Constrained Shortest Link-Disjoint Paths Selection: A Network Programming Based Approach

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

***Abstract*— Given a graph G with each link in the graph associated with two positive weights, cost and delay, we consider the problem of selecting a set of k link-disjoint paths from a node s to another node t such that the total cost of these paths is minimum and that the total delay of these paths is not greater than a specified bound. This problem, to be called the CSDP(k) problem, can be formulated as an integer linear programming (ILP) problem. Relaxing the integrality constraints results in an upper bounded linear programming problem. We first show that the integer relaxations of the CSDP(k) problem and a generalized version of this problem to be called the GCSDP(k) problem (in which each path is required to satisfy a specified bound on its delay) both have the same optimal objective value. In view of this we focus our work on the relaxed form of the CSDP(k) problem called RELAX-CSDP(k). We study RELAX-CSDP(k) from the primal perspective using the revised simplex method of linear programming. We discuss different issues such as formulas to identify entering and leaving variables, anti-cycling strategy, computational time complexity etc., related to an efficient implementation of our approach. We show how to extract from an optimal solution to RELAX-CSDP(k) a set of k link-disjoint s-t paths which is an approximate solution to the original CSDP(k) problem. We also derive bounds on the quality of this solution with respect to the optimum. We present simulation results that demonstrate that our algorithm is faster than currently available approaches. Our simulation results also indicate that in most cases the individual delays of the paths produced starting from RELAX-CSDP(k) do not deviate in a significant way from the individual path delay requirements of the GCSDP(k) problem.**

***Index Terms*— Constrained shortest paths, link-disjoint paths, QoS routing, graph theory, combinatorial optimization, linear programming, network optimization**

I. Introduction

In this paper we study a discrete optimization problem defined on graphs or networks (We use the terms ”graph” and ”network” interchangeably). Specifically, we are interested in selecting a set of paths satisfying certain constraints. This problem is fundamental and arises in several applications. In this context we encounter two problems. One of them, called the Constrained Shortest Path (CSP) problem, requires the determination of a minimum cost path from a source node s to a destination node t such that the delay of the path is within a specified bound. The other problem, denoted as CSDP(k), is to select a set of k link disjoint paths from s to t such that the total cost of these paths is minimum and that the total delay of these paths is not greater than a specified bound. Both problems are NP-hard [1], [2]. This has led researchers to propose heuristics and approximation algorithms for these problems.

For a detailed account of the different algorithms for the CSP problem, [2]–[13] may be consulted. Of special interest to us in the context of our work in this paper are [8], [10]–[13]. References [8] and [10]–[12] present the LARAC algorithm that is based on the Lagrangian dual of the CSP problem as well as some generalizations. Reference [11] presents a generalization of the LARAC algorithm and is applicable to general combinatorial optimization problems involving two additive metrics. One of such problem is the minimum cost spanning tree problem with the restriction that the total delay being within a specified limit. Several such constrained discrete optimization problems arise in the VLSI physical design area. Reference [13] shows that these algorithms are strongly polynomial. In perhaps the most recent work [14] on the CSP problem, we have studied this problem from a primal perspective.

The CSDP(k) problem arises in the context of provisioning paths that could be used to provide resilience to failures in one or more of these paths. Orda et al. [15] have studied the CSDP(2) problem and have provided several approximation algorithms. A special case of the CSDP(k) problem which does not have the delay requirement has been studied in [16]. The algorithms in [11] and [16] can be integrated to provide an approximate solution to the CSDP(k) problem. We call this the G-LARAC(k) algorithm.

The rest of the paper is organized as follows. In Section II we define the CSDP(k) problem and a generalized version of this problem called the GCSDP(k) problem. The GCSDP(k) problem requires that the delay of each path in the set of link-disjoint paths be bounded by a specified value. This is in contrast to the CSDP(k) problem wherein the delay constraint is with respect to the total delay of the paths. However, even finding two delay constrained link-disjoint paths is NP-hard and is not approximable within a factor of $2-\epsilon$ for any $\epsilon > 0$ [17]. We first show that the optimal objective values of the LP relaxations of these two problems have equal value. Hence we focus our study on the relaxed version of the CSDP(k) problem, namely, the RELAX-CSDP(k) problem. In Section III we review the G-LARAC(k) algorithm which is a dual based approach to solving RELAX-CSDP(k). In Section IV we introduce a transformation on the CSDP(k) problem. The

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: ying_xiao@ou.edu, thulsi@ou.edu). Guoliang Xue is with Arizonia State University, Tempe, AZ 85287, USA. (e-mail: xue@asu.edu)

transformed problem will be called the TCSDP(k) problem. We show that the CSDP(k) problem and the TCSDP(k) problem are equivalent. As we show later in the paper the transformed problem has several properties that enable us to achieve an efficient implementation of our approach. In the remainder of the paper we study the LP relaxation of the TCSDP(k) problem, namely, RELAX-TCSDP(k), using the revised simplex method of linear programming. In Section V, several properties of basic solutions of RELAX-TCSDP(k) are established. We also show how to extract an approximate solution to the CSDP(k) problem starting from an optimal solution to RELAX-TCSDP(k). In Sections VI-VII, the revised simplex method and several issues relating to an efficient implementation are discussed. We also develop an anti-cycling strategy and establish the pseudo-polynomial time complexity of the revised simplex method when applied on RELAX-TCSDP(k). Simulation results comparing our approach with the G-LARAC(k) algorithm and the commercially available CPLEX package are presented in Section VIII. These results demonstrate that our algorithm is faster than currently available approaches. They also indicate that in most cases the individual delays of the paths produced starting from RELAX-CSDP(k) do not deviate in a significant way from the individual delay requirements of the GCSDP(k) problem, thereby demonstrating that there is not much loss of generality in focusing on RELAX-CSDP(k) rather than on the relaxed version of the more complex GCSDP(k) problem. We conclude in Section IX with a summary of our work and pointing to certain directions for future research.

II. Constrained Shortest Link-Disjoint Paths Selection Problems: Formulations, Relaxations, and Their Equivalence

In this section we first define two classes of link-disjoint paths selection problems. One is a special case of the other. They both admit integer linear programming (ILP) formulations. They are computationally intractable because of the integrality constraints. For networks involving small numbers of nodes and links, these problems can be solved using any general purpose ILP package. For larger networks, faster algorithms are desired. So, we are interested in solving these problems after relaxing the integrality requirement and exploiting the special network structure of these problems for efficient algorithms. The relaxed versions of these problems are upper bounded linear programming problems. The main result in this section is that the relaxed versions of both problems are equivalent in the sense they have the same optimal objective value.

We begin with some basic definitions. The network is modeled as a directed graph $G(V,E)$, where V and E are the sets of nodes and links, respectively. Each link $(u,v) \in E$ is associated with a positive integer cost c_{uv} and a positive integer delay d_{uv}. A path is a sequence of distinct nodes $u_1, u_2 \ldots, u_l$ such that either $(u_i, u_{i+1}) \in E$ or $(u_{i+1}, u_i) \in E$ for all $1 \le i \le l-1$ and $u_i \ne u_j$ if $i \ne j$. A link on a path from u to v is a forward link (backward) link if its orientation agrees (disagrees) with the direction of the traversal of the path from u to v. The sets of forward and backward links on p will be denoted by p^+ and p^-, respectively. For any directed path p (or cycle with given orientation) define cost $c(p)$ and delay $d(p)$ of p as

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

Given any two nodes s and t, an s-t path is a directed s-t path if all the links on the path are forward links. For the simplicity, we shall call such directed paths simply as s-t paths. An s-t path is called feasible with respect to the delay and a specified value T if the delay of the path is at most T. Without loss of generality we assume that for every node u there is a directed path from s to u and a directed path from u to t.

General Constrained Shortest k-Disjoint Paths (GCSDP(k)) Problem: Given two nodes s and t and a positive integer T, the GCSDP(k) problem is to find a set of $k(k \ge 2)$ link-disjoint s-t paths $p_1, p_2 \ldots, p_k$ such that the delay of each path p_i is at most T and the total cost of the k paths is minimum.

Constrained Shortest k-Disjoint Paths (CSDP(k)) Problem: Given two nodes s and t, and a positive integer T, the CSDP(k) problem is to find a set of k link disjoint s-t paths $p_1, p_2 \ldots, p_k$ such that the total delay of these paths is at most kT and that the total cost of the k paths is minimum.

Both the above problems can be formulated as ILP problems. Relaxing the integrality constraints we get the following relaxed versions of these problems.

RELAX-GCSDP(k):

$$\text{Minimize} \sum_{(u,v)\in E} c_{uv} \sum_{i=1}^{k} x^i_{uv} \tag{1}$$

subject to

For $i = 1, 2 \ldots k$ and $\forall u \in V$,

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

$$\sum_{(u,v)\in E} d_{uv} \cdot x^i_{uv} \le T \tag{3}$$

$$\sum_{i=1}^{k} x^i_{uv} \le 1 \text{ and } x^i_{uv} \ge 0, \forall (u,v) \in E \tag{4}$$

The solutions to the above problem may not, in general, be integral. However, every integer solution defines a set of k link-disjoint s-t paths. In other words, an integer solution $X^i = \{x^i_{uv}\}_{(u,v)\in E}$ for $i = 1, 2 \ldots, k$ is the flow vector corresponding to the ith path p_i, i.e., link (u,v) is on path p_i iff $x^i_{uv} = 1$.

RELAX-CSDP(k):

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

subject to $\forall u \in V$,

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

$$\sum_{(u,v)\in E} d_{uv} \cdot x_{uv} \le kT \tag{7}$$

$$0 \le x_{uv} \le 1, \forall (u,v) \in E$$

We now proceed to show that the RELAX-GCSDP(k) and RELAX-CSDP(k) are equivalent in the sense that they both have optimal solutions with the same value for the objective.

Let $\Lambda = (\lambda_1, \lambda_2 \ldots, \lambda_k) \ge \mathbf{0}$ and define

$$L_G(k,\Lambda) = \min\{ \sum_{(u,v)\in E} c_{uv} \sum_{i=1}^{k} x^i_{uv} + \sum_{i=1}^{k} \lambda_i (\sum_{(u,v)\in E} d_{uv} \cdot x^i_{uv} - T)\}$$

Then the Lagrangian dual of RELAX-GCSDP(k) is as follows.

LAGRANGIAN-GCSDP(k):

Maximize $L_G(k,\Lambda)$, among all $\Lambda \ge \mathbf{0}$

subject to $\forall i = 1, 2 \ldots k$ and $\forall u \in V$,

$$\sum_{\{v|(u,v)\in E\}} x^i_{uv} - \sum_{\{v|(v,u)\in E\}} x^i_{vu} = \begin{cases} 1, & \text{for } u = s \\ -1, & \text{for } u = t \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$\sum_{i=1}^{k} x^i_{uv} \le 1 \text{ and } x^i_{uv} \ge 0, \forall (u,v) \in E \tag{9}$$

The vector Λ is called the Lagrangian multiplier. The above problem can be solved by finding the Lagrangian multiplier vector Λ that maximizes $L_G(k,\Lambda)$.

Property 1: Given any $\Lambda = (\lambda_1 \ldots, \lambda_k)$, let Λ' be obtained by permuting the components of Λ. Then $L_G(k,\Lambda) = L_G(k,\Lambda')$.

Property 2: $L_G(k,\Lambda)$ is a concave function of Λ [18].

Property 3: There exists Λ with all components equal that maximizes $L_G(k,\Lambda)$.

By Property 3, $L_G(k,\Lambda)$ can be reformulated with respect to some $\Lambda = (\lambda, \lambda \ldots, \lambda) \ge \mathbf{0}$ as follows:

$$L_G(k,\Lambda) = \min\{ \sum_{(u,v)\in E} c_{uv} \sum_{i=1}^{k} x^i_{uv} + \lambda (\sum_{(u,v)\in E} (d_{uv} \sum_{i=1}^{k} x^i_{uv}) - kT)\}. \tag{10}$$

Let

$$\bar{x}_{uv} = \sum_{i=1}^{k} x^i_{uv}, \forall (u,v) \in E. \tag{11}$$

We now define UNIFORM-LAGRANGIAN-GCSDP(k) as follows.

First let

$$L(k,\lambda) = \min\{ \sum_{(u,v)\in E} c_{uv} \cdot \bar{x}_{uv} + \lambda (\sum_{(u,v)\in E} d_{uv} \cdot \bar{x}_{uv} - kT)\}.$$

UNIFORM-LAGRANGIAN-GCSDP(k):

$$\text{Maximize } L(k,\lambda) \text{ among all } \lambda \ge 0 \tag{12}$$

subject to

$$\sum_{\{v|(u,v)\in E\}} \bar{x}_{uv} - \sum_{\{v|(v,u)\in E\}} \bar{x}_{vu} = \begin{cases} k, & \text{for } u = s \\ -k, & \text{for } u = t \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$0 \le \bar{x}_{uv} \le 1, \forall (u,v) \in E. \tag{14}$$

Note that (13) is obtained by summing up the k flow balance constraints in (8) and that λ is a scalar.

Theorem 1: UNIFORM-LAGRANGIAN-GCSDP(k) and LAGRANGIAN-GCSDP(k) have the same optimal value for the objective.

Proof: Let $\Lambda = (\lambda, \lambda \ldots, \lambda) \ge \mathbf{0}$. We first show that $L_G(k,\Lambda) \ge L(k,\lambda)$.

Let $\{x^i_{uv}\}_{(u,v)\in E, i=1\ldots,k}$ minimize $L_G(k,\Lambda)$. Then we have

$$\begin{aligned} L_G(k,\Lambda) &= \sum_{(u,v)\in E} c_{uv} \sum_{i=1}^{k} x^i_{uv} + \sum_{i=1}^{k} \lambda (\sum_{(u,v)\in E} d_{uv} x^i_{uv} - T) \\ &= \sum_{(u,v)\in E} c_{uv} \bar{x}_{uv} + \lambda (\sum_{(u,v)\in E} d_{uv} \bar{x}_{uv} - kT) \ge L(k,\lambda), \end{aligned}$$

where $\bar{x}_{uv}$ is defined as in (11).

It follows from the unimodularity [19] of the constraints (13)-(14) that for a given λ, there exists an optimal integer solution to UNIFORM-LAGRANGIAN-CSDP(k) problem. Also an integer solution $Y = \{y_{uv}\}_{(u,v)\in E}$ of UNIFORM-LAGRANGIAN-CSDP(k) that achieves the minimum in $L(k,\lambda)$ defines a set of k link-disjoint s-t paths $P_k = (p_1, p_2 \ldots, p_k)$. Let $X^i = \{x^i_{uv}\}_{(u,v)\in E}$ be the flow vector for path p_i, i.e., $x^i_{uv} = 1$ iff $(u,v) \in p_i$; otherwise, $x^i_{uv} = 0$.

Observe that $y_{uv} = \sum_{i=1}^{k} x^i_{uv}$. Then

$$\begin{aligned} L(k,\lambda) &= \sum_{(u,v)\in E} c_{uv} y_{uv} + \lambda (\sum_{(u,v)\in E} d_{uv} y_{uv} - kT) \\ &= \sum_{(u,v)\in E} c_{uv} \sum_{i=1}^{k} x^i_{uv} + \sum_{i=1}^{k} \lambda (\sum_{(u,v)\in E} d_{uv} x^i_{uv} - T) \\ &\ge L_G(k,\Lambda). \end{aligned}$$

Hence, $L(k, \lambda) \geq L_G(k, \Lambda)$. So $L(k, \lambda) = L_G(k, \Lambda)$.

By Property 3 there exists a vector $\Lambda^* = (\lambda^*, \lambda^* \ldots, \lambda^*)$ that maximizes $L_G(k, \Lambda)$. Let η^* be a maximizing multiplier for $L(k, \lambda)$ and denote $H^* = (\eta^* \ldots, \eta^*)$.

By definition of Λ^* and η^*, we have $L_G(k, \Lambda^*) = L(k, \lambda^*)$ $\leq L(k, \eta^*) = L_G(k, H^*) \leq L_G(k, \Lambda^*)$.

Hence $L_G(k, \Lambda^*) = L(k, \eta^*)$. ■

The above theorem has an important implication. It shows that the optimal objective to RELAX-GCSDP(k) can be obtained by solving UNIFORM-LAGRANGIAN-GCSDP(k). But UNIFORM-LAGRANGIAN-GCSDP(k) is the general linear programming dual of the RELAX-CSDP(k) problem (See page. 183 of [20]). Thus we have the following result by the strong duality theorem [20].

Theorem 2: RELAX-GCSDP(k) and RELAX-CSDP(k) have the same optimal objective value.

The intuition behind the above result is as follows. The indistinguishability of the k path constraints represented by (3) guarantees that if P is a set of feasible paths constituting a solution to RELAX-GCSDP(k) problem then any permutation of these paths is also a solution (Property 1). Also in the optimal solution there is no reason for paths to be weighted differently (Property 3). As formally proved, these two properties lead to Theorem 2.

Theorem 2 implies that if we are interested only in obtaining the optimal objective value of the RELAX-GCSDP(k), then starting with the RELAX-CSDP(k) does not result in any loss of generality. In view of this, we shall focus on RELAX-CSDP(k) in the rest of the paper.

III. G-LARAC(k) Algorithm: A Dual Based Approach to RELAX-CSDP(k)

The G-LARAC(k) algorithm is a generalization of the LARAC algorithm [8]–[10] that was specifically designed for CSP problem. The G-LARAC(k) algorithm may be viewed as an algorithm for solving RELAX-CSDP(k) problem using its Lagrangian dual which is the same as UNIFORM-LAGRANGIAN-GCSDP(k) repeated below.

UNIFORM-LAGRANGIAN-GCSDP(k):

Maximize $L(k, \lambda)$

subject to $\forall u \in V$,

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

$$0 \leq x_{uv} \leq 1, \forall (u,v) \in E.$$

In the rest of the paper, we shall use Δ in place of kT for the simplicity of writing.

Given λ, $L(k, \lambda)$ is achieved by a set of k link-disjoint paths with minimum total weight, where the weight associated with link (u, v) is given by $c_{uv} + \lambda d_{uv}$. The key issue is how to search for the optimal λ that maximizes $L(k, \lambda)$ and determining the termination condition for the search. The G-LARAC(k) algorithm presented as Algorithm 1 is one such efficient search procedure. In this procedure c_λ cost of a path (also called aggregated cost) refers to the weight of the path computed using $c_{uv} + \lambda d_{uv}$ as the weight of link (u, v).

Algorithm 1 G-LARAC(s, t, k, $\Delta = kT$) algorithm

```
{compute $k$-link disjoint paths with minimum total cost}
$P_c \leftarrow$ Disjoint$(s, t, c, k)$
if ($d(P_c) \leq \Delta$) then return $P_c$
{compute $k$-link disjoint paths with minimum total delay}
$P_d \leftarrow$ Disjoint$(s, t, d, k)$
if ($d(P_d) > \Delta$) then return "no solution"
loop
    $\lambda \leftarrow (c(P_c) - c(P_d))/(d(P_d) - d(P_c))$
    {compute $k$-link disjoint paths with minimum $c_\lambda$ cost}
    $R \leftarrow$ Disjoint$(s, t, c_\lambda, k)$
    if ($c_\lambda(R) = c_\lambda(P_c)$) then return $P_d$
    else if ($d(R) \leq \Delta$) then $P_d \leftarrow R$ else $P_c \leftarrow R$
end loop
```

Basically G-LARAC(k) performs the following steps.

1) In the first step, the algorithm calculates the minimum cost of a set of k link-disjoint s-t paths using link costs. This can be done by the algorithm in [16]. If the total delay of these paths is at most Δ, this is surely the required set of paths. Otherwise, the algorithm stores this set as the latest infeasible set, simply called the P_c set. Then it determines the minimum delay of a set of k link disjoint s-t paths, called the P_d set. If P_d is infeasible, there is no solution to this instance.
2) Set $\lambda = (c(P_c) - c(P_d))/(d(P_d) - d(P_c))$. With this value of λ, we can find a set of k link-disjoint paths with minimum c_λ-cost. Let this set be denoted as R. If $c_\lambda(R) = c_\lambda(P_c)(= c_\lambda(P_d))$, we have obtained the optimal λ. Otherwise, set R as the new P_c or P_d according to whether R is infeasible or feasible.

A detailed discussion of several issues relating to G-LARAC(k) and properties of solutions produced by G-LARAC(k) may be found in [12].

We wish to note that when Lagrangian dual is introduced, an integrated metric is used by adding the penalty of extra delay into the cost function. However, this integration makes the new model deviate from the original problem. For example, consider $k = 1$. If there is a path which has a delay less than Δ, it will add a negative number into the cost function as a reward. But in the original problem, there is no such reward. One way to remedy this situation is to use the formulation $\min\{c(p) + \lambda \max\{0, d(p) - \Delta\}\}$. However, if we use this formulation we cannot reduce it to the shortest path problem on $c+\lambda d$ for fixed λ. This sub-problem is non-additive shortest path problem. Finding this minimum is intractable and is as difficult as the original CSDP(k) problem.

In contrast to the dual approach taken by the G-LARAC(k) algorithm our interest in the remainder of the paper is to design an approach to obtain an approximate solution to the CSDP(k) problem using the primal simplex method of linear programming.

IV. Transformation of the RELAX-CSDP(k) Problem

To achieve an efficient implementation of our approach to the RELAX-CSDP(k) problem we consider another problem TCSDP(k) on a transformed network defined as follows.

1) The graph of the transformed problem is the same as that of the original problem, that is, $G(V,E)$,
2) For all $(u,v) \in E$, 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 $\Delta' = 2\Delta + 1$.

Let P^k denote a set of k link-disjoint s-t paths. Let $c(P^k)$ and $d(P^k)$ denote the total cost and the total delay of the k paths in P^k. The TCSDP(k) problem asks for a set of k link-disjoint s-t paths with minimum total cost and with total delay at most Δ'.

Theorem 3: P^k is a feasible solution (resp. an optimal solution) to the CSDP(k) problem iff it is a feasible solution (resp. an optimal solution) to the TCSDP(k) problem.

In view of this equivalence we only consider, in the rest of the paper, the RELAX-TCSDP(k) problem. Also we use Δ (being odd) and d_{uv} (being even) to denote the delay bound and link delay in the TCSDP(k) problem. Notice that the transformation does not change the costs of paths.

We conclude this section defining some terminology and presenting the RELAX-TCSDP(k) problem in matrix form.

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 each 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 [21]. 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 of rank $(n-1)$ as H. We also assume that the row removed from the incidence matrix corresponds to node n. We denote the column of H corresponding to e_k by the vector $\boldsymbol{h_k}$. For $e_k = (i,j), i,j \neq n$ 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 $h_{j,k} = -1$. Also for $e_k = (i,n), h_{i,k} = 1$, and for $e_k = (n,j), h_{j,k} = -1$ and all the rest components are 0. Let $D = (-d_1, -d_2 \ldots, -d_m)$ and

$$A = \begin{pmatrix} H & \mathbf{0} \\ D & -1 \end{pmatrix} = (\boldsymbol{a}_1, \boldsymbol{a}_2 \ldots, \boldsymbol{a}_m, \boldsymbol{a}_{m+1}), \tag{15}$$

$$\boldsymbol{a}_i = \begin{pmatrix} \boldsymbol{h}_i \\ -d_i \end{pmatrix}, i \leq m, \text{ and } \boldsymbol{a}_{m+1} = \begin{pmatrix} \mathbf{0} \\ -1 \end{pmatrix}. \tag{16}$$

Let $\boldsymbol{x}$ be the column vector of the m flow variables x_{uv} and the slack variable w corresponding to the delay constraint (7), and $\boldsymbol{c}$ be the row vector $(c_1, c_2 \ldots, c_m, 0)$ of the costs. Note that the cost of the slack variable is 0. Then the RELAX-TCSDP(k) problem (see (5)-(7)) can be written in matrix form as follows. Note that to conform to the standard form for a minimization problem we have used "$\geq$" form of (7) and added a slack variable w, i.e., $\sum_{(u,v)\in E} -d_{uv}x_{uv} - w \geq -\Delta$.

$$\begin{aligned} &\text{RELAX-TCSDP}(k): \\ &\quad \text{Minimize } \boldsymbol{cx} \\ &\quad \text{subject to } A\boldsymbol{x} = \boldsymbol{b} \\ &\quad \mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}, \forall (u,v) \in E \text{ and } w \geq 0, \end{aligned} \tag{17}$$

where w is the slack variable and $\boldsymbol{b} = (b_1 \ldots, b_{n-1}, -\Delta)^t$ with $b_s = k$, $b_t = -k$, and $b_i = 0$ for $i \neq s,t$.

We note that the above problem is almost the same as the minimum cost flow problem except for the additional delay constraint.

The rest of the paper is concerned with the simplex method based solution to RELAX-TCSDP(k) and several issues relating to its efficient implementation. We want to emphasize that most of these properties hold only with the transformation and we shall use "*" to denote those properties that also hold without the transformation. The cost of the optimal solution to RELAX-TCSDP(k) will be a lower bound to the optimal cost of the original CSDP(k) problem. We will show in the next section how to extract an approximate solution to TCSDP(k) (hence the CSDP(k)) problem from an optimal solution to the RELAX-TCSDP(k) problem.

V. Properties of Basic Solutions of RELAX-TCSDP(k) and Generation of an Approximate Solution to CSDP(k)

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 nonbasic. For the RELAX-TCSDP(k) problem under consideration, all the nonbasic variables in a basic solution will be 0 or 1 [22]. Note that the value of the slack variable, when it is nonbasic, must be equal to 0 because it does not have an upper bound. Given a basic solution, we shall denote the subgraph of G corresponding to the basic variables (except the slack variable if it is in the basic solution) in this solution by G_b. The subgraph G_b will be called the subgraph of the basic solution or simply the basis graph. The nonsingular submatrix of A defined by the basic variables is called a basis matrix or simply, a basis and is denoted as B. The rest of the matrix corresponding to the nonbasic variables is called the nonbasic matrix. In this section we present certain important properties of the basic solutions of the RELAX-TCSDP(k) 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 $U = (u_1, u_2, u_3 \ldots, u_m)^t$, where

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

Then, $HU = 0$ [21].

We shall use $U(W)$ to denote the vector derived from cycle W as in the above lemma. We shall denote by $d(W)$ the signed algebraic sum of the delays of the links on a cycle W as we traverse around the cycle.

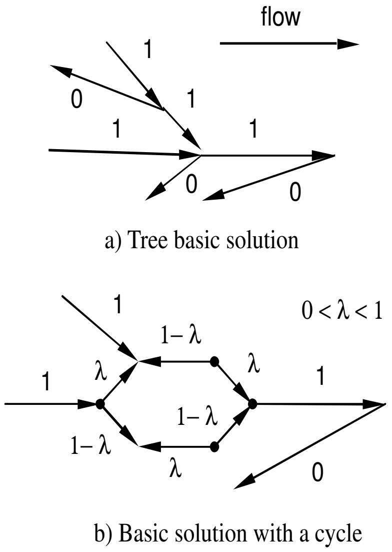


Fig. 1. Structure of basic solutions

Lemma 2: * Every basis matrix contains the last row of A.

Proof: This follows from the fact that rank(H)$= n-1 < n$. ■

Lemma 3: * The subgraph G_b of a basic solution contains at most one cycle (See Fig. 1).

Lemma 4: * If there is a cycle W in a basic solution, then $d(W) \neq 0$.

Lemma 5: If there is no cycle in a basic solution, then $\forall (u,v) \in E, x_{uv} = 0$ or 1. If there is a cycle W in a basic solution, then $\forall (u,v) \in W$, $0 < x_{uv} < 1$ and $\forall (u,v) \in E - W$, $x_{uv} = 0$ or 1.

Proof: Let $B = (\boldsymbol{b}_1, \boldsymbol{b}_2 \ldots, \boldsymbol{b}_n)$, A_N, $\boldsymbol{x}_B$ and $\boldsymbol{x}_N$ denote the basis matrix, nonbasic matrix, column vector of basic variables, and column vector of nonbasic variables in the basic solution, respectively.

Let $\boldsymbol{b}' = \boldsymbol{b} - A_N \boldsymbol{x}_N$, then we have $B\boldsymbol{x}_B = \boldsymbol{b}'$.

Since all the components in $A_N \boldsymbol{x}_N$ and $\boldsymbol{b}$ are integers, so are all the components in $\boldsymbol{b}'$.

By Cramer's rule, we have $x_i = \det(B_i)/\det(B)$, where $B_i = (\boldsymbol{b}_1 \ldots, \boldsymbol{b}_{i-1}, \boldsymbol{b}', \boldsymbol{b}_{i+1} \ldots, \boldsymbol{b}_n)$.

We first show that x_i is an integer if the corresponding link is not on the cycle. We consider two cases:

Case 1: There is no cycle in the basic solution. Thus the slack variable is a basic variable. Also G_b is a spanning tree. Let the nth column in the basis B correspond to the slack variable.

Then $\boldsymbol{b}_n = (0 \ldots, 0, -1)^t$ and so B has the following form. $B = \begin{pmatrix} H' & \boldsymbol{0} \\ D' & -1 \end{pmatrix}$, where H' is the incidence matrix of G_b and D' is the corresponding delay vector.

Since H' is the incidence matrix of a spanning tree it follows that $|\det(H')| = 1$. So $|\det(B)| = 1$. Also, $\det(B_i)$ is an integer because all the components of B_i are integers. So x_i is also an integer for all i.

Case 2: There is a cycle W in the basic solution. That is, the slack variable is not in the basis.

Let $l = |W|$, i.e., l is the number of links in W.

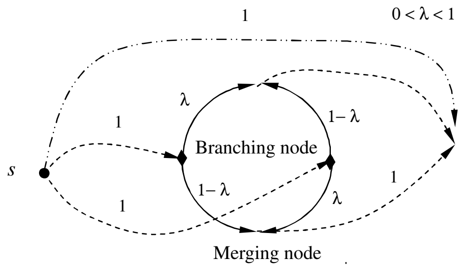


Fig. 2. Branching and merging nodes

In this case, we first show that the flow on any link i not on the cycle is an integer. Without loss of generality, let

$$B = \begin{pmatrix} H_W & H' \\ D_W & D' \end{pmatrix} \text{ and } B_i = \begin{pmatrix} H_W & H'_i \\ D_W & D'_i \end{pmatrix},$$

where the columns of $H_W = (\boldsymbol{h}_1, \boldsymbol{h}_2 \ldots, \boldsymbol{h}_l)$ correspond to the links on the cycle W and the components of D_W are the delays of these links. Note that H'_i contains the column vector $\boldsymbol{b}'$.

Let $H'_W = (\boldsymbol{h}_2 \ldots, \boldsymbol{h}_l)$ and $D'_W = (d_2 \ldots, d_l)$. Defining the direction of the link $\boldsymbol{h}_1$ as the orientation of the cycle W we get by Lemma 1 that $H_W U(W) = 0$.

Using elementary column operations, $\det(B)$ and $\det(B_i)$ can be written as:

$$\begin{aligned} \det(B) &= \det \begin{pmatrix} \boldsymbol{0} & H'_W & H' \\ D_W U(W) & D'_W & D' \end{pmatrix} \\ &= (-1)^{n+1} \det(H'_W \;\; H')(D_W U(W)), \text{ and} \\ \det(B_i) &= \det \begin{pmatrix} \boldsymbol{0} & H'_W & H'_i \\ D_W U(W) & D'_W & D'_i \end{pmatrix} \\ &= (-1)^{n+1} \det(H'_W \;\; H'_i)(D_W U(W)). \end{aligned}$$

Hence $x_i = \det((H'_W \;\; H'_i))/\det((H'_W \;\; H'))$.

Since all the components in matrix $(H'_W \;\; H'_i)$ are integer, $\det((H'_W \;\; H'_i))$ is also integer.

The denominator is equal to ± 1 because $(H'_W \;\; H')$ is the incidence matrix of the spanning tree obtained by removing link i from G_b. So it is follows that x_i is an integer. Hence $x_{uv} = 0$ or 1 because $0 \leq x_{uv} \leq 1$.

We next show that if the basis graph contains a cycle, then the flow on each link on the cycle W is less than 1 and greater than 0. Assuming the contrary we establish a contradiction. First recall that the flow on each link that is not in G_b (that is, each nonbasic variable) is either 0 or 1. If the flow on any link on W is an integer (0 or 1) then it follows from the flow balance constraints that all the flows on the links on W will be integers. But this would mean that in the current basic solution the total delay of all the links is an even integer. This violates the requirement that the total delay must be equal to Δ which is odd. ■

Definition 1: (a) On a directed cycle, a node is called a branching (resp. merging) node if it is the tail (resp. head) of two links on the cycle (See Fig. 2). A segment of the cycle is the set of all the links on the cycle between two consecutive branching and merging nodes. A segment consists of consecutive links with the same direction and the direction of a segment is defined as the direction of its links.

(b) For a subgraph G_s of G, let $d(G_s) = \sum_{(u,v)\in G_s} d_{uv}$ and $d_{\boldsymbol{x}}(G_s) = \sum_{(u,v)\in G_s} x_{uv}d_{uv}$ with respect to the flow vector $\boldsymbol{x}$.

Lemma 6: Suppose the basis graph G_b contains a cycle W. Let $e = (u,v) \in W$ and $x_{uv} = \lambda(0 < \lambda < 1)$. Define the direction of e as the orientation of W. Then for any link $e' = (i,j)$, $x_{ij} = \lambda$ if the direction of e' agrees with the orientation of W and $x_{ij} = 1 - \lambda$ otherwise (See Fig. 2).

Proof: This follows from the flow balance constraints and the fact the nonbasic variables corresponding to links have value 0 or 1. ■

Theorem 4: Given an optimal solution $\boldsymbol{x}$ to RELAX-TCSDP(k) with G_b as the corresponding basis graph.

(a) If G_b contains no cycle then G_b contains a set of k link disjoint s-t paths P^k with $d(P^k) < \Delta$, where Δ is the delay constraint in the TCSDP(k) problem. These paths constitute an optimal solution to the original CSDP(k) problem.

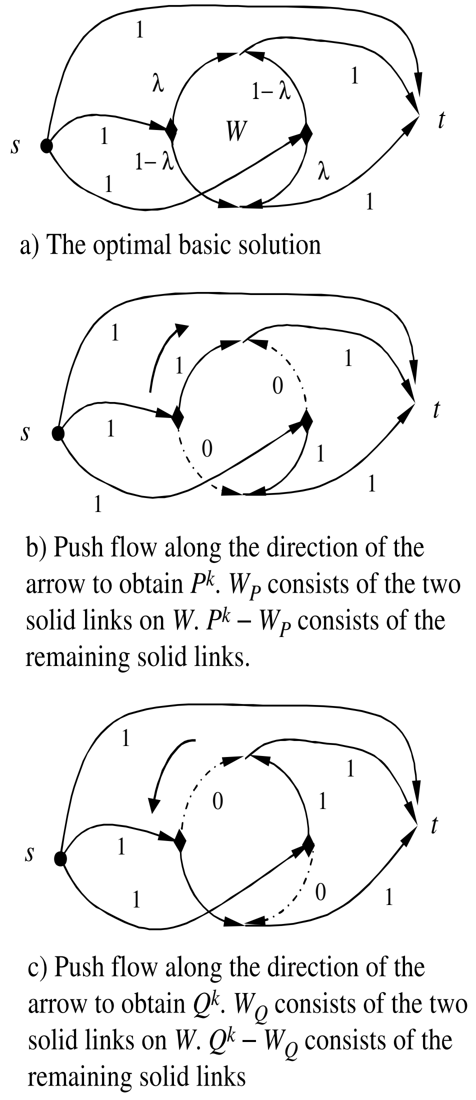
(b) Suppose that G_b contains a cycle W and let $G'(V',E')$ be obtained from $G(V,E)$ such that $V' = V$ and $E' = \{(u,v) \in E | x_{uv} > 0\}$. Then G' contains a set of k link disjoint s-t paths P^k with $d(P^k) < \Delta$, and a set of k link disjoint s-t paths Q^k with $d(Q^k) > \Delta$, such that $c(Q^k) \le OPT \le c(P^*) \le c(P^k)$, where OPT is the optimal objective value of the RELAX-TCSDP(k) problem, P^* is the optimal integer solution to the TCSDP(k) problem (equivalently, the optimal solution to the CSDP(k)) problem). Also $\frac{c(P^k)}{c(P^*)} \le 1 + \frac{1-\lambda}{\lambda}(1 - \frac{c(Q^k)}{c(P^k)})$ and $\frac{d(Q^k)}{\Delta} \le 1 + \frac{\lambda}{1-\lambda}(1 - \frac{d(P^k)}{\Delta})$, where λ is as defined in Lemma 6 with the orientation of the cycle W selected so that $d(W) < 0$.

Proof: (a) If there is no cycle in the optimal basis graph G_b, then all the link flows will be integers and the flow vector can be decomposed into unit flows along k link-disjoint s-t paths. The total delay of these paths will be even and hence less than Δ (because Δ is odd). These integral flows form an optimal integer solution to the RELAX-TCSDP(k) problem and hence an optimal solution to the original TCSDP(k) problem. By Theorem 3 this is also an optimal solution to the original CSDP(k) problem.

(b) Assume that the basis graph contains a cycle W. By Lemma 5, the flows on links on W are nonzero and thus G' contains W. Obviously, $OPT \le c(P^*)$. Also note that $d_{\boldsymbol{x}}(G') = \Delta$ because the slack variable is nonbasic and thus its value in the current basic solution is 0.

Define the orientation of W such that $d(W) < 0$. Now push flow along the orientation of W until some link's flow reaches 0 or 1 (See Fig. 3-(b)). By Lemma 6, all the resultant link flows will be either 0 or 1. Remove all the links with zero flow from G' and let G_z denote the resultant network. Evidently, the flows on all links in G_z are 1 and $d(G_z) < d_{\boldsymbol{x}}(G') = \Delta$ (because $d(W) < 0$, the network delay is reduced when we push the flow along the orientation of W). It can also be seen that $c(W) \ge 0$ for otherwise, the cost of the new flow will be less than the cost of the flow defined by G'.

Notice that the above operation does not change the amount of flow from s to t. Since the total flow from s to t in G_z is k and all the flows on all links in G_z are 1, there must be k link disjoint s-t paths P^k and $d(P^k) = d(G_z) < \Delta$.



a) The optimal basic solution

b) Push flow along the direction of the arrow to obtain P^k. W_P consists of the two solid links on W. $P^k - W_P$ consists of the remaining solid links.

c) Push flow along the direction of the arrow to obtain Q^k. W_Q consists of the two solid links on W. $Q^k - W_Q$ consists of the remaining solid links

Fig. 3. Illustrations for the proof of Theorem 4

Similarly, we can obtain Q^k(See Fig. 3-(c)). Here, the flow is pushed along W in the reverse direction. Along this direction, $d(W) > 0$ and so $d(Q^k) > \Delta$.

Since $c(W) < 0$ along this direction, $c(Q^k) \le OPT \le c(P^*) \le c(P^k)$.

In the rest of the proof, we use P^k, Q^k and W to denote the corresponding set of links. Let $W_P = P^k \cap W$ and $W_Q = Q^k \cap W$, i.e., W_P(resp. W_Q) is the set of links on both the cycle W and P^k (resp. Q^k). Evidently, $W_P \cap W_Q = \emptyset$ and $P^k - W_P = Q^k - W_Q$(See Fig. 3. (b) and (c)).

Then we have

$$\begin{aligned} c(P^*) \ge OPT &= c(P^k - W_P) + \lambda c(W_P) + (1-\lambda)c(W_Q) \\ &= \lambda c(P^k - W_P) + (1-\lambda)c(Q^k - W_Q) \\ &\quad + \lambda c(W_P) + (1-\lambda)c(W_Q) \\ &= \lambda(c(P^k - W_P) + c(W_P)) \\ &\quad + (1-\lambda)(c(Q^k - W_Q) + c(W_Q)) \\ &= \lambda c(P^k) + (1-\lambda)c(Q^k). \end{aligned}$$

The second equality holds because $P^k - W_P = Q^k - W_Q$.

Because $c(P^*) \leq c(P^k)$,

$$\begin{aligned}\frac{c(P^k)}{c(P^*)} &\leq \frac{c(P^*) - (1-\lambda)c(Q^k)}{\lambda c(P^*)}\\ &= \frac{1}{\lambda} - \frac{1-\lambda}{\lambda}\frac{c(Q^k)}{c(P^*)} \leq \frac{1}{\lambda} - \frac{1-\lambda}{\lambda}\frac{c(Q^k)}{c(P^k)}\\ &= 1 + \frac{1-\lambda}{\lambda}(1 - \frac{c(Q^k)}{c(P^k)}).\end{aligned}$$

Similarly, we can show that $\frac{d(Q^k)}{\Delta} \leq 1 + \frac{\lambda}{1-\lambda}(1 - \frac{d(P^k)}{\Delta})$. ■

Notice that all the individual paths in the above theorem can be obtained using flow decomposition [19].

VI. Revised Simplex Method for the RELAX-CSDP(k) Problem

In this section, we first briefly present the different steps in the revised simplex method of linear programming. A detailed description of this method may be found in Chapter 8 of [22] (Note that in [22] the revised simplex method is presented for a maximization problem). 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.

A. Revised Simplex Method

Consider the following linear programming problem.

$$\text{Minimize } \boldsymbol{cx}$$
$$\text{subject to } A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}.$$

For the RELAX-CSDP(k) problem A is an $n \times (m+1)$ matrix with rank(A) = n, $\boldsymbol{x} = (x_1 \ldots, x_{m+1})^t$, $\boldsymbol{c} = (c_1, c_2 \ldots, c_{m+1})$, $\boldsymbol{b} = (b_1, b_2 \ldots, b_n)^t$. Each feasible basic solution $\boldsymbol{x}^*$ of this linear program 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 A into B and 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 matrix B is the basis matrix and is nonsingular. See Sections IV and V for the form of the basis matrix and properties of basic solutions for the RELAX-TCSDP(k) problem.

Revised Simplex Method [22]

1) Solve the system $YB = \boldsymbol{c_B}$, where $Y = (y_1, y_2 \ldots y_n)$.
2) Choose an entering variable x_j. This may be any non-basic variable x_j such that, with $\boldsymbol{a}$ standing for the corresponding column of A, we have either $Y\boldsymbol{a} > c_j$, $\boldsymbol{x}_j^* < u_j$ or $Y\boldsymbol{a} < c_j$, $x_j^* > l_j$. If there is no such variable then stop; the current solution $\boldsymbol{x}^*$ is optimal.
3) Solve the system $BV = \boldsymbol{a}$, where $V = (v_1, v_2 \ldots, v_n)^t$.
4) Define $x_j(t) = x_j^* + t$ and $\boldsymbol{x}_B(t) = \boldsymbol{x}_B^* - tV$ in case $Y\boldsymbol{a} < c_j$ and $x_j(t) = x_j^* - t$, $\boldsymbol{x}_B(t) = \boldsymbol{x}_B^* + tV$ in case $Y\boldsymbol{a} > c_j$. If the constraints $l_j \leq x_j(t) \leq u_j$, $\boldsymbol{l}_B \leq \boldsymbol{x}_B(t) \leq \boldsymbol{u}_B$ are satisfied for all positive t then the problem is unbounded. Otherwise set t as the largest value allowed by these constraints. If the upper bound imposed on t by the constraints $\boldsymbol{l}_B \leq \boldsymbol{x}_B(t) \leq \boldsymbol{u}_B$ is stricter than the upper bound imposed by $l_j \leq x_j(t) \leq u_j$, then determine the leaving variable. This may be any basic variable x_i such that the upper bound imposed on t by $l_i \leq x_i(t) \leq u_i$ alone is as strict as the upper bound imposed by all the constraints $\boldsymbol{l}_B \leq \boldsymbol{x}_B(t) \leq \boldsymbol{u}_B$.
5) Replace x_j^* by $x_j(t)$ and $\boldsymbol{x}_B^*$ by $\boldsymbol{x}_B(t)$. If the value of the entering variable x_j has just switched from one of its bounds to the other, then proceed directly to Step 2 of the next iteration. Otherwise, replace the leaving variable x_i by the entering variable x_j in the basis heading, and replace the leaving column of B by column $\boldsymbol{a}$.

Steps 2-5 in the revised simplex method that generate a new basic solution from a given basic solution are called a pivot.

In the following we solve the systems of equations in Steps 1 and 3 and derive explicit formulas for Y and V.

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

Let $Y = (y_1 \ldots, y_{n-1}, \gamma)$. Here $y_1 \ldots, y_{n-1}, \gamma$ are called potentials (or dual variables) and Y is called the potential vector. Each y_i, $i = 1, 2 \ldots, n-1$ is the potential associated with node i (or row i) and γ is the potential associated with the last row (delay constraint row) of A. The potential of node n is not in Y, and may be set to zero as we will see below.

Now consider

$$YB = \boldsymbol{c}_B. \tag{18}$$

This system of equations has n equations in n variables. We get the following from (18).

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

$$\begin{aligned} &y_i - y_j - \gamma d_{ij} = c_{ij}, \text{for } i \neq n \text{ and } j \neq n,\\ &y_i - \gamma d_{in} = c_{in}, \text{for } j = n,\\ &-y_j - \gamma d_{nj} = c_{nj}, \text{for } i = n.\end{aligned} \tag{19}$$

The last two equations in (19) can be obtained by setting the potential y_n of node n to zero in the first equation, namely, $y_i - y_j - \gamma d_{ij} = c_{ij}$. Thus in all the computations that follow we set $y_n = 0$.

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 the slack variable w is given by $r(w) = \gamma$. (Note: Since the column corresponding to the slack variable w is $(0, 0 \ldots, -1)^t$, we can get the reduced cost of w from the equation in 2) by setting $y_i = y_j = 0$, $c_{ij} = 0$, and $d_{ij} = 1$.
4) The reduced cost of a path p is defined as $r(p) = \sum_{(i,j)\in p^+} r(i,j) - \sum_{(i,j)\in p^-} r(i,j)$. Recall that p^+ and p^- denote the sets of forward and backward links on p, respectively.
5) Node n is called the root node.

It can be seen from (19) that for any link (i,j) in G_b

$$r(i,j) = y_j - y_i + \gamma d_{ij} + c_{ij} = 0. \tag{20}$$

From (20) we also have that for any path p from i to j and any cycle W in G_b

$$\begin{aligned} r(p) &= y_j - y_i + \gamma d(p) + c(p) = 0, \text{ and} \\ r(W) &= \gamma d(W) + c(W) = 0. \end{aligned} \tag{21}$$

Lemma 7: * If G_b contains a cycle W, then $\gamma = \frac{-c(W)}{d(W)}$; otherwise, $\gamma = 0$.

Once we have computed the value of γ, the other potentials y_i's can be calculated using (20) and selecting the path in G_b from node n (whose potential is 0) to node i. We summarize these steps as follows:

1) Set the potential of node n to zero.
2) Compute γ as in Lemma 7.
3) For each node i, let p_i be a path in G_b from node n to node i. If there are two paths in G_b due to the cycle, we can derive the same results no matter which one is selected.
4) Set $y_i = -\sum_{(u,v)\in p_i^+} c(e_{uv},\gamma) + \sum_{(u,v)\in p_i^-} c(e_{uv},\gamma)$, 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.

C. Solving the System $BV = \boldsymbol{a}_k$

We now show how to solve the system of equations $BV = \boldsymbol{a}_k$, where $\boldsymbol{a}_k$ is the column of A corresponding to the entering variable. In the following an entering link is called an in-arc and a leaving link is called an out-arc. We consider three cases:

(a) Basis graph 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.
(b) The basic variables are associated with n links and the entering variable is $e_k = (i,j)$.
(c) The basic variables are associated with n links and the entering variable is w.

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

Theorem 5: * a) If G_b contains no cycle and the entering variable is an in-arc $e_k = (i,j)$, then the vector $V = (v_1 \ldots, v_n)^t$ defined below is the desired solution to $BV = \boldsymbol{a}_k$, 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.

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

b) If G_b contains a cycle W and the entering variable is a link $e_k = (i,j)$, then $V = -V_p' + \frac{d(W')}{d(W)}V_0$, is the solution to $BV = \boldsymbol{a}_k$, where $d(W')$ and $d(W)$ are the delays of cycles W' and W, respectively and V_p' and 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 $V = \frac{V_0}{d(W)}$ is the solution to $BV = \boldsymbol{a}_k$, where 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,

$$B = \begin{pmatrix} H_{n-1,n-1} & \mathbf{0} \\ D_{1,n-1} & -1 \end{pmatrix},$$

where $H_{n-1,n-1}$ is associated with the $(n-1)$ links in G_b and $n-1$ nodes, and $D_{1,n-1}$ is the vector of $(n-1)$ components (corresponding to the basic variables except for w) of the last row of matrix 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 $V = (v_1 \ldots, v_n)^t$ defined as in the theorem solves the system $BV = \boldsymbol{a}_k$.

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

$$B = \begin{pmatrix} H_{n-1,n} \\ D_{1,n} \end{pmatrix},$$

where $H_{n-1,n}$ is associated with the n links and $n-1$ nodes, and $D_{1,n}$ is the vector of the n components of the last row of 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

$$\begin{pmatrix} H_{n-1,n} \\ D_{1,n} \end{pmatrix} V = \begin{pmatrix} \boldsymbol{h}_k \\ -d_{ij} \end{pmatrix}. \tag{22}$$

First, let us consider

$$H_{n-1,n}V = \boldsymbol{h}_k. \tag{23}$$

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

$$\exists V_0, H_{n-1,n}V_0 = 0. \tag{24}$$

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,

$$\exists V' = \begin{pmatrix} V_p' \\ 1 \end{pmatrix},\ (H_{n-1,n}, \boldsymbol{h}_k) \begin{pmatrix} V_p' \\ 1 \end{pmatrix} = 0. \tag{25}$$

So, $H_{n-1,n}(-V_p') = \boldsymbol{h}_k$.

Because rank($H_{n-1,n}$) = $n-1$, $-V_p' + uV_0$, $u \in R$ is the solution space of (23). We can compute u as follows.

$$D_{1,n}(-V_p' + u \cdot V_0) = -d_{ij}. \tag{26}$$

Since $D_{1,n}V_0 = -d(W)$ and $D_{1,n}(-V_p') + d_{ij} = d(W')$, we get from (26)

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

Therefore we have proven that $V = -V_p' + \frac{d(W')}{d(W)}V_0$ is the desired solution to $BV = \boldsymbol{a}_k$.

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 $V = \frac{V_0}{d(W)}$ is the solution to $BV = \boldsymbol{a}_k$. Here V_0 is defined by the cycle W in G_b. ■

VII. Initialization and Pivot Rules

A. Initialization

We first compute k minimum delay link-disjoint s-t paths using Suurballe's algorithm [16]. There is no feasible solution if the total delay of these paths is greater than Δ. Assume that this is not the case. A tree T' (not necessarily a spanning tree) rooted at t can be constructed from these paths by removing links incident with s to break cycles. Note that in T' every path from a node in T' to t is a directed path. Such a tree is called a directed tree rooted at node t [21]. We next obtain a directed spanning tree rooted at t and having T' as a subtree. We proceed as follows.

First condense or coalesce all the nodes in T' into a single node P. Then for the resulting network determine a directed spanning tree rooted at P with all links orientated away from P. Such a tree exists because of our assumption that there is a directed path from node s to each node in the network and similarly there is a directed path from each node to node t. The links of the directed tree selected as above and the links in T' together constitute a directed spanning tree T.

Assigning flow of 1 to all the links on the disjoint paths and flow of 0 to all other links, we obtain a basic solution represented by T.

Definition 3: [19] 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 $x_{uv} = 0$ (resp. $x_{uv} = 1$) is oriented away from (resp. toward) the root.

It can be easily verified that the initial spanning tree T selected as above is strongly feasible.

B. Pivot Rules and an Anti Cycling Strategy

For an efficient implementation of the revised simplex method, we want to avoid directed cycles in basic solutions. This can be achieved by the following pivot rule:

P1: Slack variable w assumes the highest priority in choosing the entering variable (Step 2 of the Revised Simplex Method).

Lemma 8: The slack variable w is eligible to enter the basis iff $\gamma < 0$.

Proof: Since the reduced cost of w is equal to γ, w is eligible to enter the basis iff $\gamma < 0$. ■

Lemma 9: Suppose the Pivot rule P1 is followed. If a directed cycle W is created in G_b during a pivot, then in the next pivot the slack variable w will enter the basis and a link on W will leave the basis.

Proof: Since W is a directed cycle, $c(W) \neq 0$ and $\gamma = -c(W)/d(W) < 0$. It follows that in the pivot after the directed cycle is created, w will enter the basis and the new basis graph will be a spanning tree. ■

A basic solution in which one or more basic variables assume zero values is called degenerate [22]. 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 (or infinite looping and non-convergence). Thus we need a strategy to avoid cycling.

There are several anticylcing strategies for general linear programming problems. Cunningham developed a strategy specifically designed for the network simplex method used for solving minimum cost flow problems. Since RELAX-TCSDP(k) has almost the same structure as the minimum cost flow problem except for the presence of one additional constraint imposed by the delay requirement, we examine if Cunningham's strategy can be adopted for RELAX-TCSDP(k). We show next that the transformation introduced on the CSDP(k) problem in Section IV indeed makes Cunningham's strategy suitable for avoiding cycling in the case of RELAX-TCSDP(k).

Lemma 10: 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 that results from the degenerate pivot. Consider now the flows on the links on a cycle. By Lemma 5 these flows are not 0 or 1. So if a link on a cycle were to leave the basis during a degenerate pivot, then after the pivot it would become nonbasic with flow of value 0 or 1. But that would contradict the fact 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 (Chapter 5.1.2 of [23]). Let T be the current G_b and $e = (u, v)$ and $e' = (u', v')$ be the out-arc and the in-arc, respectively. Let $T' = T - e + e'$ be the subgraph of the new basic variables. If e is not on the cycle in the current G_b , then the new potential vector Y' associated with T' can be obtained as follows [23].

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

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

The convergence part of the following theorem closely follows the proof of Theorem 19.1 in [22].

Theorem 6: 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 and its computational time complexity is pseudo-polynomial.

Proof: First observe that in any sequence of degenerate pivots, the value of every variable, in particular, the value of the slack variable will remain the same. Also if the slack variable is a basic variable then its value is nonzero; otherwise its value is zero. So during a given sequence of degenerate pivots, the slack variable will remain basic or nonbasic during the entire sequence of degenerate pivots. 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 values for G_b.

$$C(G_b) = \sum_{(u,v)\in E} c_{uv}x_{uv} \text{ and } W(G_b) = \sum_{u\in V}(y_t - y_u).$$

Consider two consecutive basic solutions G_b^i with 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 need to show that $W(G_b^{i+1}) < W(G_b^i)$.

Note that the in-arc $e = (u,v)$ still has flow equal to 0 or 1 in G_b^{i+1}. By Lemma 10, 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 if $x_e = 1$ and oriented away from t if $x_e = 0$, which implies that node t belongs to $G_b^i(v)$ (the component of $G_b^i - f$ containing v) if $x_e = 1$ and node t belongs to $G_b^i(u)$ if $x_e = 0$. The potentials with respect to G_b^{i+1} can be calculated using (27).

Then we have $W(G_b^{i+1})$ = $W(G_b^i) - |G_b(u)|r_{uv} < W(G_b^i)$, where $r_{uv} = c(e_{uv},\gamma) + y_v - y_u > 0$ if $x_e = 1$ or $W(G_b^{i+1}) = W(G_b^i) + |G_b(u)|r_{uv} < W(G_b^i)$, where $r_{uv} = c(e_{uv},\gamma) + y_v - y_u < 0$ if $x_e = 0$.

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

Thus we have proved that the simplex method when applied on RELAX-TCSDP(k) does not cycle if all the basic feasible solutions are strongly feasible.

We next establish the pseudo-polynomial time complexity of this method. We have

$$\begin{aligned} &W(G_b^i) - W(G_b^{i+1}) = |G_b(u)| \cdot |r_{uv}| \geq |r_{uv}|, \\ &|r_{uv}| = |c(e_{uv},\gamma) + y_v - y_u| = |c_{uv} + \gamma d_{uv} + y_v - y_u|. \end{aligned}$$

We proceed to show that

$$\begin{aligned} 0 &< |y_u - y_v - \gamma d_{uv} - c_{uv}| = |\gamma d(W') + c(W')| \\ &= \begin{cases} |c(W')|, \gamma = 0, \\ |c(W')d(W) - d(W')c(W)|/|d(W)|, \gamma \neq 0. \end{cases} \end{aligned} \tag{28}$$

Since e_{uv} is an in-arc, $|y_u - y_v - \gamma d_{uv} - c_{uv}| \neq 0$. To establish the equalities on the right hand side of (28) suppose that the new cycle W' in G_b is $e_1 e_2 \ldots e_k$ where $e_1 = e_{uv}$. Since all the links on W' except e_{uv} are in G_b, the reduced costs on all these links are 0. So we get from (21) that $|r_{uv}|$ = $|y_u - y_v - \gamma d_{uv} - c_{uv}| = |\gamma d(W') + c(W')|$. 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 equalities on the right side of (28).

Since $|r_{uv}| \neq 0$, we get from (28) that $|r_{uv}| \geq |1/d(W)| \geq 1/(nd_{max})$, where d_{max} is the maximum link delay.

Also, the inequality below follows from the fact that the potential of a node is the sum of the active costs of the links on the path from that node to node t (See Section VI-B).

$$W(G_b^i) = \sum_{u\in V}(y_t - y_u) \leq n^2(c_{max} + \gamma d_{max}),$$

where c_{max} is the maximum link cost.

If $\gamma \neq 0$, then by Lemma 7, $|\gamma| = |\frac{c(W)}{d(W)}| \leq nc_{max}$. Hence $W(G_b^i) \leq n^2(c_{max} + nc_{max}d_{max})$.

So the length of the sequence of degenerate pivots is bounded by a polynomial function of c_{max}, d_{max}, and n. Similarly, we can prove that the total number of non-degenerate pivots is also a polynomial function of m, n, c_{max}, and d_{max}. Pseudo polynomial complexity of the revised simplex method when applied on RELAX-TCSDP(k) follows since each pivot takes $O(m)$ steps [22]. ■

C. Leaving Variable

Now, we investigate how to find a leaving variable (out-arc) using Theorem 5. As before, let the cycle created by adding the in-arc be denoted by W' with its orientation defined as that of the in-arc.

We note that the reduced cost of the in-arc may be positive or negative. In the following we consider only the latter case. The former case can be treated in a similar way.

Case 1: Slack variable w is in the basic solution (the current G_b is a spanning tree and so $w > 0$). This corresponds to Theorem 5(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 $d(W') \neq 0$. These entries correspond to the links of W' of the current G_b or the slack variable w. The maximum value of t is constrained by $\boldsymbol{x}_B^* - tV \geq 0$, and the corresponding constraining variables (links or w) are eligible to leave the basis. If certain links are eligible to leave the basis then we select the one which keeps the new basic solution strongly feasible (to be discussed next). In this case w will continue to be in the basis. If w is eligible to leave the basis, we select it to leave the basis. In that case the new basis graph G_b' will have a cycle. The flow values (λ or $1-\lambda$) on the links on the cycle can be determined by the equation $d_{\boldsymbol{x}}(G_b') = \Delta$ because the slack variable w is nonbasic and has zero value.

Case 2: The basic solution consists of n links, i.e., there is a cycle W in G_b. The slack variable w is eligible to enter the basis if $\gamma < 0$. Then according to pivot rule P1, we let w enter the basis and shall select one of the links on W to leave the basis. The choice can be made according to the case (c) in Theorem 5.

If $\gamma \geq 0$, an entering link will create a new cycle W' when added to the current G_b. We need to consider different subcases that capture all possibilities (See Fig. 4). For each one of these subcases we can select the leaving variable using Theorem 5 (b) and Step 4 in the revised simplex method.

Now, we need to consider how to preserve the strong feasibility of the basic solutions. We define the join of a cycle in G_b as the node on the cycle that is closest to the node t in terms of hops. Without loss of generality, assume that the current basic solution is strongly feasible and consists of n

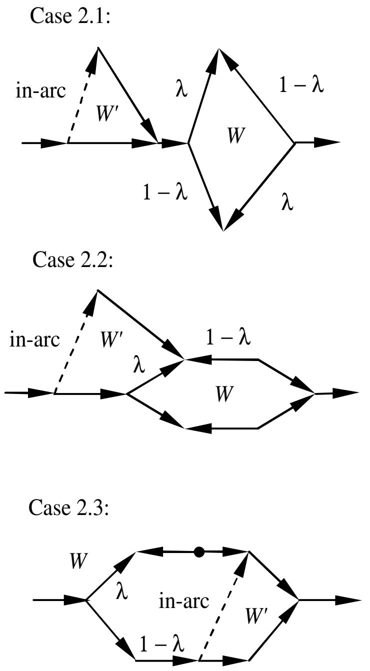


Fig. 4. Basic solution structures in case 2

links and that the leaving variable is a link f (other cases are trivial). Let $G_e = G_b + e$ be the network obtained by adding the in-arc e to G_b. Evidently, f is on some cycle $C \in \{W, W'\}$ in G_e. If $C = W'$, the orientation of C is chosen to agree (resp. disagree) with the orientation of e if $x_e = 0$ (resp. $x_e = 1$) in the current flow. If $C = W$, the orientation of C is defined such that $d(W')/d(W) < 0$ (resp. $d(W')/d(W) > 0$) if $x_e = 0$ (resp. $x_e = 1$), where the orientation of W' agrees with the direction of e. Starting from the join of C and traversing along the orientation of C, we choose the first link whose flow is 1 and whose direction agrees with the orientation of C or whose flow is 0 and whose direction disagrees with the orientation of C. This guarantees the strong feasibility of the resulting tree.

VIII. Simulation

We denote our algorithm as DISJOINT-NBS (NBS: Network Based Simplex method) and compare its performance with CPLEX and G-LARAC(k). We use three classes of network topologies: regular graphs $H_{k,n}$ (see Chapter 8, [21]), power-law out-degree graphs [24] and Waxman's random graphs [25]. For a graph $G(V, E)$, the nodes are labeled as $1, 2 \ldots, n = |V|$. Nodes $\lfloor n/2 \rfloor$ and n are chosen as the source and target nodes. The link costs and delays are randomly independently generated even integers in the range from 1 to 200. The delay bound is $1.2 \times k$ the delay of the minimum delay s-t paths in G. For regular graphs, $k = 4$. For random graphs and power-law graphs, $k = 2$. The results are shown in Fig. 5. In these figures, we use NBS to denote the DISJOINT-NBS algorithm and NBS-REGULAR to denote the running time of DISJOINT-NBS algorithm on regular graphs. Other labels can be interpreted in a similar manner. Experiments show that DISJOINT-NBS algorithm is faster than CPLEX and G-LARAC(k) on all the topologies. 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 graph is very large. So the running times of DISJOINT-NBS, G-LARAC(k), and CPLEX are close (but DISJOINT-NBS is still faster) for random graphs and power-law out-degree graphs.

Our simulation results in Tables I-III show that the delay of each path derived as in Theorem 4 deviates from the individual delay bound by a small fraction. Note that in these tables the second column specifies the delay bound on each path.

IX. Summary

In this paper we studied the CSDP(k) problem which is NP-hard. So our goal has been to design an efficient algorithm for constructing an approximate solution to this problem. Towards this end, we studied the LP relaxation of CSDP(k) problem using the revised simplex method of linear programming. This relaxed problem is an upper bounded LP problem. We have discussed several issues relating to an efficient implementation of our approach. We have shown that an approximate solution to the CSDP(k) problem can be extracted from an optimal solution to the relaxed problem. We have derived bounds on the quality of this solution with respect to the optimal solution. Our work can be considered as the study of the CSDP(k) problem from a primal perspective in contrast to the dual perspective employed in the G-LARAC(k) algorithm which is based on the algorithms in [11] and [16]. Simulation results demonstrate that our algorithm is slightly faster than both the G-LARAC(k) algorithm and the commercial quality CPLEX package in the case of random graphs and power-law out-degree graphs. On the other hand, for regular graphs our algorithm is much faster.

The GCSDP(k) problem defined in Section II requires that the delay of each individual path satisfies a specified bound, in contrast to the CSDP(k) problem where the constraint is on the total delay of all the k link-disjoint paths. We have shown in Theorem 2 that the LP relaxations of the two problems have the same optimal objective value. Thus, if one is interested in obtaining the optimal objective values of RELAX-GCSDP(k) and RELAX-CSDP(k) problems, then starting with the RELAX-CSDP(k) does not result in any loss of generality. However, the paths produced by the approximate solution derived from the optimal solution to RELAX-CSDP(k) may not satisfy the individual path delay requirements of the GCSDP(k) problem. Fortunately, our simulation results in Table I-III indicate that in most cases the individual delays of the paths produced starting from RELAX-CSDP(k) do not deviate in a significant way from the individual delay requirements of the GCSDP(k) problem.

If one were interested in studying the GCSDP(k) problem then the issue of finding feasible solutions to this problem will arise. The algorithm in the present paper may be used as a subroutine in a branch and bound scheme to find feasible solutions to the GCSDP(k) problem

One direction of further research is to develop approximation schemes for the CSDP(k) problem along the lines of the approximation algorithms given in [15] for the CSDP(2)

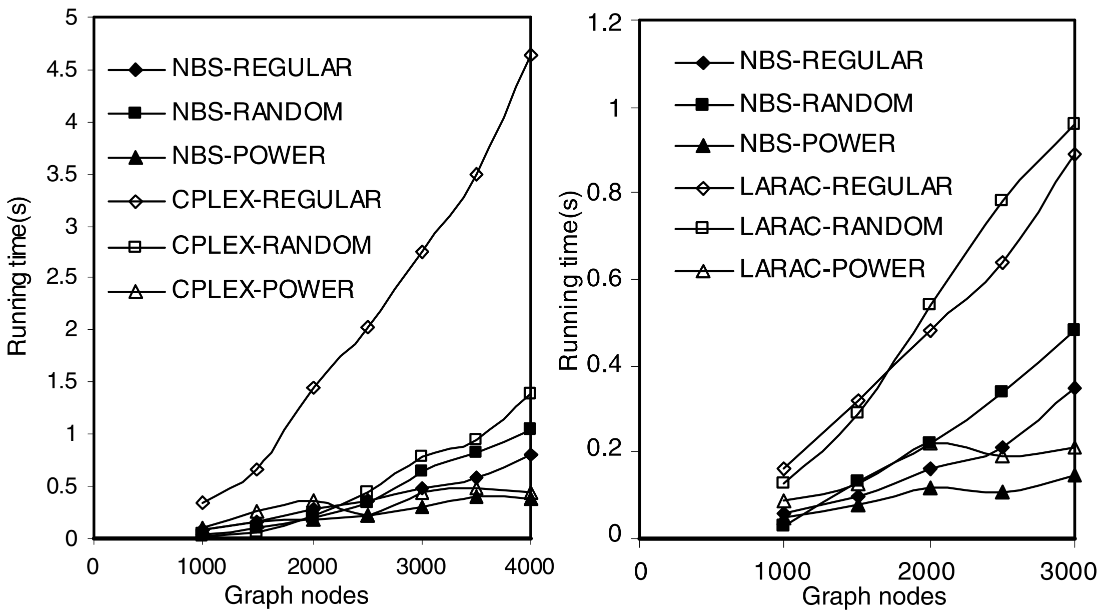


Fig. 5. Comparison of running times of DISJOINT-NBS algorithm and CPLEX and G-LARAC(k). The experiments were carried out on IBM Regatta p690 with AIX 5.1 OS and Power4 1.1 GHz CPU.

TABLE I
PATHS OBTAINED FROM THE OPTIMAL SOLUTION TO RELAX-TCSDP(2) APPLIED ON RANDOM GRAPHS

Graph Size(#Nodes)	**Delay Bound**	**Path-1(Cost, Delay)**	**Path-2(Cost, Delay)**
1000	1087	(1240, 1056)	(1536, 1082)
2000	601	(1548, 560)	(1344, 604)
3000	409	(1496, 368)	(1328, 428)

TABLE II
PATHS OBTAINED FROM THE OPTIMAL SOLUTION TO RELAX-TCSDP(2) APPLIED ON POWER-LAW GRAPHS

Graph Size(#Nodes)	**Delay Bound**	**Path-1(Cost, Delay)**	**Path-2(Cost, Delay)**
1000	109	(426, 110)	(372, 72)
2000	134	(352, 82)	(190, 172)
3000	206	(380, 206)	(254, 138)

TABLE III
PATHS OBTAINED FROM THE OPTIMAL SOLUTION TO RELAX-TCSDP(4) APPLIED ON REGULAR GRAPHS

Graph Size(#Nodes)	**Delay Bound**	**Path-1**	**Path-2**	**Path-3**	**Path-4**
1000	736	(2208, 686)	(2216, 686)	(2054, 764)	(1872, 782)
2000	1425	(3920, 1412)	(4168, 1424)	(4014, 1454)	(4198, 1406)
3000	2127	(6092, 2044)	((6126, 2104)	(5862, 2242)	(5702, 2110)

problem. Since the link-disjoint shortest paths problem is a special case of the minimum cost flow problem, it will be interesting to investigate if the ideas developed in this paper could be used to design efficient algorithms for the constrained minimum cost flow problem.

REFERENCES

[1] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, USA: Freeman, 1979.

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

[3] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math. of Oper. Res.*, vol. 17, no. 1, pp. 36–42, 1992.

[4] C. A. Phillips, "The network inhibition problem," in *STOC*, 1993, pp. 776–785.

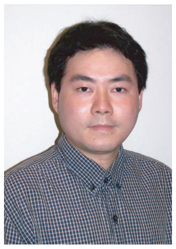
[5] D. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest paths problem," *Oper. Res. Letters*, vol. 28, pp. 213–219, 2001.

[6] 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.

[7] 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*, 2002, pp. 775–778.

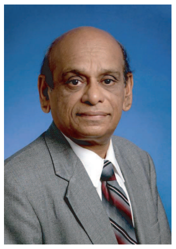
[8] G. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, pp. 293–310, 1980.

[9] K. Mehlhorn and M. Ziegelmann, “Resource constrained shortest paths,” in *ESA*, 2000, pp. 326–337.
[10] A. Jüttner, B. Szviatovszki, I. Mécs, and Z. Rajkó, “Lagrange relaxation based method for the QoS routing problem,” in *INFOCOM*, 2001, pp. 859–868.
[11] B. Blokh and G. Gutin, “An approximation algorithm for combinatorial optimization problems with two parameters,” *Australasian Journal of Combinatorics*, vol. 14, pp. 157–164, 1996.
[12] 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*, 2003, pp. 905–914.
[13] A. Jüttner, “On resource constrained optimization problems,” 2003, in review.
[14] Y. Xiao, K. Thulasiraman, and G. Xue, “The primal simplex approach to the QoS routing problem,” in *QSHINE*, 2004, pp. 120–129.
[15] A. Orda and A. Sprintson, “Efficient algorithms for computing disjoint QoS paths,” in *INFOCOM*, 2004, pp. 727–738.
[16] J. W. Suurballe, “Disjoint paths in a network,” *Networks*, vol. 4, no. 2, pp. 125–145, 1974.
[17] C.-L. Li, T. S. McCormick, and D. Simich-Levi, “The complexity of finding two disjoint paths with min-max objective function,” *Discrete Appl. Math.*, vol. 26, no. 1, pp. 105–115, 1990.
[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2003.
[19] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Networks Flows*. NJ, USA: Prentice-Hall, 1993.
[20] R. P. Bertsekas and J. N. Tsitsiklis, *Introduction to linear optimization*. Belmont, Massachusetts, USA: Athena Scientific, 1997.
[21] K. Thulasiraman and M. N. Swamy, *Graphs: Theory and algorithms*. New York: Wiley Interscience, 1992.
[22] V. Chvatalá, “A greedy heuristic for the set covering problem,” *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.
[23] D. P. Bertsekas, *Network optimization: continuous and discrete models*. Belmont, Massachusetts, USA: Athena Scientific, 1998.
[24] C. R. Palmer and J. G. Steffan, “Generating network topologies that obey power laws,” in *IEEE GLOBECOM*, 2000, pp. 434–438.
[25] B. M. Waxman, “Routing of multipoint connections,” *IEEE Journal on Selected Areas in Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.



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

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.