QoS Routing in Communication Networks: Approximation Algorithms Based on the Primal Simplex Method of Linear Programming

Ying Xiao, *Member, IEEE,* Krishnaiyan Thulasiraman, *Fellow, IEEE,* and Guoliang Xue, *Senior Member, IEEE*

***Abstract*— Given a directed network with two integer weights, cost and delay, associated with each link, Quality-of-Service (QoS) routing requires the determination of a minimum cost path from one node to another node such that the delay of the path is bounded by a specified integer value. This problem also known as the constrained shortest path problem (CSP) admits an Integer Linear Programming (ILP) formulation. Due to the integrality constraints, the problem is NP-hard. So, approximation algorithms have been presented in the literature. Among these, the LARAC algorithm, based on the dual of the LP relaxation of the CSP problem is very efficient. In contrast to most of the currently available approaches we study this problem from a primal perspective. Several issues relating to efficient implementations of our approach are discussed. We present two algorithms of pseudo-polynomial-time complexity. One of these allows degenerate pivots and uses an anti-cycling strategy and the other called the NBS algorithm is based on a novel strategy which avoids degenerate pivots. Experimental results comparing the NBS algorithm, the LARAC algorithm, and general purpose LP solvers are presented. In all cases the NBS algorithm compares favorably with others and beats them on dense networks.**

***Index Terms*— Constrained shortest path, linear programming, simplex method, graph algorithms, communication networks, routing protocols, QoS routing.**

I. Introduction

ROUTING is a fundamental problem in communication networks. In traditional data networks, routing is achieved by best effort routing. Best effort routing is primarily concerned with providing connectivity. FIFO provides best-effort service. Here, flows are not differentiated and are serviced on a first-come, first-served basis. In best effort routing the routing protocol usually characterizes the network with a single metric such as hop-count or delay and uses a shortest path algorithm for path computation. Whereas the best-effort routing paradigm is adequate to serve the needs for traditional applications such as FTP (File Transfer Protocol) it is quite inadequate in providing the stringent Quality of Service (QoS) guarantees demanded by popular multimedia applications such as real time digital video or audio transmission. To support a broad range of QoS requirements, routing protocols need to consider more complex models that incorporate multiple metrics such as cost, delay, delay variation, loss probability, and bandwidth. This has triggered efforts towards proposals for QoS based frameworks such as DiffServe and IntServ, QoS routing protocols that accommodate multiple QoS requirements such as Q-OSPF and PNNI, and QoS routing algorithms (See [1], [9], [10], [31]). Despite these efforts, there is no standardized QoS routing protocol for the Internet. To the best of our knowledge the only standardized QoS routing protocol is ATM PNNI [1].

Two activities are involved in routing: i) Capturing the network state information and disseminating the information throughout the network. This requires detection of significant changes, topology updates, distributed broadcasting (flooding) of the information to each node in the network etc. (ii) Routing algorithms that compute the paths that satisfy certain performance guarantees.

In this paper we are concerned with the latter, namely, QoS routing algorithms. QoS measures can be classified into two types of metrics, non-additive (also called bottleneck, e.g., bandwidth) and additive constraints. Each measure is modeled by associating a weight with each link. For a non-additive measure QoS weight of a path is the minimum weight along the path. In the case of additive measures such as cost, delay, reliability and delay-jitter the QoS weight of a path is the sum of the QoS weights of the links on the path. Non-additive measures can be handled easily by simply removing from the network the links that do not satisfy the required QoS measure.

In this paper we are concerned with finding paths that satisfy additive QoS metrics. In particular, we are interested in the QoS routing problem that requires the determination of a minimum cost path from a source node to a destination node in a 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 NP-hard [33]. Thus, there has been a good deal of efforts in developing efficient approximation algorithms and heuristics.

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 but are usually very slow in practice. References [12], [21], [29] and the references therein contain most of the current literature on approximation algorithms for the CSP problem. As regards heuristics, the LHWHM algorithm [22] is a simple heuristic which is very fast (requiring only one or two invocations of Dijkstra's shortest path algorithm) and produces

The work of K. Thulasiraman has been supported by NSF ITR grant ANI-0312435. The work of G. Xue has been supported by NSF ITR grant ANI-0312635.

Ying Xiao and Krishnaiyan Thulasiraman are with University of Oklahoma, Norman, OK 73019, USA (e-mail: eagle7827@gmail.com, thulasi@ou.edu). Guoliang Xue is with Arizonia State University, Tempe, AZ 85287, USA. (e-mail: xue@asu.edu)

solutions which are usually found to be of acceptable quality in practice. Reference [30] 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 [11] by Handler and Zang. This is based on a geometric approach (what is also called the hull approach [23]). More recently, in an independent work, Jüttner et al. [16] developed the LARAC algorithm which also solves the dual of the CSP problem using Lagrangian relaxation method. In contrast to the geometric method, they used an algebraic approach. In [40] Xue developed an algorithm that is similar to the LARAC algorithm. In [5] Blokh and Gutin 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 [35], [39], Xiao et al. drew attention to the fact that the algorithms in [11] and [16] are equivalent. In view of this equivalence, we shall refer to these algorithms simply as the LARAC algorithm. In [23], Mehlhorn and Ziegelmann have provided several insights on the QoS routing problem. In [15] Jüttner established the strong polynomiality of the LARAC algorithm. Ziegelmann [43] provides a fairly complete list of references to the literature on the CSP problem.

Another problem, Multi-Constrained Path (MCP) problem has also been a topic of extensive study. In this problem, each link is associated with $l > 1$ additive weights. The MCP problem is to find an s-t path that satisfies all the l constraints. Key results and algorithms on the MCP problem may be found in [7], [13], [14], [17]–[20], [24]–[26], [41], [42]. Recent works on the QoS routing problem may be found in [3], [6], [27], [37], [38].

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. This is an extended and detailed version of our work in [36] and includes proofs of all results and more extensive experimental results. The rest of the paper is organized as follows. In Section II, 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 [2] 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 in Section III of an equivalent problem on a transformed network, called the TCSP problem. Section IV deals with the structure of the basic solutions of the RELAX-TCSP problem, the relaxed form of the TCSP problem. Section V discusses the revised simplex method of linear programming, its application on RELAX-TCSP, and several strategies to achieve an efficient implementation. This results in an algorithm that allows degenerate pivots and uses an anti-cycling strategy developed in Section V-F. Another algorithm called NBS algorithm presented in Section VI avoids degenerate pivots completely. Both these algorithms are of pseudo polynomial time complexity. In Section VI-C.2, we show 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. In Section VII, experimental results comparing the NBS algorithm with the LARAC algorithm [16], the LHWHM algorithm [22], and the general purpose LP solvers are presented. Section VIII concludes with a summary of the main contributions. To conserve space proofs of a few results are omitted.

II. The CSP Problem: LP Formulation and the LARAC Algorithm

In this section, we first define the Constrained Shortest Path (CSP) problem and present an ILP formulation. Due to integrality constraints in the ILP formulation the problem is NP-hard. Relaxing the integrality constraints results in RELAX-CSP. We then present the LARAC algorithm of [16] which solves the dual of RELAX-CSP.

Definition 1: 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 $c_{uv} > 0$ (representing cost, the expense imposed by using or installing the link) and $d_{uv} > 0$ (transmission delay along the link). 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 of p.

Notice that our assumption that link weights are integers does not involve any loss of generality because, in digital systems, all numbers are represented discretely and can be scaled and rounded to integers. In order to simplify our presentation, we assume all the values to be integers. We also assume that only links impose costs and delays. If the nodes impose costs and delays, we can use the node splitting technique to transform node costs and delays into link costs and delays (See Chapter 2.4 of [2]).

We use the terms "link" and "arc" interchangeably. Without loss of generality, we assume that for every node i, there is a directed path from i to the destination node t. In the rest of the paper, $m = |E|$ and $n = |V|$.

A path is called a directed path (cycle) if there are no backward links in the path (cycle). Given two nodes s, t and an integer $\Delta > 0$, a directed s-t path p is said to be feasible if $d(p) \leq \Delta$. In the rest of the paper, a directed s-t path will be referred to simply as an s-t path.

CSP(Constrained Shortest Path) problem: Find an s-t path p_{opt} = arg $\min\{c(p)|p$ is a feasible s-t path$\}$. This is illustrated with the example in Fig. 1.

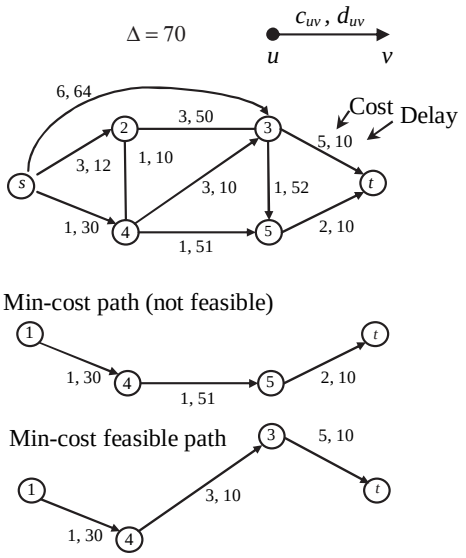


Fig. 1. An example of CSP problem.

The CSP problem can be formulated as an integer linear programming problem as below.

CSP:

$$\text{Minimize} \sum_{(u,v)\in E} c_{uv}x_{uv} \tag{1}$$

subject to

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

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

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

In (3), w is the slack variable for the delay constraint.

The main difficulty with the CSP problem lies with the integrality condition that requires that the variables x_{uv} be 0 or 1. Removing or relaxing this requirement from the above integer linear program leads to RELAX-CSP, the relaxed CSP problem.

RELAX-CSP:

$$\text{Minimize} \sum_{(u,v)\in E} c_{uv}x_{uv} \tag{5}$$

subject to

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

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

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

We will show later that by using a transformation and applying certain pivot rules we can enforce $x_{uv} \leq 1$ (the discussion after Theorem 3, Section V-E).

Dual Based Approach: LARAC Algorithm [16]

The dual of the CSP problem involves s-t paths and a variable $\lambda \geq 0$. For each link (u, v), let the aggregated cost c_λ be defined as $c_{uv} + \lambda d_{uv}$. For a given λ, let $c_\lambda(p) = c(p) + \lambda d(p)$ denote the aggregated cost of the path p. Finally define $L(\lambda)$ as:

$$L(\lambda) = \min\{c_\lambda(p) | p \text{ is an } s\text{-}t \text{ path}\} - \lambda\Delta. \tag{9}$$

Note that in the above, $\min\{c_\lambda(p)|p$ is an s-t path$\}$ can be easily obtained by applying Dijkstra's algorithm using aggregated link costs $c_{uv} + \lambda d_{uv}$. Let the s-t path which has minimum aggregated cost with respect to a given λ be denoted as p_λ. Then $L(\lambda) = c_\lambda(p_\lambda) - \lambda\Delta$ and the dual of the RELAX-CSP can be presented as follows.

DUAL-RELAX-CSP: Determine $\max\{L(\lambda)|\lambda \geq 0\}$.

The value of λ that achieves the maximum $L(\lambda)$ in DUAL-RELAX-CSP will be denoted by λ^*. Note that L^*, the optimum value of DUAL-RELAX-CSP is a lower bound on the optimum cost of the path that solves the corresponding CSP problem [16]. From the optimum solution to the RELAX-CSP problem we can extract an approximate solution to the original CSP problem. The key issue in solving DUAL-RELAX-CSP is how to search for the optimal λ. The LARAC algorithm of [16] presented in Algorithm 1 is one such efficient search procedure. In this algorithm Dijkstra(s, t, c), Dijkstra(s, t, d), and Dijkstra(s, t, c_λ) denote, respectively, Dijkstra's shortest path algorithm using link costs, link delays, and aggregated link costs with respect to the multiplier λ.

Algorithm 1 LARAC(s, t, Δ) algorithm

```
{Compute the minimum cost s-t path}
p_c ← Dijkstra(s, t, c)
if (d(p_c) ≤ Δ) then return p_c
{Compute the minimum delay s-t path}
p_d ← Dijkstra(s, t, d)
if (d(p_d) > Δ) then return "no solution"
loop
   λ ← (c(p_c) − c(p_d))/(d(p_d) − d(p_c))
   {Compute minimum c_λ cost s-t path}
   r ← Dijkstra(s, t, c_λ)
   if (c_λ(r) = c_λ(p_c)) then return p_d
   else if (d(r) ≤ Δ) then p_d ← r else p_c ← r
end loop
```

In [39] we have studied several aspects of the dual based approach such as optimality conditions and other approaches such as parametric search and binary search.

III. A Transformed Problem and Basic Concepts

In contrast to most other approaches in the literature, we study the CSP problem using the primal simplex algorithm. In order to achieve an efficient implementation of the approach we transform the problem to an equivalent one on a transformed network defined below.

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

The transformed problem will be referred to as the TCSP problem.

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

In view of the above result, we consider in the rest of the paper only the relaxed form of the TCSP problem, namely, RELAX-TCSP (same as RELAX-CSP except that the network is the transformed one as defined above). 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.

In the rest of the section we shall define certain terminology leading to a matrix representation of RELAX-TCSP. Let the links be labeled as $e_1, e_2 \ldots e_m$ and the nodes be labeled as $1, 2 \ldots, n$. 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 [8], [32]. 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 H corresponding to link e_k will be denoted by the vector $\boldsymbol{h}_k$. For $e_k = (i, j)$, we have $\boldsymbol{h}_k = (h_{1,k} \ldots, h_{i,k} \ldots, h_{j,k} \ldots, h_{n-1,k})^t$ with all its components being 0 except for $h_{i,k} = 1$ and $h_{j,k} = -1$. Let

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{H} & \boldsymbol{0} \\ \boldsymbol{D} & -1 \end{pmatrix} = (\boldsymbol{a}_1, \boldsymbol{a}_2 \ldots, \boldsymbol{a}_m, \boldsymbol{a}_{m+1}), \tag{10}$$

$$\boldsymbol{D} = (-d_1, -d_2 \ldots, -d_m), \tag{11}$$

$$\boldsymbol{a}_i = \begin{pmatrix} \boldsymbol{h}_i \\ -d_i \end{pmatrix}, i \leq m, \text{ and} \tag{12}$$

$$\boldsymbol{a}_{m+1} = \begin{pmatrix} \boldsymbol{0} \\ -1 \end{pmatrix}. \tag{13}$$

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

$$\textbf{RELAX-TCSP: Minimize } \boldsymbol{cx}$$
$$\text{subject to } \boldsymbol{Ax} = \boldsymbol{b}. \tag{14}$$

In (14), $\boldsymbol{x} \geq 0$ and $\boldsymbol{b} = (b_1 \ldots, b_{n-1}, -\Delta)^t$ with $b_s = 1, b_t = -1$, and $b_i = 0$ for $i \neq s, t$.

The rest of the paper deals with the primal simplex based solution of RELAX-TCSP.

IV. Simplex Method: 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. Note that there is no link associated with the slack variable. The subgraph G_b will be called the subgraph of the basic solution or simply the basis graph. 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.

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

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

Then, $\boldsymbol{HU}(W) = 0$ [32].

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.

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

Proof: Let

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{E}_{n-1,n} \\ \boldsymbol{D}_{1,n} \end{pmatrix}$$

be a basis matrix (submatrix of $\boldsymbol{A}$), where $\boldsymbol{H}_{n-1,n}$ is a $(n-1) \times n$ submatrix of $\boldsymbol{H}$ and $\boldsymbol{D}_{1,n}$ is the vector of n components (corresponding to the basic variables) of the last row of $\boldsymbol{A}$. Then $\boldsymbol{E}_{n-1,n}\boldsymbol{U}(W) = 0$ by Lemma 1. On the other hand, $\boldsymbol{D}_{1,n}\boldsymbol{U}(W) = -d(W)$.

Since rank ($\boldsymbol{B}$) = n, we have

$$\boldsymbol{BU}(W) = \begin{pmatrix} \boldsymbol{E}_{n-1,n}\boldsymbol{U}(W) \\ -d(W) \end{pmatrix} = \begin{pmatrix} 0 \\ -d(W) \end{pmatrix} \neq 0.$$

Thus the lemma follows. ■

Lemma 4: If the basis subgraph G_b contains no cycle that is not a directed cycle, there are exactly two s-t paths in G_b.

Thus it follows from the above lemma that the transformation we introduced guarantees that the structure of the basis subgraph will be one of the three forms shown in Fig. 2 (a spanning tree or a spanning tree plus an extra link). In a later section we shall introduce a pivot rule which will ensure that the basis subgraph will not contain any directed cycle, thereby eliminating the structure in Fig. 2.(c).

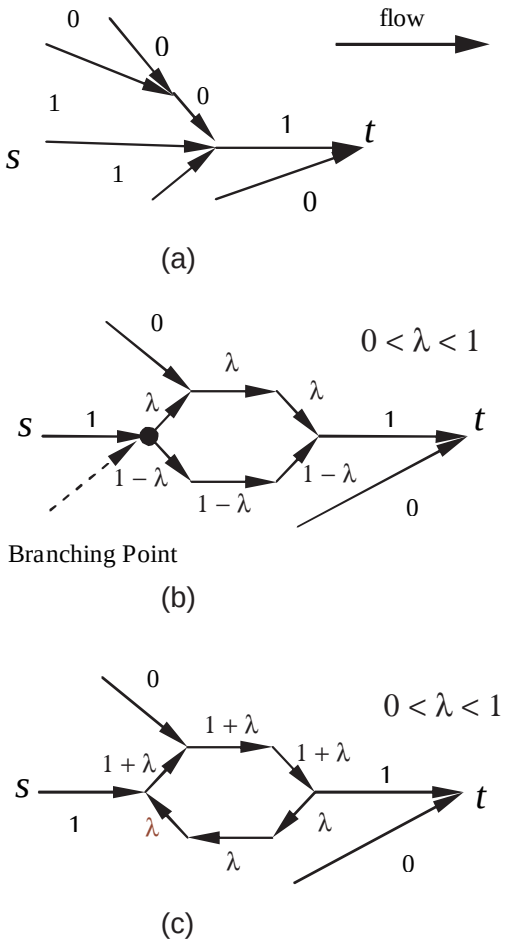


Fig. 2. Structure of basis graph: (a) tree basic solution, (b) basic solution with a cycle(not directed), and (c) basic solution with a directed cycle.

V. 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 that is described in detail in [8]. We then derive formulas required to identify the entering and the leaving variables.

A. Revised simplex method

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

$$\text{Minimize } \boldsymbol{cx}$$
$$\text{Subject to } \boldsymbol{Ax} = \boldsymbol{b}, \boldsymbol{x} \geq 0.$$

Here $\boldsymbol{A}$ is an $n \times (m+1)$ matrix with rank ($\boldsymbol{A}$) = n, $\boldsymbol{x} = (x_1 \ldots, x_{m+1})^t$, $\boldsymbol{c} = (c_1 \ldots, c_{m+1})$, and $\boldsymbol{b} = (b_1 \ldots, b_n)^t$. 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 [8]

1) Step 1: Solve the system $\boldsymbol{YB} = \boldsymbol{c}_B$, where $\boldsymbol{Y} = (y_1, y_2 \ldots, y_n)$.
2) Step 2: Choose an entering column. It may be any column $\boldsymbol{a}_i$ of $\boldsymbol{A}_N$ such that $\boldsymbol{Ya}_i$ is greater than the corresponding component of $\boldsymbol{c}_N$. The current solution is optimal if there is no such column.
3) Step 3: Solve the system $\boldsymbol{BV} = \boldsymbol{a}_i$, where $\boldsymbol{V} = (v_1, v_2 \ldots, v_n)^t$.
4) 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.
5) 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.

B. 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 Δ on path delay is odd, the slack variable $w > 0$ and so it is in the initial basic feasible solution.

In Sections V-C and V-D we solve the systems of equations in Steps 1 and 3 and derive explicit formulas for Y and V. These results are from [37] and are repeated here for the sake of completeness. If a link flow variable is chosen as the entering variable then the corresponding link is called the in-arc. Out-arcs are similarly defined.

C. Solving the System $\boldsymbol{YB} = \boldsymbol{c}_B$.

Let $\boldsymbol{Y} = (y_1 \ldots, y_{n-1}, \gamma)$. Here $y_1 \ldots, y_{n-1}, \gamma$ 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

$$\boldsymbol{YB} = \boldsymbol{c}_B \tag{15}$$

This system of equations has n equations in n variables. We get the following from (15).

For each link $e_k = (i,j)$ in $G_b, (y_1 \ldots, y_{n-1}, \gamma)h_k = c_{ij}$. That is,

$$\begin{aligned} &y_i - y_j - \gamma d_{ij} = c_{ij}, \text{ if } i \neq n \text{ and } j \neq n, \\ &y_i - \gamma d_{in} = c_{in}, \text{ if } j = n, \text{ and} \\ &-y_j - \gamma d_{nj} = c_{nj}, \text{ if } i = n. \end{aligned} \tag{16}$$

From the above, we can see that we can set the potential of node n at any constant. In all computations that follow, we shall set the potential of node n equal to zero.

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 $r(w) = \gamma$, and

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 (16) that for any link (i,j) in G_b

$$r(i,j) = y_j - y_i + \gamma d_{ij} + c_{ij} = 0. \tag{17}$$

From (17) we also have that for any path p from i to j and any cycle W in G_b

$$\begin{aligned} &r(p) = y_j - y_i + \gamma d(p) + c(p) = 0, \\ &r(W) = \gamma d(W) + c(W) = 0. \end{aligned} \tag{18}$$

Lemma 5: If G_b contains a cycle W, then $\gamma = -c(W)/d(W)$; Otherwise, $\gamma = 0$.

Proof: If there is no cycle in G_b then the slack variable w is a basic variable and the corresponding column $[0, 0 \ldots, 0, -1]^t$ will be a column of $\boldsymbol{B}$. Since the cost of the slack variable is zero, we get from (15) that $\gamma = 0$. Suppose that G_b contains a cycle W. By (18), we get $\gamma d(W) + c(W) = 0$. By Lemma 3, $d(W) \neq 0$. So $\gamma = -c(W)/d(W)$. ■

Lemma 6: A link is eligible to enter the basis if its reduced cost is negative and the slack variable is eligible to enter the basis if $\gamma < 0$.

Proof: The proof follows from Step 2 of the revised simplex method. ■

Once we have computed the value of γ as in Lemma 5, the other potentials y_i's can be calculated using equation (18) and selecting the path in G_b from node n to node i. Summarizing the above, we have the following procedure for solving $\boldsymbol{Y}\boldsymbol{B} = \boldsymbol{c}_B$ and calculating the potentials.

(1) Set the potential of node n to zero.

(2) Compute γ as in Lemma 5.

(3) For each node i, let p_i be a simple path in G_b from node n to node i. If there are two paths in G_b due to the cycle, we will get the same results no matter which path is selected.

(4) Set $c(p) = \sum_{(u,v)\in p^+} c_{uv} - \sum_{(u,v)\in p^-} c_{uv}$ and $d(p) = \sum_{(u,v)\in p^+} d_{uv} - \sum_{(u,v)\in p^-} d_{uv}$, where p_i^+ and p_i^- are the sets of forward and backward links on p_i, respectively, as we traverse the path from node n to node i.

Once the potentials are determined, an entering variable, if it exists, can be selected as in Step 2 of the revised simplex method.

D. Solving the System $\boldsymbol{B}\boldsymbol{V} = \boldsymbol{a}_k$

We next show how to solve the system of equations $\boldsymbol{B}\boldsymbol{V} = \boldsymbol{a}_k$. We consider three cases:

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

Case b): G_b contains a cycle (that is, G_b has n links) and the entering variable is $e_k = (i,j)$.

Case c): G_b contains a cycle (G_b has n links) and the entering variable is the slack variable.

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 $\boldsymbol{V} = (v_1 \ldots, v_n)^t$ defined below is the desired solution to $\boldsymbol{B}\boldsymbol{V} = \boldsymbol{a}_k$, where W' is the new cycle formed by adding the in-arc e_k and the orientation of W' is chosen to be the same as the direction of e_k.

$$v_i = \begin{cases} -1, \text{for } i < n \text{ and the link corresponding to} \\ \quad \text{the } i\text{th column of } B \text{ is on } W' \text{ and its} \\ \quad \text{orientation agrees with the cycle orientation} \\ 1, \text{ for } i < n \text{ and the link corresponding to} \\ \quad \text{the } i\text{th column of } B \text{ is on } W' \text{ and its} \\ \quad \text{orientation disagrees with the cycle orientation} \\ d(W'), \text{ for } i = n \\ 0, \text{ otherwise} \end{cases} \tag{19}$$

b) If G_b contains a cycle W and the entering variable is a link $e_k = (i,j)$, then $\boldsymbol{V} = -\boldsymbol{V}_p' + \frac{d(W')}{d(W)}\boldsymbol{V}_0$, is the solution to $\boldsymbol{B}\boldsymbol{V} = \boldsymbol{a}_k$, where $d(W')$ and $d(W)$ are the delays of cycles W' and W, respectively and $\boldsymbol{V}_p'$ and $\boldsymbol{V}_0$ are defined by the cycles W' and W, respectively (See Lemma 1).

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

Proof: **Case a)**: 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 the link $e_k = (i,j)$ is the entering variable. In this case,

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{H}_{n-1,n-1} & \boldsymbol{0} \\ \boldsymbol{D}_{1,n-1} & -1 \end{pmatrix},$$

where $\boldsymbol{H}_{n-1,n-1}$ is associated with the $(n-1)$ links in G_b and $n-1$ nodes and $\boldsymbol{D}_{1,n-1}$ is the vector of $(n-1)$ components (corresponding to the basic variables except for w) of the last row of matrix $\boldsymbol{A}$.

Let W' denote the new cycle formed by adding the in-arc $e_k = (i,j)$ and let the orientation of W' be chosen to be the same as the orientation of the in-arc. By Lemma 1, it is easy to verify that the vector $\boldsymbol{V} = (v_1 \ldots, v_n)^t$ defined as in the theorem solves the system $\boldsymbol{B}\boldsymbol{V} = \boldsymbol{a}_k$.

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

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{H}_{n-1,n} \\ \boldsymbol{D}_{1,n} \end{pmatrix},$$

where $\boldsymbol{H}_{n-1,n}$ is associated with the n links and $n-1$ nodes and $\boldsymbol{D}_{1,n}$ is the vector of the n components of the last row

of $\boldsymbol{A}$ corresponding to these n links, and

$$\boldsymbol{a}_k = \begin{pmatrix} \boldsymbol{h}_k \\ -d_{ij} \end{pmatrix}.$$

We need to solve the system of equations

$$\begin{pmatrix} \boldsymbol{H}_{n-1,n} \\ \boldsymbol{D}_{1,n} \end{pmatrix} \boldsymbol{V} = \begin{pmatrix} \boldsymbol{h}_k \\ -d_{ij} \end{pmatrix}. \tag{20}$$

First, let us consider

$$\boldsymbol{H}_{n-1,n}\boldsymbol{V} = \boldsymbol{h}_k. \tag{21}$$

Because there are n links in G_b, there is exactly one cycle, denoted by W. Therefore according to Lemma 1,

$$\exists \boldsymbol{V}_0, \boldsymbol{H}_{n-1,n}\boldsymbol{V}_0 = 0. \tag{22}$$

After adding link $e_k = (i, j)$, we get a new cycle W' and let us choose the orientation of this cycle to be the same as that of e_k. Then by Lemma 1,

$$\exists \boldsymbol{V}' = \begin{pmatrix} \boldsymbol{V}'_p \\ 1 \end{pmatrix}, \ (\boldsymbol{H}_{n-1,n}, \boldsymbol{h}_k) \begin{pmatrix} \boldsymbol{V}'_p \\ 1 \end{pmatrix} = 0. \tag{23}$$

So,

$$\boldsymbol{H}_{n-1,n}(-\boldsymbol{V}'_p) = \boldsymbol{h}_k. \tag{24}$$

Because rank($\boldsymbol{H}_{n-1,n}$) = $n-1$, $-\boldsymbol{V}'_p + u\boldsymbol{V}_0$, $u \in R$ is the solution space of (21). We can compute u as follows.

$$\boldsymbol{D}_{1,n}(-\boldsymbol{V}'_p + u\boldsymbol{V}_0) = -d_{ij}. \tag{25}$$

Since $\boldsymbol{D}_{1,n}\boldsymbol{V}_0 = -d(W)$ and $\boldsymbol{D}_{1,n}(-\boldsymbol{V}'_p) + d_{ij} = d(W')$, we get from (25)

$$d(W') - ud(W) = 0 \text{ and hence } u = d(W')/d(W).$$

Therefore we have proven that

$$\boldsymbol{V} = -\boldsymbol{V}'_p + \frac{d(W')}{d(W)}\boldsymbol{V}_0 \tag{26}$$

is the desired solution to $\boldsymbol{BV} = \boldsymbol{a}_k$.

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

Following the arguments in Case (b), we can show that $\boldsymbol{V} = \frac{\boldsymbol{V}_0}{d(W)}$ is the solution to $\boldsymbol{BV} = \boldsymbol{a}_k$. Here $\boldsymbol{V}_0$ is defined by the cycle W in G_b. ■

E. A Pivot Rule and Structure of Basic Feasible Solutions

In this subsection we present a pivot rule and study the structure of 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 nth variable in this basic solution is the slack variable $w > 0$. At this initial step, $\gamma = 0$ (Lemma 5). 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 next pivot.

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) and w becomes non-basic with respect to this solution. 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). By Lemma 4, there will be exactly two s-t paths in G_b. 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 and hence basic.

Summarizing, when the first time a G_b with a cycle is encountered, it will be necessarily of the form shown in Fig. 2.(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 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: Select the slack variable w as the entering variable if it is eligible to enter.

Theorem 3: If the pivot rule P1 is followed and the simplex method on the RELAX-TCSP problem is initialized as in Section V-B, then no basic solution subgraph G_b will contain a directed cycle.

Proof: Assume that a G_b with a directed cycle W' is created and let $e_{ij} = (i, j)$ be the in-arc with which this cycle is created.

Suppose $W' = e_{ij}e_{jj_1}e_{j_1j_2}\ldots, e_{j_ki}$ and p_{ji} is the directed path from j to i in W'.

Since e_{ij} is an in-arc and $\boldsymbol{Y} = (y_1, y_2 \ldots, y_{n-1}, \gamma)$ is the potential vector, we have $r(i, j) = y_j - y_i + \gamma d_{ij} + c_{ij} < 0$ and $r(p_{ji}) = y_i - y_j + \gamma d(p_{ji}) + c(p_{ji}) = 0$.

Summing the above, we obtain $\gamma d(W') + c(W') < 0$.

Since $d(W') > 0$ and $c(W') > 0$, $\gamma < 0$. This implies that the slack variable is eligible to enter the basis but was not selected. This is a contradiction. ■

Theorem 3 implies that pivot rule P1 along with the transformation introduced in Section III guarantees that G_b will take only the structures shown in Fig. 2.(a) and Fig. 2.(b). Under these conditions we are also guaranteed that the values of the variables x_{uv} will be restricted to the range $0 \le x_{uv} \le 1$.

F. An Anti-Cycling Strategy

A basic solution in which one or more basic variables assume zero values is called degenerate. A degenerate basic solution may result in a pivot that does not alter the basic solution. Such pivots 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 subsection is the anti-cycling strategy which is a variation and extension of Cunningham's anti-cycling strategy in [2], [4], [8]. The second strategy to be presented in Section VI is designed to avoid degenerate pivots completely.

Definition 3: Given a feasible basic solution subgraph G_b and a node called the root, we say that the link $(u,v) \in G_b$ is oriented toward (resp. away from) the root if any path 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 x is 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 basic 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. 2.(a), (b) for examples of a strongly feasible G_b. We shall select node t as the root node.

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

Proof: A degenerate pivot does not alter the basic solution. This means that each variable has the same value in the current basic solution as well as in the basic solution resulting from the degenerate pivot. The flow on each link in a cycle is non-zero. If a link on a cycle were to leave the basis, then after the degenerate pivot it would become non-basic with zero flow. But that would contradict that the current pivot is degenerate. ■

If the out-arc is not on the cycle in the current G_b, then the potentials can be updated easily as described next (See Chapter 5.1.2 of [4]). Let $\boldsymbol{T}$ be the current G_b and $e = (u,v)$ and $e' = (u',v')$ be the out-arc and the in-arc, respectively. Let $\boldsymbol{T}' = \boldsymbol{T} - e + e'$ be the subgraph of the new basic variables. If e is not on the cycle in the current G_b, the new potential vector $\boldsymbol{Y}'$ associated with $\boldsymbol{T}'$ can be obtained as follows (notice that γ does not change in this case):

$$y'_u = \begin{cases} y_u + r_{u'v'}, & \text{for } u \in T_{u'} \\ y_u, & \text{for } u \in T_{v'}. \end{cases} \tag{27}$$

where $r_{u'v'} = c(e_{u'v'}, \gamma) + y_{v'} - y_{u'}$ and $\boldsymbol{T}_{u'}(\boldsymbol{T}_{v'})$ is the component of $\boldsymbol{T} - e$ containing $u'(v')$.

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.

Proof: First observe that in any sequence of degenerate pivots, the value of the slack variable will remain the same. So the leaving and entering variables can only be the links in the network. Let G_b be a feasible basic solution subgraph and t be the root. We define two unique values for $G_b : C(G_b) = \sum_{(u,v)\in E} c_{uv}x_{uv}$ and $W(G_b) = \sum_{u\in V}(y_t - y_u)$.

Notice that for a given G_b, the value of $W(G_b)$ is unique even though the values of the potentials $\boldsymbol{Y}$ may not be unique. Consider two consecutive basic solutions $G_b^i = G_b$ and $G_b^{i+1} = G_b^i + e - f$, where e and f are the in-arc and out-arc, respectively. We first show that either $C(G_b^{i+1}) < C(G_b^i)$ or $W(G_b^{i+1}) > W(G_b^i)$.

Indeed if the pivot that generates G_b^{i+1} from G_b^i is non-degenerate, then $C(G_b^{i+1}) < C(G_b^i)$. If it is degenerate, we have $C(G_b^{i+1}) = C(G_b^i)$. In this case we shall prove $W(G_b^{i+1}) > W(G_b^i)$.

Here the in-arc $e = (u,v)$ still has zero flow in G_b^{i+1}. By Lemma 7, f is not a link on the cycle in G_b^i, so the value of γ does not change. Because G_b^{i+1} is strongly feasible, in G_b^{i+1}, link e must be oriented toward the root node t, which implies that node t belongs to $G_b(v)$ (the component of $G_b^i - f$ containing v). Now we can obtain the potentials using equation (27).

Since $r_{uv} = c(e_{uv},\gamma) + y_v - y_u < 0$, $W(G_b^{i+1}) = W(G_b^i) - |G_b(u)| r_{uv} > W(G_b^i)$.

If the simplex method cycles, then for some $i < j$, $G_b^i = G_b^j$. This means $G_b^i = G_b^{i+1} \cdots = G_b^j$. But then $W(G_b^i) > W(G_b^{i+1}) > \cdots > W(G_b^j) = W(G_b^i)$ contradicting that $W(G_b^i) = W(G_b^j)$. ■

VI. A Strategy for Avoiding Degenerate Pivots and the Network Simplex Based (NBS) Algorithm

In this section we first present in Section VI-A a strategy for avoiding degenerate pivots. We then show in Section VI-B how to select a leaving variable. In Section VI-C we present a complete description of the new Network Based Simplex (NBS) algorithm and its complexity analysis. We also show how to extract an approximate solution to the TCSP (hence the original CSP) problem and establish the performance bounds on the approximate solution.

A. Avoiding Degenerate Pivots

In this section we shall develop a strategy which avoids performing degenerate pivots which is based on the following pivot rule.

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 the directed s-t path(s) in the current G_b.

As we discussed in Section V, rule a) above guarantees that every G_b is of one of the two forms in Fig. 2.(a), (b). 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 nonzero and thus we can push positive amount of flow along the cycle until the flows 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 subsection). 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 path(s) in the current basis subgraph G_b. Assign costs to links in the network as follows: Link cost c_{uv} with $u \notin P$ and $v \in P$ is set as $c(e_{uv},\gamma) + y_v > 0$; Otherwise c_{uv} is set as $c(e_{uv},\gamma)$(See Fig. 3).

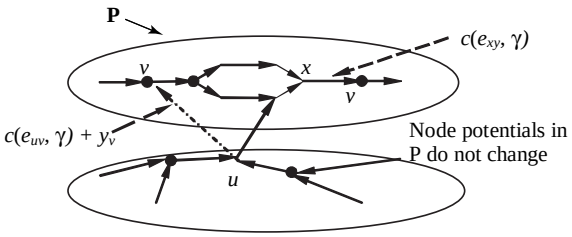


Fig. 3. Link costs for network N'.

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 basis subgraph 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 $d(u)$. Then we set the potentials of the nodes with respect to G'_b as fellows: For each node $u \notin P$, $y_u = d(u)$ and 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 those 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 since no entering variable is available.

We will show in the following section how to choose a leaving variable using Theorem 2.

B. 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. 2.(b)). In this subsection, 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.

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 $\boldsymbol{V}$ 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 satisfying the constraint $\boldsymbol{x}^*_B - t\boldsymbol{V} \geq 0$ will be determined by the inequalities $1 - t \geq 0$ and $w - td(W') \geq 0$. Thus the maximum 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 some 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.

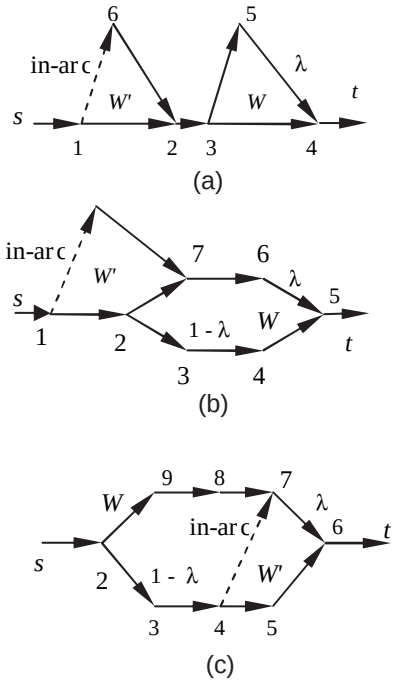


Fig. 4. Find leaving variable: (a) sub-cases 2.1, (b) sub-case 2.2, and (c) sub-case 2.3.

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. 2).

The value of λ can be calculated from the equation: $\lambda d(p_1) + (1 - \lambda) d(p_2) = \Delta$.

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 using Theorem 2 (c) of Section V-D. Suppose $\gamma > 0$. An in-arc will create a new cycle W'. This corresponds to Theorem 2(b). We need to consider three 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 (Fig. 4.(a)): 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.

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

Case 2.2 (Fig. 4.(b)): Possible out-arcs: (1, 2), (2, 7), and (2, 3). Link (7, 6) is not eligible for out-arc for otherwise $w \neq 0$ in the next basic solution due to the property of the transformed network. The out-arc is decided by the following formula as in Case 2.1.

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

Case 2.3 (Fig. 4.(c)): 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.

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

C. NBS Algorithm, Complexity Analysis, and an Approximate Solution

We now present a complete description of the Network Based Simplex (NBS) algorithm that uses the strategies developed in Section VI-A and VI-B for the RELAX-TCSP problem. We show in Section VI-C.1 that the algorithm is of pseudo-polynomial time complexity. In Section VI-C.2 we show how to extract from an optimum solution to the RELAX-TCSP problem a feasible solution to the TCSP problem and hence to the original CSP problem and derive bounds on the deviation of this solution from the cost of the optimum solution.

1) Complexity Analysis:

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 0 or at least $1/|d(W)|$.

Proof: If there is no cycle, the proof is trivial. Assume there is a cycle W. It can be seen that the flow on links not on the two paths are 0 and the flows on the paths but not on the cycle is 1. Since there is a cycle, the flow can be decomposed into two paths p_1 and p_2. Consider flows on the cycle W. Suppose the flow on p_1 and p_2 are λ and $1-\lambda$ with $0 < \lambda < 1$.

Assume $d(p_1) \geq d(p_2)$. Since $d(p_1)$ and $d(p_2)$ are both even and Δ is odd, $d(p_1) \neq \Delta$ and $d(p_2) \neq \Delta$. Also by Lemma 3, $d(W) \neq 0$. So $d(p_1) \neq d(p_2)$ because $d(W) = d(p_1) - d(p_2)$.

We also have $\lambda d(p_1) + (1-\lambda)d(p_2) = \lambda(d(p_1) - d(p_2)) + d(p_2) = \Delta$. So,

$$\min\{d(p_1), d(p_2)\} \leq \Delta \leq \max\{d(p_1), d(p_2)\} \text{ and} \\ \lambda = (\Delta - d(p_2))/(d(p_1) - d(p_2)).$$

Hence $\lambda \geq 1/d(W)$ because $\Delta - d(p_2) \geq 1$ and $d(W) = d(p_1) - d(p_2) > 0$.

Similarly, we can prove that $1 - \lambda \geq 1/d(W)$. ■

Algorithm 2 NBS

```
Transform the original network as in Section III
Find an initial feasible basic solution as in Section V-B
loop
  if γ < 0 then
    Let slack variable w be the entering variable (rule (a)
    of Pivot rule P2 in Section VI-A)
  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 Dijkstra's algorithm on the active costs of N'
    to update the potentials (See Section VI-A).
    if an in-arc satisfying rule (b) of Pivot Rule P2 is
    available then
      Choose one of them as the entering variable
    else
      Stop
    end if
  end if
end loop
Determine a leaving variable as in Section VI-B
Update the flows and the potentials as in Section V-C
```

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

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

Proof: Suppose the cycle W' is $e_1 e_2 \ldots e_k$ where $e_1 = e_{uv}$. Since all the links but e_{uv} on W' are in the basic solution, the reduced costs on all these links but e_{uv} are 0. So $|y_u - y_v - \gamma d_{uv} - c_{uv}| = |\gamma d(W') + c(W')|$. Recalling that $\gamma = -c(W)/d(W)$, if there exists a cycle W in the basic solution or $\gamma = 0$ if no such cycle exists, we get the rightmost equality.

Since e_{uv} is an in-arc, $|y_u - y_v - \gamma d_{uv} - c_{uv}| > 0$. ■

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 the strictest constraint on t is given by $x_{uv} - t(1 + |d(W')/d(W)|) \geq 0$, $t \geq 0$ and $t \leq 1$. Hence

$$\max t \geq \min\{1, \frac{1}{|d(W)| + |d(W')|}\} = \frac{1}{|d(W)| + |d(W')|}.$$

Proof: First assume there is a cycle W in the current basic solution. If we push flow t on the new cycle W', according to Theorem 2 and Step 4 of the revised simplex method, in the worse case, the flow on all links will be decreased by at most $t(1 + |d(W')/d(W)|)$. Proof follows if we recall that $x_{uv} \geq 1/d(W)$. The proof is similar if there is no cycle in the basic solution. ■

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

Proof: Follows from $|c(\boldsymbol{T}')-c(\boldsymbol{T})| = t|y_u - y_v - \gamma d_{uv} - c_{uv}|$ and Facts 2 and 3. ■

Theorem 5: NBS algorithm terminates within $2n^3D^2C$ pivots, where $n = |\boldsymbol{V}|$ and D (resp. C) is the maximum link delay (resp. cost) and hence its time complexity is pseudo-polynomial.

Proof: Let $\boldsymbol{T}_0, \boldsymbol{T}_1 \ldots \boldsymbol{T}_l$ be the sequence of consecutive feasible basic solutions. It suffices to show that $l \leq 2(nD)^3$. According to Fact 4, $c(\boldsymbol{T}_0)-c(\boldsymbol{T}_l) \geq l/(2(nD)^2)$ and $c(\boldsymbol{T}_0) \leq nC$.

This implies that $l \leq 2n^3D^2C$. Since each pivot requires $O(m)$ operations, the NBS algorithm is of pseudo-polynomial time complexity. ■

Using similar arguments, the revised simplex method that allows degenerate pivots but only uses the anti-cycling strategy of Section V-F can also be shown to be of pseudo-polynomial time complexity.

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 $c(p2) \leq c(p_1)$, then either $c(p_2) \leq c(p^*) \leq c(p_1)$ and $d(p_2) \geq \Delta \geq d(p_1)$, where p^* is the optimal path of the original CSP problem or one of the two paths p_1 and p_2 is optimal.

Proof: Let $0 < \lambda < 1$ and $1-\lambda$ be the flows on p_1 and p_2, respectively. We have

$$\lambda d(p_1) + (1-\lambda)d(p_2) = \Delta, \tag{28}$$
$$\lambda c(p_1) + (1-\lambda)c(p_2) \leq c(p^*). \tag{29}$$

It follows from (28) that

$$\min\{d(p_1), d(p_2)\} \leq \Delta \leq \max\{d(p_1), d(p_2)\}. \tag{30}$$

By (29), $c(p_1)$ and $c(p_2)$ cannot both be greater than $c(p^*)$. So $c(p_2) \leq c(p^*)$.

If $c(p_2) = c(p^*)$ then by (29), $c(p_1) \leq c(p^*)$ which implies p_1 or p_2 is an optimal solution.

Assume $c(p_2) < c(p^*)$. Now $\min\{d(p_1), d(p_2)\} = d(p_1) \leq \Delta$, for otherwise p_2 will be a feasible solution to the CSP problem with cost smaller than $c(p^*)$.

So we have the required inequality $d(p_2) \geq \Delta \geq d(p_1)$.

Also path p_1 is feasible for the original CSP problem by Theorem 1. So $c(p_1) \geq c(p^*)$.

Thus we have the required inequality $c(p_2) \leq c(p^*) \leq c(p_1)$. ■

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

Theorem 6: 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

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

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

Proof: From $\lambda c(p_1) + (1-\lambda)c(p_2) \leq c(p^*)$, we obtain

$$\frac{c(p_1)}{c(p^*)} \leq \frac{c(p^*) - (1-\lambda)c(p_2)}{\lambda c(p^*)} = \frac{1}{\lambda} - \frac{1-\lambda}{\lambda}\frac{c(p_2)}{c(p^*)}.$$

Because $c(p^*) \leq c(p_1)$,

$$\frac{1}{\lambda} - \frac{1-\lambda}{\lambda}\frac{c(p_2)}{c(p^*)} \leq \frac{1}{\lambda} - \frac{1-\lambda}{\lambda}\frac{c(p_2)}{c(p_1)} = 1 + \frac{1-\lambda}{\lambda}(1 - \frac{c(p_2)}{c(p_1)}).$$

Similarly, we can prove that

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

■

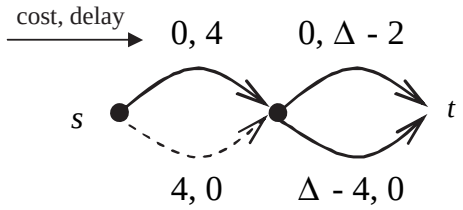


Fig. 5. An example demonstrating that the gap can be arbitrarily large.

Using a special example below, we can show that no constant factor approximation solution based on relaxation approach (including NBS and LARAC algorithm) is possible (However, 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 [39].

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. 5, the solid links correspond to the basic variables in the optimal basis. Thus $OPT_S = \Delta - 4$. Since $OPT = 4$, $|OPT_S - OPT|/OPT = (\Delta - 8)/4$, where Δ can be specified arbitrarily.

VII. Simulation and Comparative Performance Evaluation

We compared our NBS algorithm with the general purpose LP solvers, LARAC algorithm [16], parametric search based LARAC algorithm [39] (denoted as PARA), and the LHWHM algorithm [22]. The LARAC algorithm has time complexity of $O(m^2 \log^4 m)$ [15] while the parametric search based LARAC algorithm has better complexity, namely, $O((m + n \log n)^2)$ [39]. However, the complexity results are derived using the worst scenario and thus they may not be an accurate indicator of the performance of algorithms on average basis. So we compared the four methods using simulations.

We use three classes of network topologies: regular graphs $H_{k,n}$ (see [32]), Power-Law Out-Degree graph [28], and

Waxman's random graph [34]. For a network $G(V, E)$, the nodes are labeled as $1, 2 \ldots, n = |V|$. 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 for generating random graphs. We keep the original version of Power-Law Out-Degree graph as in [28]. Even though this kind of graphs favors the LHWHM algorithm, the comparison of the performance of the LARAC and NBS algorithms 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. 6-9. Experiments show that NBS algorithm can usually find better solutions 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 (Fig. 6.(c)), NBS takes more time than the LARAC algorithm. However, when the network is dense (large out-degree, See Fig. 6.(d)), NBS beats LARAC. Basically, NBS algorithm is a neighbor search algorithm in which a better solution is derived from the current solution. At each pivot, the NBS algorithm tries all the nodes in the s-t path in the current basic graph in order to find an in-arc emanating from a node in the path. When the graph is dense, it is more likely that an eligible in-arc can be found in fewer tries. On the other hand, the LARAC algorithm invokes a series of Dijkstra's shortest path algorithm. When the graph is denser, each step in Dijkstra's algorithm takes more time since Dijkstra's algorithm checks all the neighbors of the currently processed node.

We also compared the NBS algorithm with general purpose LP solvers: CPLEX 8.0 (www.ilog.com/products/cplex), QSopt (www2.isye.gatech.edu/~wcook/qsopt), and CLP (www.coin-or.org). Among all the three solvers, CPLEX is always the fastest (this is not surprising because CPLEX is recognized as one of the best LP solvers). So we only report the experiments with CPLEX. In our experiments with CPLEX, we have used the same graphs as above. Using CPLEX package, we may choose different optimizers such as the primal dual method, network simplex etc. Our experiments show that the CPLEX using the primal dual uses the least time and so our comparison is with respect to this optimizer. Notice that CPLEX can also retrieve the network structure underlying the CSP problem. But we found that this does not help decrease the running time. Actually, it takes longer time to find the optimal solution if CPLEX is directed to use the special structure of the networks. The numerical simulation results in Fig. 9 shows that the NBS algorithm is much faster.

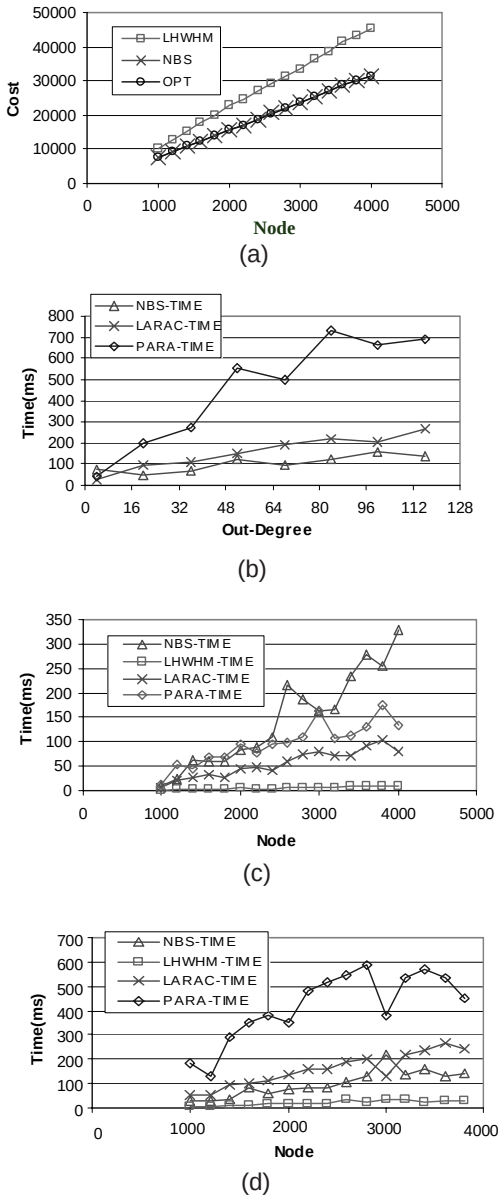


Fig. 6. Simulation on regular graph: (a) quality of solutions on regular graph with out-degree = 6, (b) computational time on regular graph with number of nodes = 2000, (c) computational time on regular graph with out-degree = 6, and (d) computational time on regular graph with out-degree = 36.

VIII. 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. These strategies include: explicit formulas to solve the systems of equations needed to find entering and leaving variables, an anti-cycling strategy, and a strategy to avoid degenerate pivots. These result in two algorithms. One of these allows degenerate pivots and uses an anti-cycling strategy developed in this paper. The other algorithm called NBS algorithm avoids degenerate pivots. We show that both algorithms are of pseudo-polynomial-time complexity. We have also shown how to extract an approximate solution to the original CSP problem

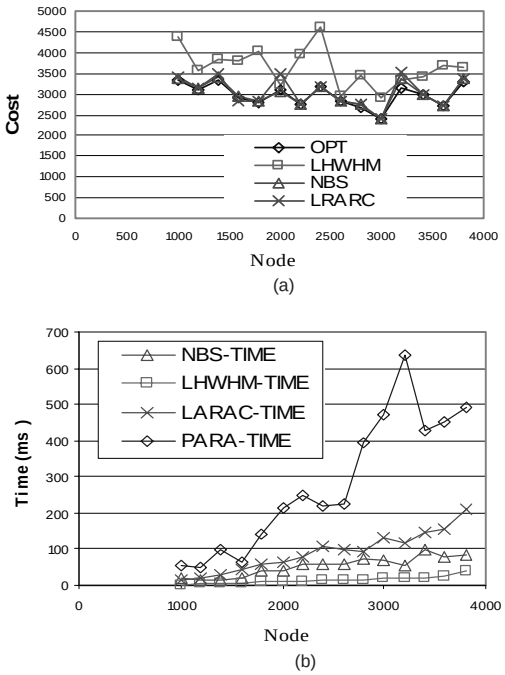


Fig. 7. Waxman's random graph: (a) quality of solutions on Waxman's random graph with $\alpha = 0.6$ and $\beta = 0.9$ and (b) computational time.

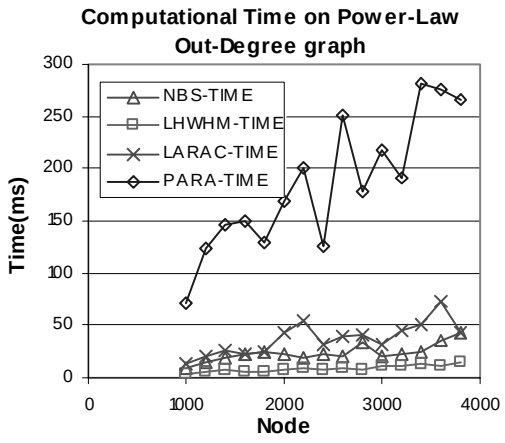


Fig. 8. Power-law out-degree graph.

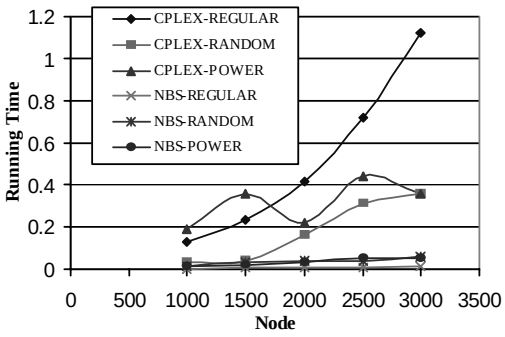


Fig. 9. NBS and CPLEX comparison.

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 algorithm and is faster on dense graphs. Also, our algorithm is faster than the general purpose LP solvers.

Besides providing insights into the structure of solutions produced, our approach based on the primal simplex offers a framework for studying other classes of problems such as the disjoint QoS paths selection problem and the QoS routing problem with multiple constraints. In [37] we have reported our results on the disjoint QoS paths selection problem. In the case of multiple constraints the structure of basic solutions may contain up to l cycles for a problem with l additive constraints. To apply our approach to the case involving multiple constraints, we need to develop efficient methods to solve the two systems of equations studied in Section V. Our approach in combination with the approach developed in [24] is expected to lead to further advances in this area.

REFERENCES

[1] Private network-network interface specification version 1.0 (PNNI 1.0). Technical report, ATM Forum Technical Committee, March 1996.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Networks Flows*. Prentice-Hall, NJ, USA, 1993.

[3] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi, and A. Sprintson. Algorithms for computing QoS paths with restoration. *IEEE Trans. of Networking*, 13(3):648–661, 2005.

[4] D. P. Bertsekas. *Network optimization: continuous and discrete models*. Athena Scientific, Belmont, Massachusetts, USA, 1998.

[5] B. Blokh and G. Gutin. An approximation algorithm for combinatorial optimization problems with two parameters. *Australasian Journal of Combinatorics*, 14:157–164, 1996.

[6] A. Chakrabarti and G. Manimaran. Reliability constrained routing in QoS networks. *IEEE Trans. of Networking*, 13(3):662–675, 2005.

[7] S. Chen and K. Nahrstedt. On finding multi-constrained path. In *ICC*, pages 874–879, 1998.

[8] V. Chvátal. *Linear Programming*. W. H. Freeman, New York, 1983.

[9] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A framework for QoS based routing in the internet. RFC 2386, Internet Engineering Task Force, November 1997. fttp://fttp.ietf.org/internet-drafts/draft-ietf-qosr-framework-02.txt.

[10] R. Guerin, S. Kamat, A. Orda, and T. Przygienda. QoS routing mechanisms and OSPF extensions. RFC 2676, Internet Engineering Task Force, March 1997.

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

[12] R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. of Oper. Res.*, 17(1):36–42, 1992.

[13] A. Iwata, R. Izmailov, B. Sengupta D.-S. Lee, G. Ramamurthy, and H. Suzuki. ATM routing algorithms with multiple QoS requirements for multimedia internetworking. *IEICE Trans. Commun.*, 8:999–1006, 1996.

[14] J. M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[15] Alpár Jüttner. On resource constrained optimization problems. in review, 2003.

[16] Alpár Jüttner, Balázs Szviatovszki, Ildikó Mécs, and Zsolt Rajkó. Lagrange relaxation based method for the QoS routing problem. In *INFOCOM*, pages 859–868, 2001.

[17] Turgay Korkmaz and Marwan Krunz. Multi-constrained optimal path selection. In *INFOCOM*, pages 834–843, 2001.

[18] Turgay Korkmaz and Marwan Krunz. A randomized algorithm for finding a path subject to multiple QoS requirements. *Computer Networks*, 36(2/3):251–268, 2001.

[19] F. A. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem. An overview of constraint-based path selection algorithms for QoS routing. *IEEE Commun. Mag.*, 40:50–55, 2002.

[20] Gang Liu and K. G. Ramakrishnan. A*prune: An algorithm for finding k shortest paths subject to multiple constraints. In *INFOCOM*, pages 743–749, 2001.

[21] D. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest paths problem. *Oper. Res. Letters*, 28:213–219, 2001.

[22] G. Luo, K. Huang, J. Wang, C. Hobbs, and E. Munter. Multi-QoS constraints based routing for ip and ATM networks. In *in Proc. IEEE Workshop on QoS Support for Real-Time Internet Applications*, 1999.

[23] Kurt Mehlhorn and Mark Ziegelmann. Resource constrained shortest paths. In *ESA*, pages 326–337, 2000.

[24] Piet Van Mieghem and Fernando A. Kuipers. Concepts of exact QoS routing algorithms. *IEEE/ACM Trans. Netw.*, 12(5):851–864, 2004.

[25] Piet Van Mieghem, Hans De Neve, and Fernando A. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37(3/4):407–423, 2001.

[26] H. De Neve and P. Van Mieghem. Tamcra: A tunable accuracy multiple constraints routing algorithm. *Comput. Commun.*, 23:667–679, 2000.

[27] Ariel Orda and Alexander Sprintson. Efficient algorithms for computing disjoint QoS paths. In *INFOCOM*, pages 727–738, 2004.

[28] C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *IEEE GLOBECOM*, pages 434–438, 2000.

[29] Cynthia A. Phillips. The network inhibition problem. In *STOC*, pages 776–785, 1993.

[30] 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. In *ISCAS*, pages 775–778, 2002.

[31] S. Shenkar, C. Patridge, and R. Guering. Specification of guaranteed quality of service. RFC 2212, Internet Engineering Task Force, September 1997.

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

[33] Zheng Wang and Jon Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996.

[34] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Commun.*, 6(9):1617–1622, Dec. 1988.

[35] Y. Xiao, K. Thulasiraman, and G. Xue. Equivalence, unification and generality of two approaches to the constrained shortest path problem with extension. In *Allerton Conference on Control, Communication and Computing, University of Illinois*, pages 905–914, 2003.

[36] Y. Xiao, K. Thulasiraman, and G. Xue. The primal simplex approach to the QoS routing problem. In *QSHINE*, pages 120–129, 2004.

[37] Y. Xiao, K. Thulasiraman, and G. Xue. Constrained shortest link-disjoint paths selection: A network programming based approach. Accepted by IEEE Trans. on Circuits and Systems, 2005.

[38] Y. Xiao, K. Thulasiraman, and G. Xue. GEN-LARAC: A generalized approach to the constrained shortest path problem under multiple additive constraints. In *ISAAC*, pages 92–105, 2005.

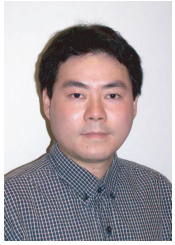
[39] Y. Xiao, K. Thulasiraman, G. Xue, and A. Jüttner. The constrained shortest path problem: algorithmic approaches and an algebra study with generalization. *AKCE J. Graphs. Combin.*, 2(2):63–86, 2005.

[40] G. Xue. Minimum-cost QoS multicast and unicast routing in communication networks. *IEEE Trans. On Commun.*, 51:817–827, 2003.

[41] G. Xue, A. Sen, and R. Banka. Routing with many additive QoS constraints. In *ICC*, pages 223–227, 2003.

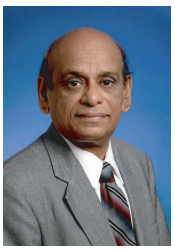
[42] Xin Yuan. Heuristic algorithms for multiconstrained quality-of-service routing. *IEEE/ACM Trans. Netw.*, 10(2):244–256, 2002.

[43] M. Ziegelmann. *Constrained shortest paths and related problems*. PhD thesis, Max-Planck-Institut fr Informatik, 2001.



Ying Xiao received the B.S. degree in computer science from the Nanjing University of Information Science and Technology (formerly Nanjing Institute of Meteorology), Nanjing, China, in 1998, M.S. degree in computer application from the Southwest Jiaotong University, Chengdu, China, in 2001, and Ph.D degree in computer science at the University of Oklahoma, Norman, in 2005.

His research interests include graph theory, combinatorial optimization, distributed systems and networks, statistical inference, and machine learning.



Krishnaiyan Thulasiraman received the Bachelor's degree (1963) and Master's degree (1965) in electrical engineering from the university of Madras, India, and the Ph.D degree (1968) in electrical engineering from IIT, Madras, India. He holds the Hitachi Chair and is Professor in the School of Computer Science at the University of Oklahoma, Norman, where he has been since 1994. Prior to joining the University of Oklahoma, Thulasiraman was professor (1981-1994) and chair (1993-1994) of the ECE Department in Concordia University, Montreal. He was on the faculty in the EE and CS departments of the IITM during 1965-1981.

Dr. Thulasiraman's research interests have been in graph theory, combinatorial optimization, algorithms and applications in a variety of areas in CS and EE: electrical networks, VLSI physical design, systems level testing, communication protocol testing, parallel/distributed computing, telecommunication network planning, fault tolerance in optical networks, interconnection networks etc. He has published more than 100 papers in archival journals, coauthored with M. N. S. Swamy two text books "Graphs, Networks, and Algorithms" (1981) and "Graphs: Theory and Algorithms" (1992), both published by Wiley Inter-Science, and authored two chapters in the Handbook of Circuits and Filters (CRC and IEEE, 1995) and a chapter on "Graphs and Vector Spaces " for the handbook of Graph Theory and Applications (CRC Press,2003).

Dr. Thulasiraman has received several awards and honors: Endowed Gopalakrishnan Chair Professorship in CS at IIT, Madras (Summer 2005), Elected member of the European Academy of Sciences (2002), IEEE CAS Society Golden Jubilee Medal (1999), Fellow of the IEEE (1990) and Senior Research Fellowship of the Japan Society for Promotion of Science (1988). He has held visiting positions at the Tokyo Institute of Technology, University of Karlsruhe, University of Illinois at Urbana-Champaign and Chuo University, Tokyo.

Dr. Thulasiraman has been Vice President (Administration) of the IEEE CAS Society (1998, 1999), Technical Program Chair of ISCAS (1993, 1999), Deputy Editor-in-Chief of the IEEE Transactions on Circuits and Systems I (2004-2005), Co-Guest Editor of a special issue on "Computational Graph Theory: Algorithms and Applications" (IEEE Transactions on CAS, March 1988), Associate Editor of the IEEE Transactions on CAS (1989-91, 1999-2001), and Founding Regional Editor of the Journal of Circuits, Systems, and Computers. Recently, he founded the Technical Committee on "Graph theory and Computing" of the IEEE CAS Society.

Guoliang Xue is a Professor in the Department of Computer Science and Engineering at Arizona State University. He received the Ph.D degree in Computer Science from the University of Minnesota in 1991 and has held previous positions at the Army High Performance Computing Research Center and the University of Vermont. His research interests include efficient algorithms for optimization problems in networking, with applications to fault tolerance, robustness, and privacy issues in networks ranging from WDM optical networks to wireless ad hoc and sensor networks. He has published over 100 papers in this area. His research has been continuously supported by federal agencies including NSF, ARO and DOE. He is the recipient of an NSF Research Initiation Award in 1994, and an NSF-ITR Award in 2003. He is an Associate Editor of the IEEE Transactions on Circuits and Systems-I, an Associate Editor of the Computer Networks Journal, and an Associate Editor of the IEEE Network Magazine. He has served on the executive/program committees of many IEEE conferences, including Infocom, Secon, Icc, Globecom and QShine. He is a TPC co-chair of IEEE Globecom'2006 Symposium on Wireless Ad Hoc and Sensor Networks.