It is assumed that the transmission time for each packet is exponential distributed with mean C_l^{-1}. Thus, each buffer is modeled as an M/M/1 queue, as considered in [3, 7, 8].

Problem Formulation

The following notations are used in the formulation.

Input values:

N^F : the set of nodes in the network.

L : the set of communication links in the communication network.

W : the set of source-destination (SD) pairs.

W_n : the set of SD pairs where node n is the source node.

r_w :(packets/sec.):the arrival rate of new traffic of each O-D pair $w \in W$,which is modeled of Poisson process for illustration purpose.

C_l : (packets/sec.),the capacity of each link $l\in L$.

P_w : a given set of of simple directed paths from the origin to the destination of O-D pair $w\in W$.

g_l : the aggregate flow over link l, which is equal to $\sum_{p\in P_w}\sum_{w\in W} x_p r_w \delta_{pl}$.

δ_{pl} : 1 if path p uses link l; 0 otherwise.

D_w : the maximum allowable end to end delay for O-D pair $w\in W$.

Decision variables:

α :percentage of capacity usage on maximum congested link.

x_p : 1 if path p is selected, 0 otherwise.

The formulation is modeled as the following integer linear programming problem.

Problem P

$$\min \ \alpha$$

Subject to:

$$g_l = \sum_{p\in P_w}\sum_{w\in W} x_p r_w \delta_{pl} \le \alpha C_l \qquad \forall l \in L \qquad (1)$$

$$\sum_{l\in L}\sum_{p\in P_w} \frac{x_p \delta_{pl}}{C_l - g_l} \le D_w \qquad \forall w \in W \qquad (2)$$

$$\sum_{p\in P_w} x_p = 1 \qquad \forall w \in W \qquad (3)$$

$$x_p = 0 \text{ or } 1 \qquad \forall p \in P_w, w \in W \qquad (4)$$

$$0 \le \alpha \le 1 \qquad (5)$$

The objective function is to minimize the largest utilization on the most congested link.

For the purpose of applying Lagrangean relaxation method, we transform the above problem formulation into an equivalent formulation P_{II}. In P_{II}, two auxiliary variables are introduced: y_{wl} is defined as $\sum_{p\in P_w} x_p \delta_{pl}$ and f_l denotes the estimate of the aggregate flow.

Decision variables:

α :percentage of capacity usage on maximum congested link.

x_p : 1 if path p is selected, 0 otherwise.

y_{wl} : 1 if source-destination pair w uses link l, 0 otherwise.

Problem P_{II}

$$\min \ \alpha$$

Subject to:

$$f_l \le \alpha C_l \qquad \forall l \in L \qquad (1)$$

$$\sum_{l\in L} \frac{y_{wl}}{C_l - f_l} \le D_w \qquad \forall w \in W \qquad (2)$$

$$\sum_{p\in P_w} x_p \delta_{pl} \le y_{wl} \qquad \forall w \in W, l \in L \qquad (3)$$

$$g_l \le f_l \qquad \forall l \in L \qquad (4)$$

$$\sum_{p\in P_w} x_p = 1 \qquad \forall w \in W \qquad (5)$$

$$x_p = 0 \text{ or } 1 \qquad \forall p \in P_w, w \in W \qquad (6)$$

$$y_{wl} = 0 \text{ or } 1 \qquad \forall w \in W, l \in L \qquad (7)$$

$$0 \le \alpha \le 1 \qquad (8)$$

$$0 \le f_l \le C_l \qquad \forall l \in L \qquad (9)$$

Redundant constraints associated with these auxiliary variables (3) ,(4),(7) and (9) are added.

Lagrangian relaxation is a general solution strategy for solving mathematical programs that permits us to decompose original problems into several subproblems such that we can exploit their special embedded structures.

We use Lagrangean relaxation to the heterogeneous Minmax end to end delay problem and decompose the original problem into several subproblems in next section.

3.Lagrangean relaxation and problem decomposition

We first dualize Constraints (1), (2), (3) and (4) to Problem P_{II} to obtain the following Lagrangean relaxation problem.

Problem (Dual_P):

$$Z_{dual}(\rho) = \min\{[1-\sum_{l\in L} v_l c_l]\alpha + [\sum_{w\in W} s_w(\sum_{l\in L}\frac{y_{wl}}{c_l - f_l} - D_w)] + \sum_{w\in W}\sum_{l\in L} t_{wl}(\sum_{p\in P_w} x_p \delta_{pl} - y_{wl}) + \sum_{l\in L}[u_l(g_l - f_l) + v_l f_l]\} \quad -(*)$$

subject to constraints (5), (6), (7), (8), and (9) .

$$\sum_{p\in P_w} x_p = 1 \qquad \forall w \in W \qquad (5)$$

$$x_p = 0 \text{ or } 1 \qquad \forall p \in P_w, w \in W \qquad (6)$$

$$y_{wl} = 0 \text{ or } 1 \qquad \forall w \in W, l \in L \qquad (7)$$

$$0 \le \alpha \le 1 \qquad (8)$$

$$0 \le f_l \le C_l \qquad \forall l \in L \qquad (9)$$

Reorganizes formulation (*), dual (P) becomes =>

$$Z_{dual}(\rho) = \min\{[1-\sum_{l\in L} v_l c_l]\alpha + [\sum_{w\in W}\sum_{l\in L}\sum_{p\in P_w}(t_{wl} + u_l r_w) x_p \delta_{pl}] + [\sum_{l\in L}(\frac{\sum_{w\in W} s_w y_{wl}}{c_l - f_l} - \sum_{w\in W} t_{wl} y_{wl} + (v_l - u_l) f_l) - \sum_{w\in W} s_w D_w]\}$$

subject to Constraints (5), (6), (7), (8), and (9) and vectorρ= (s,t,u,v) is the non-negative Lagrangean multiplier.

Problem (Dual_P) can be decomposed into following three independent subproblems (S1, S2 and S3) by separating the decision variables α, x, y. Therefore, we have $Z_{dual}=Z_{S1}+Z_{S2}+Z_{S3}-\sum_{w\in W} s_w D_w$, where

$$Z_{S1}(\boldsymbol{v}) = \min \left(1-\sum_{l\in L} v_l c_l\right)\alpha$$

subject to Constraint (8),

$$Z_{S2}(\boldsymbol{t},\boldsymbol{u}) = \min \left[\sum_{w\in W}\sum_{l\in L}\sum_{p\in P_w}(t_{wl}+u_l r_w)x_p\delta_{pl}\right]$$

subject to Constraints (5) and (6) , and

$$Z_{S3}(s,t,u,v) = \min\{\sum_{l\in L}(\frac{\sum_{w\in W} s_w y_{wl}}{c_l-f_l}-\sum_{w\in W} t_{wl}y_{wl}+(v_l-u_l)f_l)\}$$

subject to Constraints (7) and (9).

Solving Subproblem 1

Subproblem S1 is a problem for decision variable α. Variableαis set to 1 if the corresponding cost $1-\sum_{l\in L} v_l c_l$ is negative; otherwiseαis set to 0. Subproblem 1 runs on O(L) computation time.

```
procedure subproblem1;
begin
Z:=1;
for all link l∈L do
Z:= Z−v_l×C_l;
if Z>0 then α:=0 ;
else α:=0;
end;
```

Solving Subproblem 2

Subproblem S2 is a problem for decision variable $\boldsymbol{x}$. It consists of $|W_n|$ independent problems. Each one is an edge-disjoint-path problem rooted at the given source node and destined to all destination nodes for the SD pairs with non-zero traffic demand. To solve the problem, one can view the input network as a graph. This graph contains (L) arcs and (N) nodes. We set each arc l have C_l capacity (it means that the transmission time for each packet is exponentially distributed with mean C_l) and non-negative arc weight, $(t_{wl}+u_l r_w)$. In such graph, the subproblem is a minimum cost flow problem to send minimum cost flow from the source node to all its destination nodes with specified traffic demands. We use traditional minimum cost flow algorithm such as successive shortest path algorithm [9] to solve the problem.

```
procedure subproblem2;
begin
    for each link l∈L do
            cost_lw:=t_wl+u_l r_w;
    for each node src∈S_n do
        run successive-shortest-path(src, cost) to determine x;
    update Z_s2;
end;
```

Solving Subproblem 3

Subproblem S3 is a problem for decision variable $\boldsymbol{y}$. It consists of $|L|$ independent problems.

For each link $l\in L$:

$$\min[\frac{\sum_{w\in W} s_w y_{wl}}{c_l-f_l}-\sum_{w\in W} t_{wl}y_{wl}+(v_l-u_l)f_l]$$

subject to (5) and (8).

For different values of f_l, the value of y_{wl} for minimum objective function, denoted as $y_{wl}^*(f_l)$ may be different. As an example, consider the case that f_l =0. The objective function is minimized by assigned $y_{wl}^*(0)$ to 1 if $(\frac{s_w}{c_l}-t_{wl})\le 0$ and to 0 otherwise. We define a set of *break points* of f_l as those points where $(\frac{s_w}{C_l-f_l}-t_{wl})=0$ for each w. These break points are sorted and denoted as $f_l^1, f_l^2, \ldots, f_l^n$. Note that there are at most |W| break points. We observe that when $f_l^i \le f_l \le f_l^{i+1}$ the value, *the value of* $y_{wl}^*(f_l)$ *remains constant for all* $w\in W$. Within the above internal, $y_{wl}^*(f_l)$ is 1 if $(\frac{s_w}{c_l-f_l}-t_{wl})\le 0$ and is 0 otherwise. Therefore, within an interval, $[f_l^i, f_l^{i+1})$, the objective is only a function of f_l, and the minimum point within the interval can be found analytically. By examining at most |W| +1 intervals, we can find the global minimum point by comparing those local minimum points.

When examining an interval, we first determine $y_{wl}^*(f_l^i)$ within the interval for each w. We denote $\sum_{w\in W} s_w y_{wl}^*(f_l^i)$ as a_l and $\sum_{w\in W} t_{wl} y_{wl}^*(f_l^i)$ as b_l. Note that a_l and b_l are non-negative. Within the interval, the objective function can then be expressed as: $Z_{sub3\_l}=\frac{a_l}{C_l-f_l}-b_l+(v_l-u_l)f_l$. A typical curve of the objective function vs. f_l within the interval $f_l^i \le f_l \le f_l^{i+1}$ is shown in Figue 1. The curve of the objective function vs. f_l is shown in Fig. 2. The local minimum point is either at the boundary point, f_l^i or f_l^{i+1} , or at point $f_l^* = C_l\sqrt{\frac{a_l}{u_l-v_l}}, ((u_l-v_l)\neq 0)$.

```
procedure subproblem3;
begin
      Step 1. Solve (s_w/(C_l − f_l) − t_wl = 0) for each O-D pair w, call
              them the break points of f_l.
      Step 2. Sorting these break points and denoted as:
              f_l^1, f_l^2 ..........., f_l^n.
      Step 3. At each interval, f_l^i ≤ f_l ≤ f_l^(i+1), y_wl(f_l) is 1 if
              s_w/(C_l − f_l) − t_wl ≤ 0 and is 0 otherwise.
      Step 4. Within the interval, f_l^i ≤ f_l ≤ f_l^(i+1), let a_l be
              Σ_(w∈W) s_w y_wl(f_l) and b_l be Σ_(w∈W) t_wl y_wl(f_l), then the local
              minimum is either at the boundary point, f_l^i or f_l^(i+1),
              or at point f_l^* = C_l − sqrt(a_l/(u_l − v_l)).
       Step 5. The global minimum point can be found by comparing
               these local minimum points.
end;
```

Subgradient Optimization Procedure

From the weak Lagrangian duality theorem, $Z_{dual}(\boldsymbol{\rho})$ is a lower bound of the Problem (P) for any non-negative Lagrangean multiplier vector $\boldsymbol{\rho}=(s, t, u ,v)\ge \boldsymbol{0}$. Naturally, one wants to determine the largest lower bound by $Z_{lower\_bound}=\max_{\rho\ge 0} Z_{dual}(\rho)$ The subgradient method can be applied to solve (11). The solution to Problem (Dual_P) at iteration k of the subgradient optimization procedure is given below. In subgradient solution procedure, the Lagrangian multiplier vector $\boldsymbol{\rho}$ is updated by $\rho_{k+1}=\rho_k+\theta_k b_k$ where $\boldsymbol{b}$ is a subgradient of $Z_{dual}(\boldsymbol{\rho})$ with vector size $|W+LW+L+L|$. The step size θ_k is determined by

$$\theta_k=\frac{\lambda_k(UB-Z_{dual}(\rho))}{\|b_k\|^2}$$

UB is an upper bound obtained from a heuristic solution described in the next section and λ_k is a constant in a range from 0 to 2.
The details of this procedure see below:

```
procedure update-step-size;
begin
    ∀_{1≤i≤|W|}  b_i := Σ_{l∈L} y_wl/(C_l − f_l) − D_w ;
    ∀_{|W|+1≤i≤|W|+|L||W|}  b_i := Σ_{p∈P_w} x_p δ_pl − y_wl ;
    ∀_{|W|+|L||W|+1≤i≤|W|+|L||W|+|L|}  b_i := g_l − f_l ;
    ∀_{|W|+|L||W|+|L|+1≤i≤|W|+|L||W|+|L|+|L|}  b_i := f_l − αC_l ;
    step size  θ = λ(UB − Z_dual)/||b||^2 ;
end;
```

```
procedure update-multiplier;
begin
    for l:=1 to |W| do
        s_w:=[s_w+θ(Σ_{l∈L} y_wl/(C_l − f_l) − D_w)]^+ ;
    for l:=1 to |L| do
        for w:=1 to |W| do
            t_lw:=[t_lw+θ(Σ_{p∈P_w} x_p δ_pl − y_wl)]^+ ;
    for l:=1 to |L| do
        u_l:=[u_l+θ(g_l − f_l)]^+ ;
    for l:=1 to |L| do
        v_l:=[v_l+θ(f_l − αC_l)]^+ ;
end;
```

Summary of Lagrangean Relaxation Method

The algorithms are described below: LRM denotes the Lagrangean relaxation method.

```
algorithm LRM;
begin
    multiplier vector s:=0, t=0 , u:=0 and v:=0;
    UB:=1 and LB:=0;
    unimproved_count:=0;
    step size coefficient λ:=2;
    for each k:=1 to MaxIteration do
    begin
        run subproblem1, subproblem2 and subproblem3;
        Z_dual=Z_s1+Z_s2+Z_s3−Σ_{w∈W} s_w D_w;
        if Z_dual>LB then LB:=Z_dual and unimproved_count:=0;
        else unimproved_count:=unimproved_count+1;
        if unimproved_count>=Max_unimproved_count then
            λ:=λ/2 and unimproved_count:=0;
        run primal-heuristic;
        if ub<UB then UB:=ub;
        run update-step-size;
        run update-multiplier;
    end;
end.
```

4. Lagrangean-based Heuristic Algorithm

Since the Lagrangean relaxation is obtained by the relaxation of some constraints from the problem formulation, the solution to the dual problem might be infeasible for the original primal problem resulting from dissatisfaction of those relaxed constraints. However, such solution can still be used as a base to develop efficient heuristic algorithms to seek feasible solutions and obtain upper bounds for the original problem. In practice, in each iteration of the subgradient solving procedure, the solution of Lagrangean relaxation is used to obtain a lower bound of the primal problem. In addition, we verify the feasibility of the solution in the constraints of primal problem. If the solution is feasible, it is used to calculate an upper bound of the primal problem (Actually it is an optimal solution.). If the solution is not feasible, the following heuristic is applied to find a feasible solution.

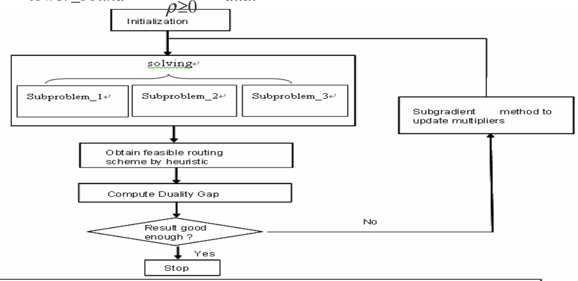
Proposed Lagrangean-based Heuristic Algorithm

Based on the solution obtained from solving Lagrangean relaxation in each iteration. The traffic demand is then routed onto the network with cost assigned as the same as they assigned in the Lagrangean relaxation. Traffic demands are then routed onto the network for each SD pair sequentially (from short delay required connections to long delay required ones) by applying Dijkstra's shortest path algorithm. The capacity of those arcs used by the above accepted paths are updated by subtracting the flow of this connection from the capacity of this link. If the utilization of a link will become greater than *the best known lower bound* (*LB*)×|C_l| when a further virtual circuit setups on it, the weight of this arc is replaced by multiply a constant term on its weight as a penalty for avoiding further setup paths on this link. The process continues until all of the traffic demands are satisfied or the network cannot accommodate the traffic request. A feasible solution is obtained in the former case.

Evaluation of the Feasible Schedule

From the weak Lagrangian duality theorem, $Z_{dual}(\boldsymbol{\rho})$ is a lower bound of the Problem (*P*) for any non-negative Lagrangean multiplier vector $\boldsymbol{\rho} = (s, t, u, v) \geq \mathbf{0}$. Naturally, one wants to determine the largest lower bound by

$$Z_{lower\_bound} = \max_{\rho\geq 0} Z_{dual}(\rho) \qquad (11)$$




```
procedure primal-heuistic;
begin
    for each link l∈L do
    begin
        for each o-d pair w∈W do
            cost_l:=t_wl+u_l r_w;
    end;
    no-path-setup:=0;
    repeat
        for each SD pair sd:=1 to |S| do
        begin
            if no_path_setup_sd<1 then
            begin
                src=source(sd);
                dest=destination(sd);
                run Dijkstra's-shortest-path(cost, src, dest)
                if the shortest path exists then
                    for all link l on the shortest path do
                    begin
                        f_l:= f_l +r_w;
                        if f_l > lb*C_l
                          cost_l:= cost_l * penalty;
                    end
                else
                    return infeasible;
            end;
        end;
    until all SD demand satisfied;
    update upper bound ub;
end;
```

5. Experimental Results

Performance Comparisons

We first carried out numerical computation of the lower and upper bound values of α, the maximum aggregate arrival rates of a link divided by the capacity of each link, C_l, using our LRM-based method and a Linear Programming Relaxation (LPR)-based method. In the computation, we considered three widely used networks. They are: NSFNET with 14 nodes and 42 links; PACBELL with 15 nodes and 42 links; and GTE with 11 nodes and 46 links.

In the LRM-based heuristic algorithm, we adopted a penalty term of 2. In addition, if the Lagrangean lower bound remains unimproved for 50 iterations (UC=50), the step size coefficient (λ_k) would be divided by two. The simulation was written in the C language and terminated at the end of 2000 iterations and operated on a PC running Windows XP with a 1.8 GHz CPU power.

In the LPR-based method, by removing Constraints (6) and (7), the original Integer Linear Programming (ILP) problem is relaxed to a Linear Programming (LP) problem. Thus, the solution to the relaxed problem is a legitimate lower bound of the original ILP problem. To obtain an upper bound, we also develop a corresponding heuristic algorithm. The algorithm ranks all SD pairs in accordance with the desired packet delay. The next feasible path founded in the LP solution is then assigned to the SD pair with the smallest packet delay. There may be multiple feasible paths for an SD pair; we select the shortest path with the largest x_p value in the algorithm. The path assignment process repeats until either the traffic demands of all SD pairs are satisfied (i.e., feasible), or there is no remaining resource (i.e., infeasible). In the simulation, the LP problem was solved using the *CPLEX* software, operating in the same PC environment previously described.

Numerical results for the NSFNET, PACBELL, and GTE are summarized in Table II, III, and IV, respectively. The traffic demands (i.e., the traffic arrival rate) for all SD pairs are randomly determined with their mean value shown in the first column of the tables. Moreover, the Gap in the third column of the tables is computed as the ratio of the difference of the upper and lower bounds to the lower bound in percentage.

As shown in Table II for NSFNET, the LPR-based method reaches a low guarantee of 20% gap, incurring high CPU computation time. Compared to it, the LRM-based method achieves ideal lower and upper bounds (gap< 5%) under all four traffic demand cases except case 1. The algorithm also improves the CPU computation time by one order of magnitude. We discover that, even though both methods achieve optimal lower bounds, the LRM-based heuristic algorithm arrives at much improved upper bounds due to the use of the Lagrangean multipliers derived upon seeking the Lagrangean relaxation solution.

In Table III for PACBELL, the LPR-based method reaches a low guarantee of 29% gap. Compared to it, the LRM-based method again achieves ideal lower and upper bounds (<8%). The LRM-based algorithm also improves the CPU computation time by two order of magnitude. It is worth mentioning that in the case of the mean traffic demand being equal to 3.0, while the LPR-based method fails to obtain a feasible solution, the LRM-based method arrives at the optimal solution. Finally, in Table IV for the GTE network, the LRM-based method outperforms the LPR-based method in both the solution superiority and the computation time in all traffic cases. Specifically, the LPR-based method again reaches fairly low guarantee of 27% gap. The method produces a non-optimal solution but with an improved guarantee of 13% gap. This justifies the viability of the LRM-based method for providing efficient QoS routing method.

Table 2 Numerical results for NSFNET Network

Demand	LB	LP_UB	LP_CPU (sec)	LP_Diff %	LP Feasibility	Lag_UB	Lag_CPU (sec)	Lag_Diff %	Lag Feasibility
0.5	0.134885	0.15625	1108	15.840	Yes	0.156250	28	15.840	Yes
1.0	0.260588	0.31250	1694	19.921	Yes	0.281250	41	7.929	Yes
1.5	0.403905	0.46875	1754	16.055	Yes	0.406250	54	0.581	Yes
2.0	0.533426	0.62500	1809	17.167	Yes	0.562500	64	5.450	Yes
2.5	0.646050	0.75000	1947	17.152	Yes	0.656250	71	1.579	Yes
3.0	0.820311	0.90625	2027	10.476	Yes	0.843750	82	2.857	Yes

Table 3 Numerical results for PACBELL Network

Demand	LB	LP_UB	LP_CPU (sec)	LP_Diff %	LP Feasibility	Lag_UB	Lag_CPU (sec)	Lag_Diff %	Lag Feasibility
1.0	0.289678	0.375000	2335	29.454	Yes	0.312500	41	7.878	Yes
1.5	0.448961	0.531250	2151	18.329	Yes	0.468750	53	4.408	Yes
2.0	0.596959	0.687500	2414	15.167	Yes	0.625000	67	4.697	Yes
2.5	0.757723	0.906250	2697	19.602	Yes	0.781250	80	3.105	Yes
3.0	0.951497	NA	2651	NA	No	0.96875	90	1.812	Yes

Table 4 Numerical results for GTE Network

Demand	LB	LP_UB	LP_CPU (sec)	LP_Diff %	LP Feasibility	Lag_UB	Lag_CPU (sec)	Lag_Diff %	Lag Feasibility
1.0	0.177474	0.218750	410	23.257	Yes	0.187500	65	5.649	Yes
1.5	0.270522	0.343750	440	27.069	Yes	0.281250	85	3.966	Yes
2.0	0.348616	0.406250	712	16.532	Yes	0.375000	103	7.568	Yes
2.5	0.406115	0.468750	830	15.423	Yes	0.437500	124	7.728	Yes
3.0	0.468669	0.562500	897	20.020	Yes	0.531250	135	13.353	Yes
3.5	0.546826	0.687500	912	25.726	Yes	0.593750	148	8.581	Yes
4.0	0.616869	0.687500	954	11.450	Yes	0.656250	157	6.384	Yes
4.5	0.655807	0.750000	1024	14.363	Yes	0.718750	160	9.598	Yes
5.0	0.701118	0.781250	1231	11.429	Yes	0.75000	162	6.972	Yes
5.5	0.804162	0.906250	1403	12.695	Yes	0.875000	174	8.809	Yes

5. Conclusions and Future Works

In this paper, we have improved a QoS routing problem using a Lagrangean Relaxation based approach augmented with an efficient primal Heuristic algorithm, called LRH. With the aid of generated Lagrangean multipliers and lower bound indexes, the primal heuristic algorithm of LRH achieves a near-optimal upper-bound solution. Our method has three major characteristics. First, we start to consider user's perspective and system's perspective jointly. Second, in our routing procedure, the candidate path set does not need to be prepared in advance and the best paths are generated while solving the subproblems in our approach. Third, our method can both provide the upper bound and lower bound to the problem, and this distinguishing feature can help us to verify the performance of our solutions.

Future Works

We are able to reconfigure the virtual topology to adapt to changing traffic patterns. Some reconfiguration studies on virtual networks have been reported before [10, 11, 12]; however, these studies assumed that the new virtual topology *was known a priori*, and were concerned with the cost and sequence of branch-exchange operations to transform from the original virtual topology to the new virtual topology. We propose a methodology to obtain the new virtual topology, based on optimizing a given objective function, as well as minimizing the changes required to obtain the new virtual topology from the current virtual topology. This approach would result in the minimum number of switch re-tunings, thus minimizing the number of disrupted virtual paths. Consequently, this approach also minimizes the time it takes to complete the reconfiguration process.

In the ideal situation, given a small change in the traffic matrix, we would prefer for the new virtual topology to be largely similar to the previous virtual topology, in terms of the constituent virtual paths and the routes for these virtual paths, i.e., we would prefer to minimize the changes needed to adapt from the existing virtual topology to the new topology. More formally, it would be preferable if a large number of the δ_{pl} variables retain the same values in the two solutions, without compromising the quality of the solution (in terms of minimizing the congested link utilization).

Let us consider the snapshot of two traffic matrices, λ_{sd}^1 and λ_{sd}^2, taken at two not-too-distant time instants. We assume that there is a certain amount of correlation between these two traffic matrices. Given a certain traffic, there may be many different virtual topologies, each of which has the same optimal value with regard to the objective function. But we will terminate after the first such optimal optimal solution is found. Our reconfiguration algorithm finds the virtual topology corresponding to λ_{sd}^2 which matches "closest" with the virtual topology corresponding to λ_{sd}^1 (based on our above definition of "closeness").

Reconfiguration Algorithm

We perform the following sequence of actions:

1). Generate formulations $F(1)$ and $F(2)$ corresponding to traffic matrices λ_{sd}^1 and λ_{sd}^2, respectively, based on the formulation in Section 3.

2). Derive solutions $S(1)$ and $S(2)$, corresponding to $F(1)$ and $F(2)$, respectively. Denote the variables' values in $S(1)$ as $x_p(1)$ and $y_{wl}(1)$, and those in $S(2)$ as $x_p(1)$ and $y_{wl}(1)$, respectively. Let the value of the objective function for $S(1)$ and $S(2)$ be OPT_1 and OPT_2, respectively.

3). Modify ($F(2)$ to $F'(2)$) by adding the new constraint

$$\alpha = OPT_2 \tag{10}$$

This ensures that all the virtual topologies generated by $F'(2)$ would be optimal with regard to the objective function. The new objective function for $F'(2)$ is

$$\text{Minimize: } \sum_w \sum_l |y_{wl}(1) - y_{wl}(2)| \tag{12}$$

Note that the mod operation, $|x|$, is a nonlinear function. If we assume that y_{wl} can only take on binary values, then (12) become linear, i.e., if $y_{wl}(1) = 1$, then $|y_{wl}(1) - y_{wl}(2)| \equiv (1 - y_{wl}(2))$; else if $y_{wl}(1) = 0$, then $|y_{wl}(1) - y_{wl}(2)| \equiv y_{wl}(2)$. Hence, $F'(2)$ may be solved directly.

References

[1] B. Gavish and S.L. Hantler, "An Algorithm for Optimal Route Selection in SNA Networks". IEEE Trans. on Communications COM-31, pp. 1154-1160, Oct. 1983.

[2] L. Fratta and M. Cerla and L. Kleinrock, "The flow deviation method: An approach to tore-and-forward communication network design". Networks 3, pp. 97-133 1973.

[3] F. Y. S. Lin and J. R. Yee, "A New Multiplier Adjustment Procedure for the Distributed Computation of Routing Assignments in Virtual Circuit Data Networks". ORSA Journal on Computing, Vol. 4, No. 3, Summer 1992.

[4] P. J. Courtois and P. Semal, "An Algorithm for the Optimization of Nonbifurcated Flows in Computer Communication Networks". Performance Evaluation, pp. 139-152, 1981.

[5]K. T. Cheng and F. Y. S. Lin, "Minmax End-to-End Delay Routing and Capacity Assignment for Virtual Circuit Networks". Proc. IEEE Globecom, pp. 2134-2138, 1995.

[6]L. Kleinrock, "Queueing Systems". Wiley-Interscience, New York, Vol.1 and Vol.2 1975-1976

[7] A. Shaikh, J. Rexford, and K. Shin, "Dynamics of quality-of-service routing with inaccurate link-state information". U. of MI Tech. rep. CSE-TR-350-97, Nov. 1997.

[8] B. Awerbuch et al., "Throughput- Competitive On-line Routing". 34th Annual Symp. Foundations of Comp. Sci., Pala Alto, CA, Nov. 1993.

[9]R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network Flows: Theory, Algorithms, and Applications". Prentice-Hall, 1993.

[10] D. Bienstock and O. Gunluk, "A degree sequence problem related to network design". Networks, vol. 24, no. 4, pp. 195–205, July 1994.

[11]J. F. P. Labourdette, G.W. Hart, and A. S. Acampora, "Branch-exchange sequences for reconfiguration of lightwave networks". IEEE Trans. Commun., vol. 42, pp. 2822–2832, Oct. 1994.

[12] G. N. Rouskas and M. H. Ammar, "Dynamic reconfiguration in multihop WDM networks". J. High Speed Networks, vol. 4, no. 3, pp. 221–238, 1995.