The Primal Simplex Approach to the QoS Routing Problem

Ying Xiao, Krishnaiyan Thulasiraman
University of Oklahoma, Norman, OK
{ying_xiao, thulasi}@ou.edu
Guoliang Xue
Arizona State University, Tempe, AZ
xue@asu.edu

Abstract

Quality-of-Service (QoS) routing problem requires the determination of a minimum cost path from a source node s to a destination node t in a data network such that the delay of the path is bounded by Δ *(> 0). This problem also known as the constrained shortest path (CSP) problem is NP-hard. So, heuristics and approximation algorithms have been presented in the literature. Among the heuristics, the LARAC algorithm, based on the dual of the LP relaxation or the Lagrangian relaxation of the CSP problem is very efficient. In this paper we study the primal simplex approach to the LP relaxation of the CSP problem and present an approximation algorithm to this problem. Several issues relating to efficient implementations of our approach are discussed. Experimental results comparing the performance of the new algorithm with that of the LARAC algorithm are presented.*

1. Introduction

Recently there has been considerable interest in the design of communication protocols that deliver certain performance guarantees that are usually referred to as Quality of Service (QoS) guarantees. A problem of great interest in this context is the QoS routing problem that requires the determination of a minimum cost path from a source node to a destination node in a data network that satisfies a specified upper bound on the delay of the path. This problem is also known as the constrained shortest path (CSP) problem. The CSP problem is known to be NP-hard [1]. So, in the literature, heuristic approaches and approximation algorithms have been proposed. Heuristics, in general, do not provide performance guarantees on the quality of the solution produced, though they are usually fast in practice. On the other hand, ε-approximation algorithms deliver solutions within arbitrarily specified precision requirement. References [2]–[4] and the references therein contain most of the current literature on approximation algorithms for the CSP problem. As regards heuristics, the LHWHM algorithm [5] is a simple heuristic which is very fast (requiring only one or two invocations of Dijkstra's shortest path algorithm) and produces solutions which are usually found to be of acceptable quality in practice. Reference [6] also discusses further enhancements of the LHWHM algorithm. There are heuristics that are based on sound theoretical foundation. These algorithms are based on solutions to the dual of the linear programming relaxation of the CSP problem. The first such algorithm was reported in [7] by Handler and Zang. This is based on a geometric approach (what is also called the hull approach by Mehlhorn and Ziegelmann [8]). More recently, in an independent work, Jüttner *et al*. [9] developed the LARAC algorithm which also solves the Lagrangian relaxation of the CSP problem. In contrast to the geometric method, they used an algebraic approach. In another independent work Blokh and Gutin [10] defined a general class of combinatorial optimization problems of which the CSP problem is a special case, and proposed an approach to this problem. In a recent work, Xiao *et al*. [11] drew attention to the fact that the algorithms in [7]-[10] are equivalent. In view of this equivalence, we shall refer to these algorithms simply as the LARAC algorithm. Mehlhorn and Ziegelmann [8] have developed several insightful results. In an unpublished work [12], Jüttner established the strong polynomiality of the LARAC algorithm. Ziegelmann [13] provides a fairly complete list of references to the literature on the CSP problem. Another reference where one could find a survey of QoS routing research is [14]. A recent work on QoS routing is [15] where the authors study and present approximation algorithms for minimum cost disjoint paths selection under delay constraints. In [16] the authors study the disjoint paths selection problem using the primal simplex method.

In this paper we present a novel approach to the QoS routing problem, making a departure from currently available approaches. We study the problem using the primal simplex method of linear programming and exploiting certain structural properties of networks. The rest of the paper is organized as follows. In section 2, we define the CSP problem and present its integer linear programming (ILP) formulation as well as its linear programming (LP) relaxation. This formulation is the same as the LP formulation of the minimum cost flow problem except for an additional constraint due to the delay requirement. This additional constraint gives rise to several questions that need to be investigated to achieve an efficient implementation of the primal simplex method. This leads us to the definition of an equivalent problem on a transformed network, called the TCSP problem. Sections 3 – 9 discuss the revised simplex method and its application on RELAX-TCSP, the relaxed form of the TCSP problem. The complete algorithm called NBS algorithm and its pseudo-polynomial time complexity are presented in section 10. This section also shows how to extract an approximate solution to the original CSP problem from the optimum solution to the RELAX-TCSP problem and derives bounds on the quality of this solution with respect to the optimum solution. In section 11, experimental results comparing the NBS algorithm with the LARAC algorithm are presented. Section 12 concludes with a summary of the main contributions. To conserve space proofs of results are omitted.

2. The CSP problem and an equivalent transformed problem (TCSP)

In this section, we first define the constrained shortest path problem. To achieve an efficient implementation of our algorithm, we shall define an equivalent problem on a transformed network. Then we present the integer linear programming formulation of this transformed problem called TCSP problem. Relaxing the integrality constraints of the TCSP problem leads to the RELAX-TCSP problem.

Definition 1. Constrained Shortest Path (CSP) Problem: Consider a directed network $G(V, E)$ where V is the set of nodes and E is the set of links of the network. Each link $(u, v) \in E$ is associated with two integer weights $c_{uv} > 0$ (cost) and $d_{uv} > 0$ (delay). For any path p (or cycle with a given orientation) define the cost $c(p)$ and delay $d(p)$ of p as

$$c(p) = \sum_{(u,v) \in p^+} c_{uv} - \sum_{(u,v) \in p^-} c_{uv},\ d(p) = \sum_{(u,v) \in P^+} d_{uv} - \sum_{(u,v) \in P^-} d_{uv},$$

where p^+ (p^-) is the set of forward (backward) links on p as we traverse p from the start node to the end node. ■

A path is called a directed path if there are no backward links in the path. Given two nodes s, t and integer $\Delta > 0$, a directed s-t path p is said to be feasible if $d(p) \le \Delta$. The CSP problem is to find an s-t path $p^* = \arg\min\{c(p) | p \text{ is a feasible } s\text{-}t \text{ path}\}$.

We use the terms "link" and "arc" interchangeably. Sometimes we may use $c(u, v)$, $d(u, v)$ and e_{uv} to represent the link cost c_{uv}, link delay d_{uv} and the link $e = (u, v)$, respectively. Without loss of generality we assume that for every node i, there is a directed path from i to the destination node t.

Transformation of the CSP problem: To solve the CSP problem efficiently, we transform it to an equivalent one on a transformed network defined as follows:

- The graph of the transformed network is the same as that of the original problem, that is, $G(V, E)$.
- For $(u, v) \in E$, d'_{uv} and c'_{uv} in the transformed problem are given by $d'_{uv} = 2\, d_{uv}$ and $c'_{uv} = c_{uv}$.
- The new upper bound Δ' in the transformed problem is given by $\Delta' = 2\,\Delta + 1$.

Theorem 1. An s-t path p^* is a feasible solution (an optimal solution) to the CSP problem iff it is a feasible solution (an optimal solution) to the TCSP problem. ■

In the rest of the paper, we only consider the TCSP problem. Even though many of our results hold for the CSP problem too, Lemmas 4 and 7 and Theorems 4 - 5 hold true only for the TCSP problem. Also we use Δ (being odd) and d_{uv} (being even) to denote the delay bound and link delay in the transformed problem, respectively. Notice that the transformation does not change the cost of any path in the network. The TCSP problem can be formulated as the following ILP problem.

$$\text{TCSP: Min} \sum_{(u,v) \in E} c_{uv} \cdot x_{uv} \tag{1}$$

Subject to

$$\sum_{\{v|(u,v) \in E\}} x_{uv} - \sum_{\{v|(v,u) \in E\}} x_{vu} = \begin{cases} 1, & if \quad u = s; \\ -1, & if \quad u = t; \\ 0, & otherwise. \end{cases} \tag{2}$$

$$\sum_{(u,v) \in E} - d_{uv} \cdot x_{uv} - w = -\Delta \tag{3}$$

$$\forall\ (u, v) \in E,\ x_{uv} = 0 \text{ or } 1 \tag{4}$$

Note: The variable w in (3) is the slack variable corresponding to the delay constraint.

Relaxing the integrality constraint in (4) results in the RELAX-TCSP problem given below.

$$\text{TCSP: Min} \sum_{(u,v) \in E} c_{uv} \cdot x_{uv} \tag{5}$$

Subject to

$$\sum_{\{v|(u,v)\in E\}} x_{uv} - \sum_{\{v|(v,u)\in E\}} x_{vu} = \begin{cases} 1, & if \quad u = s; \\ -1, & if \quad u = t; \\ 0, & otherwise. \end{cases} \tag{6}$$

$$\sum_{(u,v)\in E} -d_{uv} \cdot x_{uv} - w = -\Delta \tag{7}$$

$$\forall\ (u, v) \in E, x_{uv} \geq 0 \tag{8}$$

Let the links be labeled as e_1, e_2 …, e_m and the nodes be labeled as 1, 2 …., n. We shall denote the delay of edge e_i as d_i and the cost of e_i as c_i. The incidence matrix of G has m columns, one for each link and n rows, one for each node. The rank of this matrix is (n - 1), and removing any row of this matrix will result in a matrix of rank (n - 1). We denote this resulting matrix as $\boldsymbol{H}$. We also assume that the row removed from the incidence matrix corresponds to node n. Also we assume that the column of $\boldsymbol{H}$ corresponding to link e_k will be denoted by the vector $\boldsymbol{h}_k$. For $e_k = (i, j)$, we have $\boldsymbol{h}_k = (h_{1,k}, ..., h_{i,k} ..., h_{j,k} ..., h_{n-1,k})^t$ with all components being 0 except for $h_{i,k} = 1$ and $h_{j,k} = -1$. Let

$$A = \begin{pmatrix} H & 0 \\ D & -1 \end{pmatrix} = (a_1, a_2 ... a_m, a_{m+1}), \tag{9}$$

where

$$\boldsymbol{D} = (-d_1, -d_2 ... -d_m), \tag{10}$$

$$a_i = \begin{pmatrix} h_i \\ -d_i \end{pmatrix}, i \leq m, \text{ and} \tag{11}$$

$$a_{m+1} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}. \tag{12}$$

Also, let $\boldsymbol{x}$ be the column vector of the m flow variables x_{uv} and the slack variable w, and $\boldsymbol{c}$ be the row vector of the costs (c_1, c_2 …, c_m, 0). Note that the cost of the slack variable is 0. The above LP formulation of the RELAX-TCSP problem can now be written in matrix form as follows.

RELAX-TCSP: Min: $\boldsymbol{c\,x}$

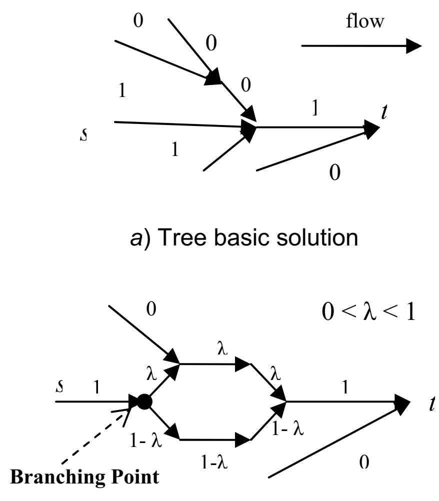
Subject to $\boldsymbol{A\,x} = \boldsymbol{b}$, (13)

$\boldsymbol{x} \geq 0$ for $\forall(u, v) \in E$, where $\boldsymbol{b} = (b_1 \ldots, b_{n-1}, -\Delta)^t$ with $b_s = 1$, $b_t = -1$ and $b_i = 0$ for $i \neq s, t$.

The rest of the paper is concerned with the simplex method based solution of RELAX-TCSP.

3. Basic solutions of RELAX-TCSP

Simplex method of linear programming starts with a basic solution and proceeds by constructing one basic solution from another. A basic solution consists of two sets of variables, basic and non-basic. For the RELAX-TCSP problem under consideration, all the non-basic variables in a basic solution will have zero values. Given a basic solution, we shall denote by G_b the subgraph of G corresponding to the basic variables (except the slack variable if it is in the basic solution) in this solution. The subgraph G_b will be called the subgraph of the basic solution. The non-singular submatrix of $\boldsymbol{A}$ defined by the basic variables is called a basis matrix or simply, a basis. In this section we present certain important properties of the basic solutions of the RELAX-TCSP problem.



a) Tree basic solution

b) Basic solution with a cycle

Fig.1. Structure of basic solution graph

Lemma 1 [17]: Let G (V, E) be a directed network with at least one cycle W (not necessary directed). Assigning an arbitrary orientation to W, let $\boldsymbol{U} = (u_1, u_2, u_3 \ldots u_m)^t$, where

$$u_j = \begin{cases} 1, & if\ e_j \in W\ and\ the\ orientation\ of \\ & e_j\ agrees\ with\ the\ orientation\ of\ W; \\ -1, & if\ e_j \in W\ and\ the\ orientation\ of \\ & e_j\ disagrees\ with\ the\ orientation\ of\ W; \\ 0, & otherwise. \end{cases}$$

Then, $\boldsymbol{H} \bullet \boldsymbol{U} = 0$. ■

We shall denote by $d(W)$ the signed algebraic sum of the delays of the links in a cycle W as we traverse around the cycle along the given orientation.

Lemma 2. The subgraph G_b of a basic solution contains at most one cycle (See Fig. 1). ■

Lemma 3. If there is a cycle W in G_b, then $d(W) \neq 0$. ■

Lemma 4. The flow vector $\boldsymbol{x}$ in (13) satisfies $0 \leq \boldsymbol{x} \leq 1$ and a link assumes positive flow (> 0) iff it is on some directed s-t path in G_b (See Fig. 1). If G_b contains no cycle or the cycle is link disjoint with the s-t paths in G_b, then the link flows are integers (0 or 1). ■

Lemma 5. The basis matrix contains the last row of $\boldsymbol{A}$

defined in (9) and the (n - 1) rows of $\boldsymbol{H}$. ■

4. Revised simplex method on the RELAX-TCSP problem

In this section, we first briefly present the different steps in the revised simplex method of linear programming which is described in detail in [18]. We then derive formulas required to identify the entering and the leaving variables that are needed to generate a new basic solution from a given basic solution.

4.1 Revised simplex method

Consider an arbitrary linear programming (LP) problem in the standard form.

Min $\boldsymbol{c}\,\boldsymbol{x}$
Subject to $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0$.

Here $\boldsymbol{A}$ is an $n \times (m + 1)$ matrix with $rank\ (\boldsymbol{A}) = n$, $\boldsymbol{x} = (x_1\ldots, x_{m+1})^t$, $\boldsymbol{c} = (c_1\ldots\ c_{m+1})$, $\boldsymbol{b} = (b_1, b_2, \ldots\ b_n)^t$. Each feasible basic solution $\boldsymbol{x^*}$ is partitioned into two sets, one set consisting of the n basic variables and the other set consisting of the remaining $m + 1 - n$ non-basic variables. This partition induces a partition of $\boldsymbol{A}$ into $\boldsymbol{B}$ and $\boldsymbol{A_N}$, a partition of $\boldsymbol{x}$ into $\boldsymbol{x_B}$ and $\boldsymbol{x_N}$ and a partition of $\boldsymbol{c}$ into $\boldsymbol{c_B}$ and $\boldsymbol{c_N}$, corresponding to the set of basic variables and the set of non-basic variables, respectively. The basis matrix $\boldsymbol{B}$ is nonsingular.

Revised Simplex Method:

Step 1: Solve the system $\boldsymbol{Y} \bullet \boldsymbol{B} = \boldsymbol{c_B}$, $\boldsymbol{Y} = (y_1, y_2 \ldots y_n)$.
Step 2: Choose an entering column. It may be any column $\boldsymbol{a_i}$ of $\boldsymbol{A_N}$ such that $\boldsymbol{Y} \bullet \boldsymbol{a_i}$ is greater than the corresponding component of $\boldsymbol{c_N}$. The current solution is optimal if there is no such column.
Step 3: Solve the system $\boldsymbol{B} \bullet \boldsymbol{V} = \boldsymbol{a_i}$, $\boldsymbol{V} = (v_1, v_2\ldots, v_n)^t$.
Step 4: Find the largest t such that $\boldsymbol{x^*_B} - t\,\boldsymbol{V} \geq 0$. If there is no such t, then the problem is unbounded; otherwise, at least one component of $\boldsymbol{x^*_B} - t\,\boldsymbol{V}$ is equal to 0 and the corresponding variable leaves the basis.
Step 5: Set the value of the entering variable as t and replace the values $\boldsymbol{x^*_B}$ of the basic variables by $\boldsymbol{x^*_B} - t\,\boldsymbol{V}$. Replace the leaving column of $\boldsymbol{B}$ by the entering column and in the basis heading, replace the leaving variable by the entering variable. Then go to step 1.

In the following we solve the systems of equations in steps 1 and 3 and derive explicit formulas for $\boldsymbol{Y}$ and $\boldsymbol{V}$.

4.2 Solve the system $\boldsymbol{Y} \bullet \boldsymbol{B} = \boldsymbol{c_B}$

Let $\boldsymbol{Y} = (y_1\ \ldots, y_{n-1}, \gamma)$. Here $y_1\ldots, y_{n-1}$, γ are called potentials (or dual variables) and $\boldsymbol{Y}$ is called the potential vector. Each y_i, $i = 1, 2\ \ldots, n - 1$ is the potential associated with node i (or the row i) and γ is the potential associated with the last row (delay constraint row) of $\boldsymbol{A}$. Now consider

$$\boldsymbol{Y} \cdot \boldsymbol{B} = \boldsymbol{c_B} \tag{14}$$

This system of equations has n equations in n variables. We get the following from (14).

For each link $e_k = (i, j)$ in G_b, $(y_1\ \ldots, y_{n-1},\gamma)\ h_k = c_{ij}$. That is, $y_i - y_j - \gamma\, d_{ij} = c_{ij}$, if $i \neq n$ or $j \neq n$; $y_i - \gamma\, d_{in} = c_{in}$, if $j = n$ and $-y_j - \gamma\, d_{nj} = c_{nj}$, if $i = n$ (15)

From the above, we can see that we can set the potential of the node n at any constant, for example 0, in all computations that follow.

Definition 2:

1) For link $e_k = (i, j)$, $c(e_k, \gamma) = \gamma\, d_{ij} + c_{ij}$ is called the active cost of link (i, j).

2) $r(i, j) = y_j - y_i + \gamma\, d_{ij} + c_{ij}$ is called the reduced cost of link (i, j).

3) The reduced cost of w is given by $r(w) = \gamma$.

4) The reduced cost of a path p is defined as

$$r(p) = \sum_{(i,j) \in p^+} r(i, j) - \sum_{(i,j) \in p^-} r(i, j)$$ ■

It can be seen from (15) that for any link (i, j) in G_b

$$r(i, j) = y_j - y_i + \gamma\, d_{ij} + c_{ij} = 0. \tag{16}$$

From (16) we also have that for any path p from i to j and any cycle W in G_b

$$r(p) = y_j - y_i + \gamma d(p) + c(p) = 0, \text{ and} \tag{17}$$
$$r(W) = \gamma\, d(W) + c(W) = 0.$$

Lemma 6: If G_b contains a cycle W, then $\gamma = - c(W)/d(W)$; Otherwise, $\gamma = 0$. ■

It can be seen from step 2 of the revised simplex method that a non-basic variable is eligible to enter the basis if its reduced cost is negative. Note that the slack variable is eligible to enter the basis if $\gamma < 0$. Once we have computed the value of γ as in Lemma 6, the other potentials y_i's can be calculated using equation (17) and selecting the path in G_b from node n to node i.

4.3 Solve the system $\boldsymbol{B} \bullet \boldsymbol{V} = \boldsymbol{a_k}$

We show how to solve the system of equations $\boldsymbol{B} \bullet \boldsymbol{V} = \boldsymbol{a_k}$. Consider three cases:

Case 1: G_b contains only n - 1 links, i.e., there is no cycle in G_b and the slack variable w is a basic variable, and some link $e_k = (i, j)$ is the entering variable.

Case 2: The basic variables are associated with n links and the entering variable is $e_k = (i, j)$.

Case 3: The basic variables are associated with n links and the entering variable is the slack variable w.

Solutions in all the three cases are summarized in the following theorem.

Theorem 2. a) If G_b contains no cycle and the entering variable is an in-arc $e_k = (i, j)$, then the vector $\boldsymbol{V}$ defined below is the desired solution to $\boldsymbol{B} \bullet \boldsymbol{V} = \boldsymbol{a}_k$, where W' is the new cycle formed by adding the in-arc e_k and the orientation of W' is chosen to agree with the direction of e_k. The vector $\boldsymbol{V} = (v_1 \dots v_n)^t$ is defined as :

$$v_i = \begin{cases} -1, \textit{if } i < n \textit{ and the link corresponding to the} \\ \quad i \textit{ th column of } B \textit{ is in } W' \textit{ and its orientation} \\ \quad \textit{agrees with the cycle orientation}; \\ 1, \textit{ if } i < n \textit{ and the link corresponding to the} \\ \quad i \textit{ th column of } B \textit{ is in } W' \textit{ and its orientation} \\ \quad \textit{disagrees with the cycle orientation}; \\ d(W'), \textit{if } i = n \\ 0, \textit{otherwise} \end{cases}$$

b) If G_b contains a cycle W and link $e_k = (i, j)$ enters the basis, then $\boldsymbol{V} = -\boldsymbol{V}'_p + (d(W')/d(W)) \bullet \boldsymbol{V}_0$ is the solution of $\boldsymbol{B} \bullet \boldsymbol{V} = \boldsymbol{a}_k$, where $d(W')$ and $d(W)$ are the delays of cycle W' and W, respectively and $\boldsymbol{V}'_p$ and $\boldsymbol{V}_0$ are defined by the cycles W' and W, respectively.

c) If G_b contains a cycle W and the entering variable is the slack variable w, then $\boldsymbol{V} = (1 / d(W))\ \boldsymbol{V}_0$ is the solution to $\boldsymbol{B} \bullet \boldsymbol{V} = \boldsymbol{a}_k$, where $\boldsymbol{V}_0$ is defined by cycle W. ■

5. Initialization

To construct an initial basic feasible solution we first determine a spanning tree containing a feasible s-t path. This can be done by applying Dijkstra's algorithm to compute the shortest path tree with respect to the delay from all nodes to the destination node t. If the resulting s-t path in the tree is infeasible, then no feasible path exists and the algorithm terminates. Without loss of generality we assume that the s-t path is feasible.

Clearly in the basic solution corresponding to the spanning tree selected as above, the flows in all the links in the s-t path in the spanning tree will be equal to one, and flows in all other links will be zero. Since the delay of every link in the TCSP problem is even and the upper bound on path delay is odd we can see that the slack variable w has nonzero value and thus it is in the initial basic feasible solution.

6. Pivot rules and structure of basic solutions for the TCSP problem

In this section we study the structure of the subgraphs of basic solutions generated by the simplex method.

The subgraph G_b of the initial basic feasible solution has (n - 1) links and the n^{th} variable in this basic solution is the slack variable w, and $w > 0$ in the solution. At this initial step, $\gamma = 0$. Define

$$d(G_b) = \sum_{(u,v) \in G_b} x_{uv} d_{uv} .$$

By (7), $d(G_b) = \Delta - w$. Now one of the following two possibilities occurs in the following pivots.

1. The simplex method constructs a new spanning tree solution with the slack variable w remaining nonzero in the new solution.

2. The simplex method constructs a G_b that contains one cycle W (formed by adding the in-arc). If the slack variable w becomes 0 first when we push the flow along the cycle W, then w becomes nonbasic with respect to this solution. If the cycle W does not share links with the s-t path in G_b, then the flows on all the links in W will be zero and thus all the link flows will be either 0 or 1. This would then imply that the delay of the s-t path (being even) will not be equal to the upper bound Δ (being odd), making the value of w nonzero. This is a contradiction. So the only cycle W in G_b must have some common links with the s-t path in G_b. The cycle W cannot be a directed cycle. If it were a directed cycle, then the reduced cost of the entering link will be equal to the sum of the costs of the links in W. This sum is a positive number contradicting the requirement that the reduced cost of the entering link must be negative (step 2 of the revised simplex method). Also, the flow values on all the links in W must be nonzero, for otherwise all the link flows will be either 0 or 1 making w nonzero.

Summarizing, when the first time a G_b with a cycle is encountered it will be necessarily of the form shown in Fig.1. (*b*). Flows on the links in the cycle will be λ or $1 - \lambda$. The simplex method will select the value of $\lambda > 0$ in such a way such that $d(G_b) = \Delta$.

Though the cycle in the G_b encountered the first time after initialization will not be a directed cycle, in a subsequent step a G_b with a directed cycle may be created. To achieve an efficient implementation of the simplex method, we would like to avoid generating any G_b containing a directed cycle. This can be achieved by the pivot rule P1 given next.

Pivot Rule P1**:** The slack variable w assumes the highest priority to be selected as entering variable.

Theorem 3. If the pivot rule P1 is followed and the simplex method on the TCSP is initialized as in section 5, then no basic solution subgraph G_b containing a directed cycle will be created. ■

7. Anti-cycling strategy

A basic solution in which one or more basic variables assume zero values is called degenerate. Simplex pivots that do not alter the basic solution are called degenerate. Furthermore, a basic solution generated at one pivot and reappearing at another will lead to cycling. Since degenerate pivots do not result in any improvement of the solutions, they are also a cause of inefficiency. We present two strategies to handle degeneracy. The first one to be presented in this section is the anti-cycling strategy which is the modification and extension of Cunningham's anti-cycling strategy [18]. The second strategy to be presented in the following section is designed to avoid performing degenerate pivots. In our discussions we shall assume that the node t has been selected as the root node.

Definition 3. Given a feasible basic solution subgraph G_b, we say that the link $(u, v) \in G_b$ is oriented toward (*resp.* away from) the root if any of the paths in G_b from the root to u (*resp.* v) passes through v (*resp.* u). A feasible basic solution G_b with corresponding flow vector $\boldsymbol{x}$ is said to be strongly feasible if every link (u, v) of G_b with $x_{uv} = 0$ is oriented toward the root. ■

If the out-arc (u, v) is not a link of the cycle in the basis solution, then G_b - (u, v) contains exactly two components $G_b(u)$ and $G_b(v)$ such that $u \in G_b(u)$ and $v \in G_b(v)$. If the root is in $G_b(v)$, link (u, v) is oriented toward the root; otherwise it is oriented away from the root. See Fig. 1 for an example of a strongly feasible G_b. Actually, by Lemma 4, all links in G_b are oriented toward t if the basis is strongly feasible.

Lemma 7. For any degenerate pivot, the out-arc is not on the cycle of the current G_b. ■

Theorem 4. If the subgraphs G_b's of feasible basic solutions generated by the simplex method are strongly feasible then the simplex method does not cycle. ■

8. Avoiding degenerate pivots

In this section, we develop a strategy which avoids performing degenerate pivots.

Enhanced Pivot Rule P2: If there is a choice for selecting the entering variables, then select an entering variable in the following order of preference:

a) The slack variable if it is eligible to enter.

b) Eligible links whose tail nodes are on some directed s-t path in the current G_b.

As we discussed in section 6, rule a) above guarantees that every G_b is of one of the two forms in Fig.1. Both these subgraphs of basic solutions are strongly feasible. Consider next rule b). Suppose we can find an in-arc $e = (u, v)$ according to rule b). Let W' denote the new cycle in $G_b + e$ with its orientation defined as the direction of e. It can be seen that the flows on all links in W' whose directions disagree with that of W' are of nonzero and thus we can push positive amount of flow along the cycle until the flow on some links of the s-t path (whose directions disagree with the orientation of W') reach zero. By removing one such link with zero flow, we obtain a new G_b. In fact, we can select the out-arc in such a way that the resulting G_b is also strongly feasible (see next section). This pivot will not lead to degeneracy. On the other hand, if no such link is eligible to enter the basis (note: in this case γ is nonnegative), then we have no option but to perform a degenerate pivot. To avoid performing degenerate pivots we proceed as follows.

Let P be the set of nodes on the s-t paths in the current basic subgraph G_b. Assign costs to links in the network as follows:

Link cost $c(u, v)$ with $u \notin P$ and $v \in P$ is set as $c(e_{uv}, \gamma) + y_v$; Otherwise $c(u, v)$ is set as $c(e_{uv}, \gamma)$.

Now condense all the nodes in P into a single node, say, R. and reverse the directions of all the links. Let the resulting network be called N'. Note that none of the links with both its ends in P will be in N'. Now use Dijkstra's algorithm on N' and obtain the shortest path tree with node R as the start node. The links of G corresponding to the links of the shortest path tree of N' and the links with their both end nodes in P will be a new G'_b (Notice that this operation preserves the strongly feasibility of G_b and will not change the value of γ). Let the shortest distance value of the node u computed by the algorithm be denoted as $d(u)$. Then the potentials of the nodes with respect to G'_b will be:

1. For each node $u \notin P$, $y_u = d(u)$, and
2. For all other nodes (all the nodes in P) the potentials are the same as in the previous G_b.

Now, $\forall\ (u, v)$, $u \notin P$, $y_u = d(u) \leq d(v) + c(e_{uv}, \gamma) = y_v + c(e_{uv}, \gamma)$, which implies that for all such links, $r(u, v) = y_v - y_u + \gamma d_{uv} + c_{uv} \geq 0$ and thus links whose tails are not in P are not eligible for choice as in-arc. Since the above operation does not affect the value of γ, w is not eligible either. Thus we can only consider arcs whose tails are in P (part (b) of enhanced rule P2). If we still cannot find an in-arc according to enhanced rule P2 after the above operation, it implies that we have got the optimal basic solution because no entering variable is available. This procedure is denoted as Modified-Dijkstra and its implementation is much simpler than the description above.

9. Finding a leaving arc (out-arc)

Suppose the current feasible basic solution G_b is strongly feasible and link $e = (u, v)$ is the in-arc. If G_b contains a cycle W, then the flow can be decomposed into exactly two s-t paths. We define the branching point as the first node on W as we traverse the paths from node s

to *t* (see Fig. 1. (b)). In this section, *e* and *e'* always denote the in-arc and out-arc, respectively.

Claim 1: If the current basic solution G_b is strongly feasible and is not optimal, then one of the arcs *e'* incident to the branching node or the tail node of the in-arc *e* is eligible for choice as out-arc and $G_b + e - e'$ is still strongly feasible. ■

We prove the claim by enumerating all possible cases and determining the leaving variable in each case using Theorem 2 and step 4 of the revised simplex method. Let the cycle created by adding the in-arc be denoted by *W'* with its orientation defined as that of the in-arc. When we choose the leaving variable, we always choose it such that the strong feasibility of the basic solution is preserved.

Case 1: Slack variable *w* is in the basic solution (the current G_b is a tree, $\gamma = 0$ and $w > 0$). This corresponds to Theorem 2 (a). According to step 4 of the revised simplex method, we need to consider only the entries of ***V*** that are 1 or $d(W')$ if $d(W') > 0$. Without loss of generality, assume $d(W') > 0$. These entries correspond to the links of *W'* that lie on the *s*-*t* path of the current G_b or the slack variable *w*. The corresponding entries in the current basic solution $\boldsymbol{x^*_B}$ are 1 for the links and its current value for *w*. The minimum value of *t* that satisfies the constraint $\boldsymbol{x^*_B} - t\ V \geq 0$ will be determined by the inequalities $1 - t \geq 0$ and $w - t\ d(W') \geq 0$. Thus the minimum value of *t* will be $\min\{1, w / d(W')\}$. Since $w = \Delta - d(G_b)$ is odd and $d(W')$ is even, $w / d(W') \neq 1$. So, if $w < d(W')$, *w* will leave the basis. Otherwise, the links in *W'* that lie on the *s*-*t* path in the current G_b are eligible to leave the basis. We shall select the unique link *e'* on the *s*-*t* path in G_b that is incident to the tail node of the in-arc. This guarantees that the new G_b, denoted as G'_b is strongly feasible.

Notice that if *w* leaves the basis, $w = 0$ in G'_b. This means that $d(G'_b) = \Delta$. In this case, G'_b contains two *s*-*t* paths p_1 and p_2 with flow λ and $1 - \lambda$, respectively (see Fig. 1).

The value of λ can be calculated from the equation

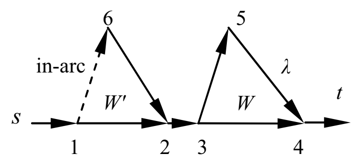
$$\lambda\ d(p_1) + (1 - \lambda)\ d(p_2) = \Delta\ .$$

This line of argument will be used in determining the leaving arc in Case 2 discussed below.

Case 2: The basic solution consists of *n* links, i.e., there is a cycle *W* with branching point *a* in the basic solution. The slack variable *w* is eligible to enter the basis if $\gamma < 0$. Then according to part *a*) of pivot rule P2, we let *w* enter the basis and shall select one of the two links in the current G_b that are incident on the branching point *a* to leave the basis. The choice can be made according to Case 3 in section 4.3.

Suppose $\gamma > 0$. An in-arc will create a new link *W'*. This corresponds to Case 2 in section 4.3. We need to consider four sub-cases that capture all possibilities. Without loss of generality, we assume that the orientation of *W* is clockwise and the orientation of *W'* agrees with the direction of the in-arc.

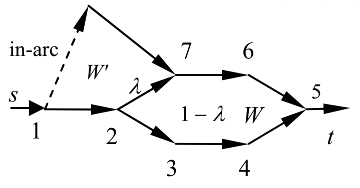
Case 2.1: Possible out-arcs: (1, 2), (3, 5) and (3, 4).



Here, $(x_{12}, x_{35}, x_{34}) = (1, \lambda, 1 - \lambda)$ and thus the out-arc corresponds to the first zero component in the following formula as *t* increases from 0.

$$(1, \lambda, 1 - \lambda) - t\ (1, d(W') / d(W), - d(W') / d(W))$$
$$= (1 - t, \lambda - t\ d(W') / d(W), 1 - \lambda + t\ d(W') / d(W)).$$

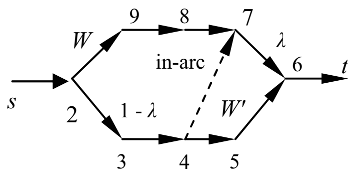
Case 2.2: Possible out-arcs: (1, 2), (2, 7) and (2, 3).



Then the out-arc corresponds to the first component reaching 0 in the following formula as *t* increases.

$$(x_{12}, x_{27}, x_{23})$$
$$- t\ (1, 1 + d(W') / d(W), - d(W') / d(W))$$
$$= (1, \lambda, 1 - \lambda) - t\ (1, 1 + d(W') / d(W), - d(W') / d(W)).$$

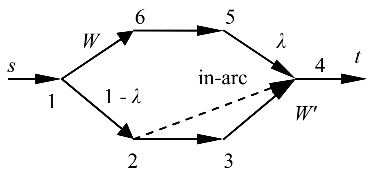
Case 2.3: Possible out-arcs: (2, 3), (2, 9) and (4, 5).



The out-arc corresponds to the first zero component in the following formula when *t* increases.

$$(x_{23}, x_{29}, x_{45})$$
$$- t\ (- d(W') / d(W), d(W') / d(W), 1 - d(W') / d(W)).$$

Case 2.4: Possible out-arcs: (2, 3), (1, 6) and (1, 2)



Out-arc corresponds to the first zero component in the following formula as *t* increases.

$$(x_{23}, x_{16}, x_{12})$$
$$- t\ (1 - d(W') / d(W), d(W') / d(W), - d(W') / d(W)).$$

10. The Network Based Simplex algorithm (NBS): complexity and performance

We first present the complete algorithm (given below) for the Network Based Simplex (NBS) algorithm for the RELAX-TCSP problem. Then we show that our algorithm is of pseudo polynomial time complexity. We also show that we can extract from the optimum solution to the RELAX-TCSP problem a feasible solution to the TCSP (and hence the original CSP) problem and derive bounds on how much the cost of this solution deviates from the cost of the optimum solution to the original CSP problem.

```
Procedure NBS
 Transform the original network as in section 2.
 Find an initial feasible basic solution as in section 5.
 loop {
  if ($\gamma < 0$) then
     Let slack variable $w$ be the entering variable (rule (a) of
         Pivot rule P2)
   else if an in-arc satisfying rule (b) of Pivot Rule P2 is
        available then
          Choose one of them as the entering variable.
      else {
         invoke procedure Modified-Dijkstra on the
               active costs to update the potentials.
         if an in-arc satisfying rule (b) of Pivot Rule P2 is
             available then
               Choose one of them as the entering variable.
         else stop /*has reached the optimal condition*/
      }
  }
  Determine a leaving variable as in section 9.
  Update the flows and the potentials as in section 4.
```

10.1. Complexity analysis

We next present a complexity analysis of the NBS algorithm and establish its pseudo-polynomial time complexity.

Fact 1: If there is no cycle in the basic solution subgraph, then for each link e_{uv}, the associated flow x_{uv} is either 1 or 0. If there is a cycle W in G_b, x_{ij} is either 0 or no less than $1 / |d(W)|$.

Fact 2: If e_{uv} is the in-arc and W' and W are the newly created cycle and the old cycle (if it exists), respectively, we have

$$0 < |y_u - y_v - \gamma\, d_{uv} - c_{uv}| = |\gamma\, d(W') + c(W')| = \begin{cases} |\, c(W'), & \gamma = 0; \\ |\, c(W')d(W) - d(W')c(W)\,| \,/\, d(W), & \gamma \neq 0. \end{cases}$$

Fact 3: Let t be the maximal flow that can be pushed on the new cycle W'. Suppose that e_{uv} and x_{uv} are a link and its flow in the basic solution, respectively. Then t is only constrained by

$x_{uv} - t\,(1 + |d(W') / d(W)|) \geq 0$, $t \geq 0$ and $t \leq 1$ and hence
$max\ t \geq min\{1, 1/(|d(W)| + |d(W')|)\} = 1/(|d(W)| + |d(W')|)$.

Fact 4: Let T and T' be two consecutive feasible basic solutions in the simplex method and $c(T)$ denote the cost of the flow associated with the basic solution T. If $c(T') < c(T)$ and D is the maximal link delay, then $|c(T') - c(T)| = t\,|y_u - y_v - \gamma\, d_{uv} - c_{uv}| \geq 1/(2n^2D^2)$.

Theorem 5. NBS algorithm terminates within $2n^3D^2C$ pivots, where n is the number of nodes and D (*resp.* C) is the maximal link delay (*resp.* cost). ■

10.2. An approximate solution to the TCSP / CSP problem and performance bounds

If the optimal basic solution subgraph for the RELAX-TCSP problem contains no cycle, then clearly the *s-t* path in this subgraph is also the optimum solution to the original CSP problem. On the other hand, if the optimal basic solution graph contains a cycle, then the optimum flow can be decomposed into flows along two directed *s-t* paths p_1 and p_2 with positive flow along each path.

Lemma 8. If $c(p_2) \leq c(p_1)$, then $c(p_2) \leq c(p^*) \leq c(p_1)$ and $d(p_2) \geq \Delta \geq d(p_1)$, where p^* is the optimal path of the original CSP problem and Δ is the delay bound. ■

It follows from the above lemma that the path p_1 is a feasible solution to the CSP problem. We may use this as an approximate solution to the original CSP problem. We next evaluate the quality of this approximate solution.

Theorem 7. Let p_1 and p_2 be the two paths derived from the optimal solution to the RELAX-TCSP problem with $c(p_1) \geq c(p_2)$, then

$$\frac{c(p_1)}{c(p^*)} \leq 1 + \frac{1-\lambda}{\lambda}\left(1 - \frac{c(p_2)}{c(p_1)}\right), \text{ and}$$

$$\frac{d(p_2)}{\Delta} \leq 1 + \frac{\lambda}{1-\lambda}\left(1 - \frac{d(p_1)}{\Delta}\right),$$

where λ is the flow on path p_1 at termination and Δ is the delay bound. ■

Using a special example below, we can show that no constant factor approximation solution based on relaxation approach (including our approach and LARAC) is possible. However, numerical simulations show that the approximate solution is very close to the optimum. For closing the gap between the optimum value and the approximate value see [19].

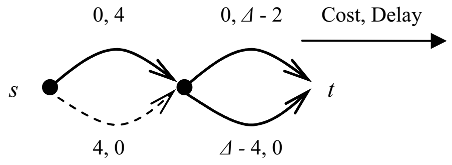


Fig. 2. Counter example

Let OPT, OPT_S and Δ denote the optimal cost, the cost of the path returned by relaxation method and the delay upper bound. In Fig.2, the solid links correspond to the basic variables in the optimal basis. Thus $OPT_S = \Delta - 4$.

Obviously, $OPT = 4$. So $|OPT_S - OPT| / OPT = (\Delta - 8) / 4$, where Δ can be specified arbitrarily.

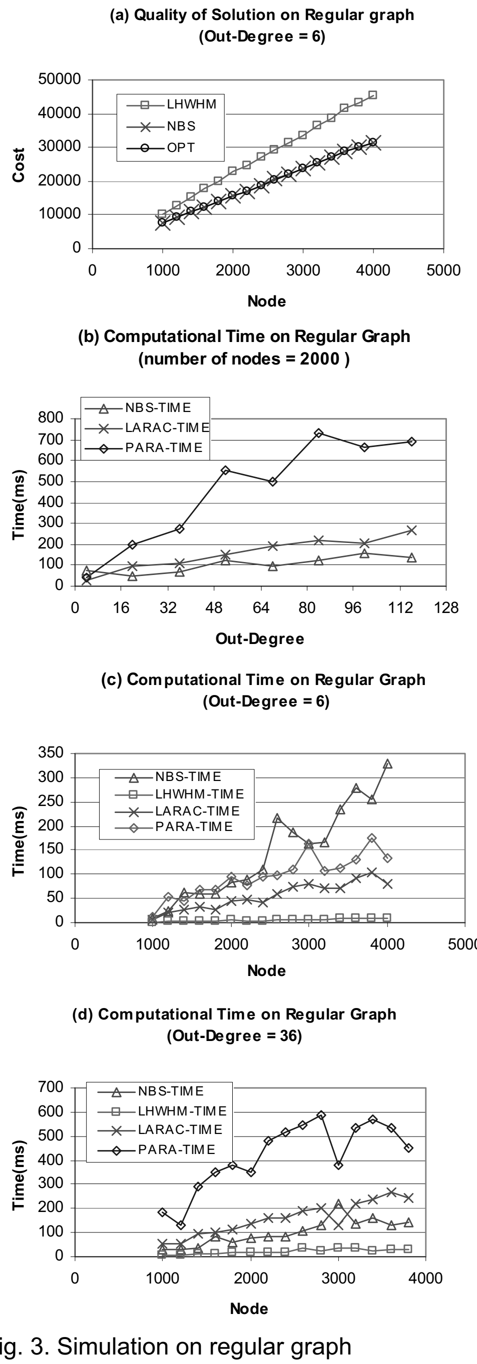


Fig. 3. Simulation on regular graph

11. Simulation and comparative evaluation

We compare NBS algorithm with the LARAC and LHWHM algorithms. We also compare the NBS algorithm with a strongly polynomial algorithm based on parametric search [19], denoted as PARA.

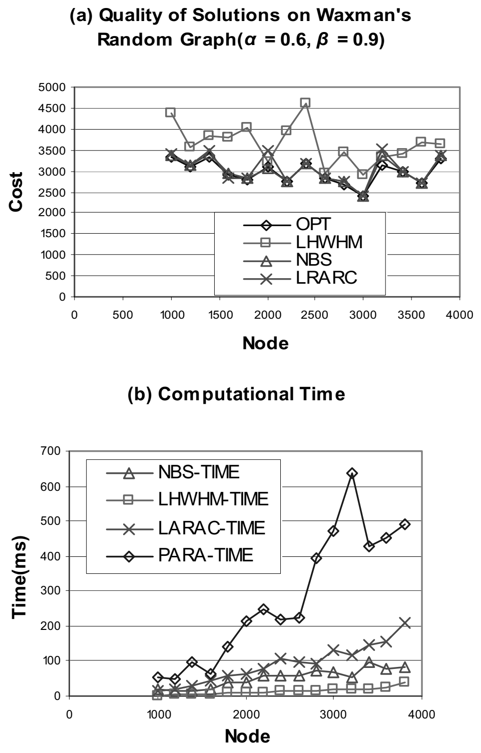


Fig. 4. Waxman's random graph

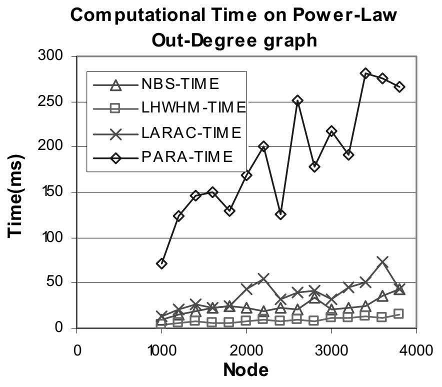


Fig.5. Power-Law Out-Degree graph.

We use three classes of network topologies: regular graphs $H_{k,n}$ (see [17]), Power-Law Out-Degree graph [20] and Waxman's random graph [21]. For a network $G(V, E)$, the nodes are labeled as 1, …, $n = |V|$. Nodes $n / 2$ and n are chosen as the source and target nodes. For the Power-Law Out-Degree graph and Waxman's random graph, the hop number of feasible s-t paths is usually very small even when the network is very large. This will bias the results in favor of the LHWHM algorithm. So, for Waxman's random graphs, a link joining node u and v is added if $|u - v| < |V| / 50$ besides other rules generating the random graph. We keep the original version of Power-Law Out-Degree graph as in [20]. Even though this kind of graphs favors the LHWHM algorithm, the comparison of the performance of the LARAC and NBS algorithm is still an indicator of the merits of NBS. The link costs and delays are randomly independently generated even integers in the range from 1 to 200. The delay bound is 1.2 times the delay of the minimum delay *s-t* paths in *G*. The results are shown in Fig. 3 - 5. Experiments show that NBS algorithm may find better solution than the LARAC algorithm by selecting the best feasible path encountered during the execution instead of the optimum path to the RELAX-TCSP problem. We also find that for sparse graphs, NBS takes more time than the LARAC algorithm. However, when the network is dense (larger out-degree), NBS beats LARAC (see Fig. 3. (*b*)).

12. Summary

In this paper, we studied the QoS routing problem (or equivalently the CSP problem) from the primal perspective in contrast to most of the currently available approaches that studied the problem from a dual perspective. Specifically we applied the revised simplex method on the primal form of the RELAX-TCSP problem. Several strategies are employed to achieve efficient implementation of the revised simplex method. We show that our algorithm is of pseudo-polynomial time complexity. We have also shown how to extract an approximate solution to the original CSP problem from the optimum solution to the RELAX-TCSP problem and derive bounds on the quality of this solution with respect to the optimum solution. Extensive simulation results are presented to demonstrate that our approach compares favorably with the LARAC and is faster on dense graphs.

Acknowledgment: The works of K. Thulasiraman and Guoliang Xue have been supported by NSF ITR grants ANI-0312435 and ANI-0312635.

References

[1] Z. Wang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications," *IEEE JSACs*, vol.14, no.7, pp.1228-1234, Sept. 1996.

[2] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math. of Oper. Res.*, 17(1), 1992, pp.36-42.

[3] C. Phillips, "The network inhibition problem," *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, May, 1993

[4] D. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest paths problem," *Oper. Res. Letters*, vol. 28, pp. 213-219.

[5] G. Luo, K. Huang, J. Wang, C. Hobbs and E. Munter, "Multi-QoS constraints based routing for IP and ATM networks," in *Proceedings of IEEE Workshop on QoS Support for Real-Time Internet Applications,* Vancouver Canada, June 1, 1999

[6] R. Ravindran, K. Thulasiraman, A. Das, K. Huang, G. Luo and G. Xue, "Quality of services routing: heuristics and approximation schemes with a comparative evaluation," *ISCAS*, 2002.

[7] G. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks* 10, 293-310, 1980

[8] K. Mehlhorn and M. Ziegelmann, "Resource constrained shortest path," *Proc.* 8*th European Symposium on Algorithms* (*ESA*2000)

[9] A. Jüttner, B. Szviatovszki, I. Mécs and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," *IEEE INFOCOM*-2001, pp. 859-868.

[10] D. Blokh and G. Gutin, "An approximation algorithm for combinatorial optimization problems with two parameters," *Australasian Journal of Combinatorics*, vol. 14, 1996, pp.157-164.

[11] Y. Xiao, K. Thulasiraman and G. Xue, "Equivalence, unification and generality of two approaches to the constrained shortest path problem with extension," Allerton Conference on Control, Communication and Computing, University of Illinois, Urbana-Champaign, Oct. 1- 3, 2003

[12] A. Jüttner, "On resource constrained optimization problems," submitted for publication.

[13] M. Ziegelmann, "Constrained shortest paths and related problems," PhD thesis, Max-Planck-Institut für Informatik, 2001.

[14] S. Chen and K. Nahrstedt, "An overview of Quality-of-Service routing for the next generation high-speed networks: problems and solutions," *IEEE Network Magazine*, 12(6), Nov. / Dec, 1998.

[15] A. Orda and A. Sprintson, "Efficient algorithms for computing disjoint QoS paths," *IEEE INFOCOM*, 2004.

[16] Y. Xiao, K. Thulasiraman and G. Xue, "Disjoint QoS paths selection for protection against failures: a network programming based approach," Allerton Conference on Control, Communication and Computing, University of Illinois, Urbana-Champaign, Oct. 2004.

[17] K. Thulasiraman and M. N. Swamy, *Graphs*: *Theory and algorithms*, Wiley Interscience, New York, 1992.

[18] V. Chvάtal, *Linear programming*, W. H. Freeman and Company, New York (1983)

[19] Y. Xiao, K. Thulasiraman, G. Xue and A. Jüttner, "The constrained shortest path problem: algorithmic approaches and an algebraic study with generalization," submitted for publication.

[20] C. R. Palmer and J. G. Steffan, "Generating network topologies that obey power laws," *IEEE GLOBECOM*, 2000.

[21] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Selected Areas in Comm.* 6(9), Dec. 1988