

Technical Report 07-EMIS-01

**Optimization Models and Algorithms for Minimizing the Maximum  
Link Utilization in OSPF Data Networks**

by

J. Kennington

and

A. Madhavan

{jlk, amadhava}@engr.smu.edu

Department of Engineering Management, Information, and Systems

School of Engineering

Southern Methodist University

Dallas, Texas 75275

June, 2007

Comments and criticisms from interested readers are cordially invited.

## **Abstract**

Dijkstra's simple shortest path algorithm provided the building block for link-state routing protocols in data networks. The open shortest path first (OSPF) protocol uses this algorithm in a distributed procedure in an attempt to balance the trade-off between routing implementation complexity and optimal network utilization. Today, powerful optimization software tools are available that can be used by a central processor in an attempt to improve network performance via implementation of better routing strategies. Two environments can be considered. In the first case, the routing software remains unchanged and the central processor simply determines optimal link weights that are supplied to the routers. Each router uses the link weights and Dijkstra's algorithm to determine its routing table. In the second environment, the routing software is modified to accept an optimal routing table. This investigation presents both node-arc and arc-path optimization models for both environments, along with an empirical analysis of all four models.

**Keywords:** internet protocol, OSPF, weight-setting, traffic engineering, routing protocols

## **Acknowledgement**

This research was partially supported by the Office of Naval Research under award number N00014-07-1-0192.

## 1. Introduction

The increase in demand for high bandwidth applications that involve video, audio streaming, and voice over IP pose new and interesting challenges for traffic engineers. One of the challenges is to distribute the traffic in the network such that network congestion is minimized as this may lead to delay in transferring messages from origin to destination. The Internet is composed of many thousands of *Autonomous Systems* (ASs) that are managed by a single institution. Computer messages sent from an origin to a destination may traverse several of these ASs with various delays along the way. Messages between computers connected to the Internet are divided into packets that flow through the data network. When a router receives a packet, it forwards the packet on one of its outbound links based on a routing table. ASs that run the popular OSPF protocol determine routing tables using a shortest path algorithm as a function of known integer link weights. If all link weights are set to one, then all routers forward packets along shortest-hop paths. Network managers can adjust the link weights in an attempt to improve the operation of a given autonomous system. Given the network topology and a set of link weights, each router can use Dijkstra's shortest path algorithm using itself as the root to determine a routing table.

### 1.1. Description of the Problem

The routers in an AS that adhere to the OSPF algorithm maintain a database that contains information about the state of the links in the network, and the weights associated with links. The weight assigned to a link may depend on one or a combination of the following metrics: bandwidth, load balance, delay, error probability, link length, and administrative cost. At Verizon they call the weights costs and the costs are set by a network administrator, Jarrar (2007). The status of links incident to a given router is transmitted in the network as *link state advertisement* packets so that the databases on all routers in the network are identical. A list of shortest paths from itself to all other routers is constructed by each router. The demands in the network are routed based on the paths generated from this list. One of the main advantages of such an algorithm is that it can be implemented in a distributed way.

Note that this strategy does not use the demand matrix in selecting paths for each  $(o, d)$  demand pair. Hence, it's possible that certain links become congested while others in the network may support very little traffic. The problem addressed in this investigation is to construct optimization models and algorithms that incorporate both the point-to-point demand values and link capacities such that the resulting routing tables produce the smallest maximum link utilization. The *weight-setting problem* as defined by Pioro and Medhi (2004) is as follows:

- (i) *Given*: the network topology, the link capacities, and a set of point-to-point demands,
- (ii) *Find*: weights and flows for each link,
- (iii) *Constraints*: the demands must be satisfied, the capacities cannot be violated, and at each router the total entering flow to a given destination is split equally among all out-going links that lie on the shortest paths to that destination.

Ideally this optimization problem would be solved at a central processor and the link weights would be communicated to each router. The routers could then each apply Dijkstra's shortest path algorithm to determine their routing tables. If there are multiple shortest-paths to a given destination, then all such paths are maintained in the routing tables. This is called the *equal-cost multi-path (ECMP) split rule* in OSPF networks (Pioro and Medhi 2004). This means that the routing table for router  $r$  may have multiple outbound links corresponding to a given destination  $d$  and the total flow out of router  $r$  with destination  $d$  must be split equally among all of these outbound links. Dijkstra's algorithm has been extended such that all shortest paths will be discovered. This strategy does not require any software modifications at the routers. That is, given the network topology and a set of costs (weights) each router will use a shortest path procedure to create its routing table. A potential disadvantage of this procedure is the creation of numerous long and circuitous paths for some (origin, destination) pairs.

An alternative strategy that can alleviate this disadvantage is to use the central processor to determine an optimal routing table for each router and then communicate this to each router as opposed to a set of weights. Of course this would require a software modification on each router. This investigation will examine strategies which do not

require a software modification at the routers as well as those that require some modification.

## **1.2. Survey of Literature**

Moy (1998) presents an introduction to routers, routing protocols, and a detailed discussion of the OSPF protocol. His manuscript discusses the design, the management, and multicasting extensions of the OSPF protocol. The basic mathematical model for optimal routing is presented by Fortz and Thorup (2000) where the objective is a piecewise-linear cost function for each arc (a link can be denoted by two oppositely directed arcs) which is based on the load on each arc. The manuscript discusses a dynamic weight-setting procedure for the OSPF problem and a hash table implementation to reduce the solution time of the procedure. The complexity of a general OSPF routing problem is discussed in detail by Juttner et al. (2000). The manuscript demonstrates that the OSPF weight-setting problem is *NP*-complete, and suggests heuristic methods for solving the problem.

The OSPF Design Guide (Cisco. 2001) presents a discussion of parameters used for link weight calculation, and provides a detailed account of the advantages of the OSPF protocol over traditional routing protocols. A detailed step-by-step implementation of the OSPF protocol, and simulation of an OSPF router is presented by Moy (2001), as well as the implementation of Dijkstra's shortest path algorithm to calculate the next hop in the network. The comparison of OSPF and IS-IS (Intermediate System-Intermediate System) protocols based on total bandwidth, memory access, and average arrival time is discussed in detail by Sharon (2001). The manuscript on optimal routing in shortest path data networks by Ramakrishnan and Rodrigues (2001) discusses the classification of routing problems. A combinatorial approach to solving the optimal shortest path routing problem is also presented in the manuscript.

Basu and Riecke (2001) discuss stability related issues in OSPF routing that include network convergence times, load on processors, and changes in the routing table in the event of a network failure. The article compares the performance of several heuristics with a genetic algorithm in terms of costs and maximum utilization. The article on traditional IP routing protocols by Fortz et al. (2002) provides an introduction

to traffic engineering concepts. The article summarizes the results of both OSPF and IS-IS optimizing weight techniques. Ericsson et al. (2002) propose a heuristic that uses genetic algorithm principles for solving the OSPF weight-setting problem. The arc-path formulation of the IP routing problem as a multi-commodity flow problem is discussed by Murphy et al. (2002). The manuscript also discusses the computation of splitting ratios for flows in the network in order to balance the traffic. Based on the value of the splitting ratio metric, flows are either split evenly along shortest paths, or routed along one shortest path.

Fortz and Thorup (2002) present a system of algorithms for managing sudden spikes in demand, and critical link failures by making limited weight changes, thereby avoiding hot spots in the network. The importance of a good set of initial weights and minimizing the changes in link weights when links are congested is presented in detail by Murphy et al. (2003). Simulation of OSPF and TCP protocols on large scale networks is discussed in detail by Yaun et al. (2003). The manuscript effectively tackles network scalability issues with optimal simulation techniques, and demonstrates performance of OSPF and TCP simulation models. Fortz and Thorup (2004) propose a local search heuristic for the weight-setting procedure. The manuscript describes the strict Tabu Search methodology implemented using hash tables employed in exploring the neighborhood and changes in arc-weights. A similar approach is described to ensure even split in arc flows for OSPF routing. An elaborate empirical analysis and comparison of different OSPF weight setting procedures versus local search heuristics is presented. In addition, theoretical proofs of NP-hardness of OSPF routing for minimizing the maximum utilization is given. Buriol et al. (2005) present a heuristic that combines the genetic algorithm proposed by Ericsson et al. (2002) with a local improvement procedure. The local procedure works on reduced neighborhoods and examines the effect of increasing weights on a subset of arcs. A procedure for fast updates of flows on arcs due to the change in weights and a dynamic shortest path algorithm are presented. Sridharan et al. (2005) propose a heuristic for achieving near optimal solutions for existing OSPF/IS-IS networks, and also describes possible applications of this technique. The manuscript explains a selective next hop approach which is computed based on minimizing the maximum ratio of assigned flow to optimal

flow over all hops. This heuristic does not guarantee integer link weights. Buriol et al. (2007) present an evolutionary algorithm for the OSPF weight setting procedure. In addition, the manuscript considers the survivability of a network design by addressing router and link failures. The comparison of network costs for various generations of the evolutionary algorithm is also presented in the manuscript. An excellent survey of route optimization in IP networks may be found in Rexford (2006).

### **1.3. Contribution of the Investigation**

Pioro and Medhi (2004) give an elegant mixed-integer optimization model to determine an optimal set of integer weights that minimizes the maximum utilization of any single link. Since the delay is closely related to link utilization, optimal weights will tend to minimize delay and improve operation from the user's point-of-view. The Pioro-Medhi weight-setting model uses a node-arc formulation. Unfortunately, this model suffers from three critical deficiencies. The first is that only small problems are solvable using current optimization software such as CPLEX version 10.0. The second is that path flows that satisfy the equal-cost multi-path split rule are not easily determined from the node-arc model. The third is that an optimal solution may result in a large number of shortest paths some of which may be quite circuitous. In our investigation, an arc-path formulation and column generation algorithm for the OSPF weight-setting problem is proposed that addresses the deficiencies of the Pioro-Medhi model. The major challenge is to enforce the equal-cost multi-path split rule required by the OSPF protocol. Our column generation scheme generates paths at each iteration until an optimality criterion is satisfied. While our column generation scheme is a substantial improvement over the Pioro-Medhi model, only small problem instances are computationally tractable using our algorithm. The OSPF protocol was not designed with optimization in mind and consequently efficient algorithms for large problem instances of the OSPF weight-setting problem continues to elude the research community (Rexford 2006). Our proposal for conquering this challenge is to relax the equal-cost multi-path split rule. This will require some modification of the OSPF protocol and a slight modification of router software. Optimization models and an empirical analysis demonstrate the merit of the revised strategy.

## 2. A Strategy Using Existing Router Software

IP routers have software that use a shortest-path algorithm to determine a routing table. The inputs are the routers, links, and link weights of a graph and the output is a routing table. Since at a point in time the routers and links are fixed, the outside central processor will need to provide only the link weights. Current routers use 16 bit unsigned integers (1 – 65535) as link weights.

### 2.1. A Node-Arc Model

A mixed integer programming formulation of the OSPF weight-setting problem is presented in this Section. Let  $V$  and  $L$  denote the sets of routers and links. For this investigation, links are undirected and arcs are directed. Each link is represented by a pair of arcs. The set of arcs is denoted by  $E = \{(i, j), (j, i) : (i, j) \in L\}$ . Let  $h^{od}$  be the demand volume to be routed from origin  $o$  to destination  $d$  where  $o, d \in V$  and let  $D$  be the set of  $(o, d)$  demand pairs such that  $h^{od} > 0$ . Let  $H$  denote the sum of demand values i.e.  $H = \sum_{(o,d) \in D} h^{od}$  and let  $\bar{D} = \{d : (o, d) \in D\}$ . The model is a multi-commodity flow problem with  $|\bar{D}|$  commodities. The capacity of arc  $(i, j) \in E$  is given by  $c_{ij}$ . The requirement for commodity  $d \in \bar{D}$  at router  $i$ , denoted by  $g_i^d$  is defined below:

$$g_i^d = \begin{cases} - \sum_{(o,d) \in D} h^{od} & , \quad i = d, d \in \bar{D} \\ h^{id} & , \quad \forall (i, d) \in D, d \in \bar{D} \\ 0 & , \quad \text{otherwise} \end{cases}$$

Let  $w_{ij}$  denote the weight on arc  $(i, j)$  and  $x_{ij}^d$  denote the flow on arc  $(i, j)$  with destination  $d \in \bar{D}$ . The variable  $z_{id}$  denotes the distance from router  $i$  to router  $d$  on the shortest path to router  $d \in \bar{D}$ . Let  $y_{id}$  be the common value of the flow assigned to arcs

originating at  $i$  and contained in shortest paths from  $i$  to  $d \in \bar{D}$ . For the example illustrated in Figure 1,  $z_{3e} = 3$ ,  $z_{6e} = 1$ ,  $z_{2e} = 3$ ,  $y_{3e} = \eta$ ,  $y_{6e} = \sigma$ , and  $y_{2e} = \theta$ .

-----  
Figure 1: About Here  
-----

Let the binary decision variable  $u_{ij}^d = 1$  if arc  $(i, j)$  belongs to the shortest path to router  $d$ ; and  $u_{ij}^d = 0$ , otherwise. Let  $\mathbf{U}$  denote the maximum arc utilization. Using the above notation the node-arc formulation for the OSPF weight-setting problem is now presented. The objective is,

$$\text{minimize } \mathbf{U} \quad (1)$$

The first set of constraints ensures that demand at router  $d \in \bar{D}$  is satisfied and ensures the conservation of flow at each router.

$$\sum_{(v,j) \in E} x_{vj}^d - \sum_{(i,v) \in E} x_{iv}^d = g_v^d \quad \forall v \in V, \forall d \in \bar{D} \quad (2)$$

The arc capacity constraints are

$$\sum_{d \in \bar{D}} x_{ij}^d \leq c_{ij} \quad \forall (i, j) \in E \quad (3)$$

The third set of constraints are needed to calculate the maximum arc utilization

$$\sum_{d \in \bar{D}} x_{ij}^d \leq c_{ij} \mathbf{U} \quad \forall (i, j) \in E \quad (4)$$

The fourth set of constraints ensures that all flow from router  $i$  to destination  $d$  on shortest paths to  $d$  are equal.

$$0 \leq y_{id} - x_{ij}^d \leq (1 - u_{ij}^d)H \quad \forall d \in \bar{D}, \forall (i, j) \in E \quad (5)$$

The following set of constraints prevents flow on any arc that is not in a shortest path for demand router  $d$ .

$$x_{ij}^d \leq u_{ij}^d H \quad \forall d \in \bar{D}, \forall (i, j) \in E \quad (6)$$

The sixth set of constraints guarantee that the weights on arcs  $(i, j)$  and  $(j, i)$  are equal.

$$w_{ij} = w_{ji} \quad \forall (i, j) \in L \quad (7)$$

The next set of constraints ensures that the lengths of shortest paths for a  $(o, d)$  pair are equal.

$$0 \leq z_{jd} + w_{ij} - z_{id} \leq (1 - u_{ij}^d)H \quad \forall d \in \bar{D}, \forall (i, j) \in E \quad (8a)$$

$$1 - u_{ij}^d \leq z_{jd} + w_{ij} - z_{id} \quad \forall d \in \bar{D}, \forall (i, j) \in E \quad (8b)$$

The boundary conditions are:

$$1 \leq w_{ij} \leq 65535 \text{ and integer} \quad \forall (i, j) \in E \quad (9a)$$

$$x_{ij}^d \geq 0 \quad \forall d \in \bar{D}, \forall (i, j) \in E \quad (9b)$$

$$y_{id} \geq 0 \quad \forall i \in V, \forall d \in \bar{D} \quad (9c)$$

$$z_{id} \geq 0 \quad \forall i \in V, \forall d \in \bar{D} \quad (9d)$$

$$u_{ij}^d \in \{0,1\} \quad \forall d \in \bar{D}, \forall (i, j) \in E \quad (9e)$$

A version of this model first appeared in Pioro and Medhi (2004).

## 2.2. The Equal Cost Multi-Path Split Procedure

Routers that implement the OSPF protocol, use a shortest path strategy to route packets from an origin to a destination in a data network. Given a set of link weights, each router

-----  
Figures 2, 3: About Here

Table 1: About Here  
-----

The OSPF protocol supports multiple shortest paths between pairs of routers. For router 2 in Figure 2, the following multiple paths (2-4, 2-1-4; 2-5, 2-3-5; 2-3-6, 2-5-6, 2-3-5-6) will be incorporated into the routing table as illustrated in Table 2. The equal cost multi-path split procedure requires that traffic at a router be split equally among outgoing links in the shortest paths to a given destination. A traffic demand of 10 Mbps between routers 2 and 6 would be allocated as illustrated in Figure 4. At router 2, 5 Mbps are assigned to each of the links (2,3) and (2,5). At router 3, 2.5 Mbps are assigned to links (3,5) and (3,6). Note that this is not equivalent to assigning one third of the traffic to each of the three shortest paths (2-3-6, 2-5-6, 2-3-5-6).

-----  
Figure 4: About Here

Table 2: About Here  
-----

Consider a network  $G[V,L]$  with  $V$  routers and  $L$  links. For each  $(i, j) \in L$ , the reverse arc  $(j, i)$  has weight  $w_{ji}$  equal to  $w_{ij}$ . A modified Dijkstra's algorithm is implemented to identify the multiple shortest paths between an origin-destination pair as presented in Figure 5. Multiple shortest paths imply that a router  $j \in V$  can have more than one parent router say  $i_1, i_2 \in V$ . In this instance, the output parameter from the modified Dijkstra's procedure,  $\rho_{ji_1}$  and  $\rho_{ji_2}$  will be set to 1 where  $(i_1, j)$  and  $(i_2, j) \in E$ . Given the set  $T$  of arcs in all shortest paths between a pair of routers ( $o$  and  $d$ ), the equal flow distribution procedure assigns flow on the arcs in set  $T$  such that the flows from a parent router to its child routers are equal. This procedure is presented in Figure 6. Combining the modified Dijkstra procedure and the equal flow distribution procedure results in the equal cost multi-path split procedure. With the inputs  $(V, E, D, w, h, c)$ , defined in the ECMP split procedure, at a given instance of time the maximum utilization for this network using the OSPF protocol will be the  $\mathbf{U}$  calculated in Figure 7.

-----  
 Figures 5, 6, 7: About Here  
 -----

### 2.3. Arc-Path Model

The principle behind our *arc-path model* is that we seek to distribute the flow on the shortest paths for a given  $(o, d)$  pair such that the arcs from a common router on these paths have equal flow. Suppose we have three paths for a given  $(o, d)$  pair. All flow may be assigned to a single path, flow may be split among two shortest paths of equal length, or flow may be split among all three equal length shortest paths. For a three path problem there are seven cases for each  $(o, d)$  demand pair as illustrated in Table 3. For our model, each demand-case combination is assigned a unique pattern number. Selection of pattern 4 from Table 3 implies that the unique shortest path for  $(o_1, d_1)$  is  $p_3$ . That is, paths  $p_1$  and  $p_2$  must be longer than  $p_3$ . Selection of pattern 13 implies that paths  $p_5$  and  $p_6$  are of equal length and are shorter than  $p_4$ . For our arc-path model, the critical decision is the selection of a pattern for each  $(o, d)$  demand pair.

-----  
 Table 3: About Here  
 -----

#### 2.3.1. Sets, Constants, and Variables

Let  $n$  denote the maximum number of paths that can be initially generated for each  $(o, d) \in D$ . Let  $P_{od}$  denote the set of paths for demand pair  $(o, d) \in D$  and let  $\mathbf{D} = \bigcup_{(o,d) \in D} P_{od}$ . Let  $J_p$  denote the arcs in path  $p \in \mathbf{D}$  and, let  $R_{od}$  denote the set of patterns for each  $(o, d)$  pair. The number of patterns for a demand pair  $(o, d) \in D$  can be computed using the formula  $2^{|P_{od}|} - 1$ . Hence, the total number of patterns is  $\sum_{(o,d) \in D} (2^{|P_{od}|} - 1)$ . Let  $\mathbf{R} = \bigcup_{(o,d) \in D} R_{od}$  and, let  $M_p$  denote the set of patterns that include path  $p \in \mathbf{D}$ . For each  $r \in R_{od}$  and each  $(i, j) \in E$ , we use the equal cost multi-path split rule to determine the flow values

$b_{rij}$ . The structure of the flow pattern matrix  $b_{rij}$  is illustrated in an example from the network in Figure 8. Let paths computed using the least hop criterion be 1-3-4, 1-2-4, 1-2-3-4, 2-4, and 2-3-4 corresponding to paths  $p_1, p_2, p_3, p_4,$  and  $p_5$ . The structure of  $b_{rij}$  for demand values of 10 on origin-destination pair, (1,4), and 12 on (2,4) is illustrated in Table 4. In the case of two demand pairs  $(o_1, d)$ , and  $(o_2, d)$  with the same destination, the equal cost multi-path split rule will allow only certain combinations of patterns to be selected. The flows on the outgoing arcs from a router on the shortest paths to destination  $d$  have to be equal irrespective of the different origin routers. The flow values in patterns  $r_2 \in R_{14}$  and  $r_9 \in R_{24}$  from Table 4, contradict the equal flow rule as illustrated in Figure 9a. Likewise, patterns  $r_3$  and  $r_{10}$  are invalid as illustrated in Figure 9b. Let  $C$  denote the set of pairs of patterns, both of which cannot be selected. The set  $C$  will be used to prevent the selection of pairs of patterns for which the equal-flow multi-path split rule will not be satisfied.

-----  
 Figures 8, 9: About Here

Table 4: About Here  
 -----

Let  $l_p$  be the length of path  $p \in \mathbf{D}$ . Let  $k_{od}$  denote the length of the shortest path(s) for  $(o, d) \in D$ . The binary variable  $q_r$  is 1 if pattern  $r \in \mathbf{R}$  is selected; and 0, otherwise. Let  $\psi_p$  be 1 if path  $p \in \mathbf{D}$  is selected; and 0, otherwise.

### 2.3.2. Mixed Integer Linear Programming Model

The objective is to minimize the maximum arc utilization.

$$\text{minimize } \mathbf{U}$$

The first set of constraints ensures that only one pattern is selected for each  $(o, d)$  pair.

$$\sum_{r \in R_{od}} q_r = 1 \quad \forall (o, d) \in D$$

The capacity constraints are as follows:

$$\sum_{r \in \mathbf{R}} b_{rij} q_r \leq c_{ij} \mathbf{U} \quad \forall (i, j) \in E$$

The third set of constraints calculates the length of each path.

$$\sum_{(i,j) \in J_p} w_{ij} = l_p \quad \forall p \in \mathbf{D}$$

The fourth set of constraints guarantee that the weights on arcs  $(i, j)$  and  $(j, i)$  are equal.

$$w_{ij} = w_{ji} \quad \forall (i, j) \in L$$

The next set of constraints gives the relationship between patterns and paths.

$$\psi_p = \sum_{r \in M_p} q_r \quad \forall p \in \mathbf{D}$$

The following set of constraints ensure that if a pattern is chosen, then the length of the path(s) used by that pattern is(are) shortest. Let  $\mathbf{G}$  denote the maximum length of a path. Since the maximum weight is 65535,  $\mathbf{G}$  is set to  $65535|L|$ .

$$-\mathbf{G}(1 - \psi_p) \leq l_p - k_{od} \leq \mathbf{G}(1 - \psi_p) \quad \forall (o, d) \in D, \forall p \in P_{od}$$

The seventh set of constraints ensure that if a pattern is selected then the corresponding path with no flow is longer than the shortest path(s) for each demand pair

$$(1 - \psi_p) \leq l_p - k_{od} \quad \forall (o, d) \in D, \forall p \in P_{od}$$

The following set of constraints eliminates incompatible patterns.

$$q_m + q_r \leq 1 \quad \forall (m, r) \in C$$

The boundary conditions are given below:

$$1 \leq w_{ij} \leq 65535 \text{ and integer} \quad \forall (i, j) \in E$$

$$l_p \geq 0 \quad \forall p \in \mathbf{D}$$

$$\psi_p \geq 0 \quad \forall p \in \mathbf{D}$$

$$k_{od} \geq 0 \quad \forall (o, d) \in D$$

$$q_r \in \{0,1\} \quad \forall r \in \mathbf{R}$$

## 2.4. The Arc-Path OSPF Weight-Setting Algorithm

The algorithm iterates between a problem instance of the arc-path model and the ECMP procedure as presented in Figure 10. Initially an instance of the arc-path model using at most  $n$  paths per  $(o, d)$  pair is solved. The optimal weights are passed as inputs to the ECMP procedure. Using these weights shortest paths are determined and flows are allocated to these paths using the equal flow distribution rule. If the maximum utilization for the arc-path model, denoted by  $U_{Arcpath}$  and the maximum utilization from the ECMP procedure, denoted by  $U_{ECMP}$  are within a small tolerance, then the algorithm terminates. Otherwise, ECMP utilizes some paths not included in the arc-path model. The new paths from the ECMP split procedure are then included in the set of paths  $P_{od}$  where  $(o, d) \in D$ . New patterns are generated for the additional paths and included in  $R_{od}$  where  $(o, d) \in D$  and this procedure repeats until the stopping criterion is satisfied.

-----  
Figure 10: About Here  
-----

## 2.5. Empirical Analysis and Comparison

Forty-five problem instances for both the node-arc model and the arc-path OSPF algorithm were solved using AMPL ([www.ampl.com](http://www.ampl.com)) and CPLEX 10.0 ([www.cplex.com](http://www.cplex.com)). All test cases were run on a Microway, 2.2GHz dual core AMD dual processor with 16 GB of RAM. Default parameters were used except for a few large problem instances of the arc-path OSPF algorithm in which the integer programming probe option was set to 3. Default settings for these problems did not yield an optimal solution within the time limit. A mipgap of 5% and a time limit of one hour was used in all test cases. In order to improve computational efficiency, the weights have an upper bound of 64 rather than the theoretical limit of 65535. A random seed value was used to generate different instances of the model. The five graphs illustrated in Figures 11 through 16 were used to determine the network topology.

-----  
Figures 11, 12, 13, 14, 15,

16: About Here  
-----

Problems P110 – P160, Q210 – Q260, R310 – R360, and S410 – S460 presented in Table 5 correspond to the topologies illustrated in Figures 11 through 14, respectively. The demand pairs, demand values, and link capacities are randomly generated for each problem instance. The average node degree is computed using the formula  $2|L| / |V|$ . The column labeled “LP Relaxation” denotes the optimal objective value of the linear programming version of the node-arc model. Note that the gaps between the linear programming version and the integer program are fairly large. The columns labeled “Paths Used” indicate the number of shortest paths along which the flows were distributed. The columns labeled “Max Hops” indicate the maximum number of hops in a path with a flow value greater than 0. The column labeled “% Diff (Node-Arc vs. Arc-Path)” in Table 5 indicates the percentage difference between the optimal objective values for the two models. Twenty problems had a 0% difference indicating identical solutions with both models. A deviation greater than 0% implies that the node-arc model produced a better solution while a negative deviation implies the converse. For the first eighteen problems in Table 5, the optimal solutions were approximately the same and the solution times were small except for Q260 which required almost an hour for the node-arc model. Two of the largest problems, S440 and S450, yielded quite different objective values. The run with the node-arc model on S440 timed out with a very poor solution. The run with the arc-path model on S450 terminated with an inferior solution. The solution for the node-arc model used an excessive number of paths (17) to satisfy the five demand pairs. Demand pair (1,6) used 5 unique paths to satisfy 27 Mbps of demand. The corresponding arc-path model used a single path. Even though the node-arc model produced a lower maximum utilization, it required the use of a large number of circuitous paths to do so. Hence, most network operators will prefer the solution to the arc-path model. The two solutions are given in Appendix A. The last row in Table 5 labeled “Total” clearly illustrates that for the forty-five problem instances the total CPU time for the node-arc problem was approximately 2 hours and that for the arc-path algorithm was approximately 9 minutes. The total number of paths used by the arc-path problem instances were 10% less than the node-arc instances.

-----  
 Table 5: About Here  
 -----

Problems R510 – R600, and U410 – U460 presented in Table 6 are derived from larger networks in Figures 15 and 16, respectively. As illustrated in Table 6 the node-arc model does not yield a feasible solution within the one hour time limit when the number of demand pairs is at least 5. Clearly, when the node-arc model can be solved, it produces the least maximum utilization. As illustrated in Appendix A, this solution may not be a preferred one. In addition, it is limited in the size of problem that can be solved. Hence, we believe that the arc-path algorithm produces the best solution for this operational problem. Its major advantages are that larger problems are computationally tractable and the user can control the paths that are available for assignment. Its disadvantage is that it cannot guarantee that a least utilization solution been found as illustrated with S450, and most of the problems corresponding to Figure 15. For two of the forty five problems, the optimal utilization produced by the arc-path model exceeded the optimal value by more than 10%. The node-arc model for these two problems used a total of twenty three paths versus only nine for the arc-path algorithm.

-----  
Table 6: About Here  
-----

Although, our arc-path procedure appears to be superior to the node-arc procedure, it is still limited to small problem instances. Most network operators would like to solve problems with hundreds of nodes and links, (Allen 2004, Buriol 2005, Jarrar 2007). The next section presents our strategy for large scale problems.

### 3. A Strategy Using Modified Router Software

If router manufacturers are willing to make some minor changes to their software, then improved network performance can be achieved. We propose two simple modifications. First, routers are not required to adhere to the equal-flow multi-path split rule; and second, a routing table can be passed to a router in contrast to link weights from which a routing table is constructed. Optimal routing tables would be determined by a central processor and sent to each individual router. Packets destined for a given demand router would still be routed over multiple paths, but the equal split rule would not be imposed. Rather than a  $\frac{1}{3}$  split over three paths, one path could be allocated 50% of the traffic and the other two 25% each. Minor software changes are required to implement this new rule.

With the relaxed rules described above, the new optimization problem is called the *OSPF routing table problem*. Both node-arc and arc-path models are presented for this problem and both are empirically evaluated on real-world problems. The node-arc model for minimizing the maximum arc utilization is simply (1), (2), (4), and (9b) defined previously.

An arc-path version of the OSPF routing table problem is also very simple. In addition to the sets  $D$ ,  $E$ ,  $P_{od}$ , and  $\mathbf{D}$  defined in Section 2, a set  $Q_{ij} \subset \mathbf{D}$  with  $(i, j) \in E$  is needed. Let the paths that contain arc  $(i, j) \in E$  be denoted by  $Q_{ij}$ . Let  $f_p$  denote the flow on path  $p \in \mathbf{D}$ . The objective is to minimize the maximum utilization as stated in the previous model, minimize  $\mathbf{U}$

The first set of constraints ensures that the flow on paths satisfies the demand from origin  $o$  to destination  $d$

$$\sum_{p \in P_{od}} f_p = h^{od} \quad \forall (o, d) \in D$$

The next set of constraints relates the path flows to the arc utilization

$$\sum_{p \in Q_{ij}} f_p \leq c_{ij} \mathbf{U} \quad \forall (i, j) \in E$$

The boundary conditions are  $f_p \geq 0$  for all  $p \in \mathbf{D}$ .

Note that this model assumes *flow bifurcation*. That is, flow for a given commodity can be routed using several paths and each path can carry any fraction of the total demand,  $h^{od}$ . Of course, this model can be extended to the non-bifurcated case where the demand for each commodity must be assigned to a single path.

Recall, some of the larger problems could not be solved within the one hour time limit using the arc-path column generation algorithm (see Table 6). The requirement of the ECMP split rule makes the problems quite difficult to solve. Our empirical analysis for the new models may be found in Table 7. Problems EU, US, NA, N0, and V are real world networks (see Birkan (2006), Birkan et al. (2007), Jarrar (2004), and Allen (2004) ). The demand values for each origin-destination pair were randomly generated within the range [10, 80] for problems EU, US, NA, and N0. For network V, the demand values were from a fixed set of values in the range [5,486]. The capacities on the links were also randomly generated for problems EU, US, and NA. For problems N0 and V the capacities on the links were assigned from the sets {622, 2488} and {435, 1991, 89580}, respectively. Problems V1, V2, V3, and V4 are modifications of V in which the  $(o,d)$  demand pair values and capacities are randomly generated. Note that in all problem instances except V4 the percentage difference in utilization between the node-arc version and the arc-path version is less than 10%. Since the maximum hop count is large and computation time is excessive for the node-arc version, we believe that operators will prefer the arc-path version.

-----  
 Table 7 : About Here  
 -----

The networks in Table 7 are sparse with an average node degree between 3 and 4. In order to study the performance of the 2 modified models on denser networks, random graphs with an average node degree between 10.5 and 19.4 were generated (Sampathkumar 2007). Of the 20 instances, there were 4 problem instances in which the utilization of the node-arc version was superior to the arc-path version and in 5 other problem instances the utilization of the arc-path version was superior. The arc-path version clearly outperformed the node-arc version in terms of run times and the number of hops. The maximum number of hops in the arc-path version was 5 whereas in the node-arc version it was 17.

-----  
 Table 8 : About Here  
 -----

## **4. Summary and Conclusions**

For the environment in which router software remains unchanged, our arc-path based procedure is clearly superior to the node-arc based procedure that appears in the literature. However, the equal-flow multi-path restriction is so onerous, that only small problem instances can be solved. For this environment, a heuristic procedure such as the one described by Buriol et al. (2007) appears best. However, if the manufacturers of routers are willing to make some minor modifications to their software, then our new arc-path model can be used to obtain excellent solutions to the routing problem. In addition our proposed strategy allows for the control of path hop counts. For all other strategies that use the existing OSPF protocol, circuitous routing with large hop counts may be selected.

## References

Allen D., Personal Communication, October 13, 2004, Verizon.

Basu, A., J. Riecke. 2001. Stability issues in OSPF routing. *Proc ACM SIGCOM Computer Communications Review* **31** 225–236.

Birkan, G. 2006. Design Procedures and Restoration Schemes for Backbone Transport Networks: Optimization-Based Models, Heuristic Algorithms, and Unavailability Computations, A Dissertation, School of Engineering, Southern Methodist University, Dallas, Texas.

Birkan, G., J. Kennington, E. Olinick, A. Ortynski, G. Spiride. 2007. Design Strategies for Meeting Unavailability Targets Using Dedicated Protection in DWDM Networks *Journal of Lightwave Technology* **25** 1120-1129.

Buriol, L., M. Resende, C. Ribeiro, M. Thorup. 2005. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks* **46** 36-56.

Buriol, L., M. Resende, M. Thorup. 2007. Survivable IP network design with OSPF routing. *Networks* **49** 51-64.

Cahn, R. 1998. *Wide Area Network Design – Concepts and Tools for Optimization*. Morgan Kaufmann Publishers, San Francisco, CA.

Cisco, 2001, OSPF Design Guide. <http://www.cisco.com/warp/public/104/1.html>, retrieved August 19, 2005.

Ericsson, M., M. Resende, P. Pardalos. 2002. A genetic algorithm for the weight setting problem in OSPF routing. *Journal of Combinatorial Optimization* **6** 299–333.

Fortz, B., M. Thorup. 2000. Internet traffic engineering by optimizing OSPF weights. *Proc. 19<sup>th</sup> IEEE Conf. Computer Communications*, Tel Aviv, Israel. 519–528.

Fortz, B., M. Thorup. 2002. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications* **20** 756–767.

Fortz, B., J. Rexford, M. Thorup. 2002. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine* **40** 118–124.

Fortz, B., M. Thorup. 2004. Increasing internet capacity using local search. *Computational Optimization and Applications* **29** 13–48.

Fourer, R., D. Gay, B. Kernighan. 2003. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole-Thomson Learning, Pacific Grove, CA.

Holmberg, K., D. Yuan. 2004. Optimization of internet protocol network design and routing. *Networks* **43** 39-53.

Jarrar, S. 2004. Formulation and Evaluation of Optimization Models for MPLS Traffic Engineering with QOS Requirements. A Praxis, School of Engineering, Southern Methodist University, Dallas, TX.

Jarrar, S. 2007. Personal Communication, February 27, 2007, Verizon.

Juttner, A., A. Szentesi, J. Harmatos, M. Pioro. 2000. On solvability of an OSPF routing problem. *Proc.15<sup>th</sup> Nordic Teletraffic Seminar*, Lund, Sweden. 1–9.

Moy, J. 1998. *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley, Reading, MA.

Moy, J. 2001. *OSPF Complete Implementation*. Addison-Wesley, Boston, MA.

Murakami, K., H. Kim. 1997. Comparative study on restoration schemes of survivable ATM networks. *Proc. IEEE Conf. Computer Communications*, Kobe, Japan. 345–352.

Murphy, J., R. Harris, R. Nelson. 2002. Traffic engineering using OSPF weights and splitting ratios, *IFIP Interworking*, Perth, Australia. 277–287.

Murphy, J., R. Suryasaputra, X. Doan, R. Nelson, R. Harris. 2003. Link congestion avoidance using weight setting in an MPLS network – a simple LP approach, *ATNAC*, Melbourne, Australia. (CD ROM - ISBN: 0-646-42229-4).

Parkhurst, W. 1998. *Cisco Router OSPF: Design & Implementation Guide*. McGraw-Hill Inc., New York, NY.

Pioro, M., D. Medhi. 2004. *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, San Francisco, CA.

Ramakrishnan, K., M. Rodrigues. 2001. Optimal routing in shortest-path data networks. *Bell-Labs Technical Journal* **6** 117–138.

Rexford, J. 2006. Route optimization in IP networks. Chapter 25 in *Handbook of Optimization in Telecommunications*. Editors M. Resende, P. Pardalos, Springer, New York, New York.

Sampathkumar B., Personal Communication, May 14, 2007, Southern Methodist University, Dallas, TX.

Sharon, O. 2001. Dissemination of routing information in broadcast networks: OSPF versus IS-IS, *IEEE Network* **15** 56–65.

Sridharan, A., R. Guerin, C. Diot. 2005. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Transactions on Networking* **13** 234–247.

Walpole, R., R. Myers. 1985. *Probability and Statistics for Engineers and Scientists*. Macmillan, New York, NY.

Yaun, G., D. Bauer, H. Bhutada, C. Carothers, M. Yuksel, S. Kalyanaraman. 2003. Large-scale network simulation techniques: Examples of TCP and OSPF models. *ACM SIGCOMM Computer Communications Review* **33** 27–41.

## **Appendix A:**

The node-arc model solution and arc-path solution for problem S450 are illustrated in Figure A1 and Table A1. The numbers on the arcs indicate the arc-flows. The demand pairs are (2,4), (1,6), (5,3), (8,3), and (7,3) with demand values of 11, 27, 23, 21, and 24, respectively. The total number of paths used to route the above demands were 17 using the node-arc approach and just 6 using the arc-path approach. We believe that most network operators will prefer the solution given by the arc-path model.

-----  
Figure A1: About Here

Table A1: About Here  
-----

**Table 1: Routing Table for Example in Figure 2**

Destination	Root Router 1	Root Router 2	Root Router 3	Root Router 4	Root Router 5	Root Router 6
	Send To	Send To	Send To	Send To	Send To	Send To
1	-	1	2	1	4	5
2	2	-	2	2	2	5
3	2	3	-	2	3	3
4	4	4	2	-	4	5
5	4	3	5	5	-	5
6	4	3	6	5	6	-

**Table 2: OSPF Routing Table for Router 2**

Destination	Send To	Shortest Paths
1	1	2-1
2	-	-
3	3	2-3
4	1 or 4	2-1-4 2-4
5	3 or 5	2-3-5 2-5
6	3 or 5	2-3-6 2-5-6 2-3-5-6

**Table 3: Flow Distribution Pattern for 2 (o,d) Pairs with 3 Paths**

Path Pattern	Demand ( $o_1, d_1$ )			Demand ( $o_2, d_2$ )		
	$p_3$	$p_2$	$p_1$	$p_6$	$p_5$	$p_4$
$r_1$	0	0	1			
$r_2$	0	1	0			
$r_3$	0	1	1			
$r_4$	1	0	0			
$r_5$	1	0	1			
$r_6$	1	1	0			
$r_7$	1	1	1			
$r_8$				0	0	1
$r_9$				0	1	0
$r_{10}$				0	1	1
$r_{11}$				1	0	0
$r_{12}$				1	0	1
$r_{13}$				1	1	0
$r_{14}$				1	1	1

**Table 4: Flow Distribution Matrix -  $b_{rij}$**

$r \backslash i, j$	(1,2)	(1,3)	(2,3)	(2,4)	(3,4)
$r_1$	0	10	0	0	10
$r_2$	10	0	0	10	0
$r_3$	5	5	0	5	5
$r_4$	10	0	10	0	10
$r_5$	5	5	5	0	10
$r_6$	10	0	5	5	5
$r_7$	5	5	2.5	2.5	7.5
$r_8$	-	-	0	12	0
$r_9$	-	-	12	0	12
$r_{10}$	-	-	6	6	6

**Table 5: Minimizing the Maximum Utilization - Node-Arc Vs. Arc-Path**

( Time Limit Equals 1 Hour, Demand Range is [10,30], Capacity Range is [50,100], Optimality Gap equals 5% )

Name	Nodes	Links	Avg. Node Deg.	# of (o,d) Demand Pairs	Node-Arc Model				Arc-Path OSPF Algorithm					% Diff (Node-Arc vs. Arc-Path) ((2)-[1])/[2]	
					LP Relaxation		IP Version		Total Paths	Paths Used	Max Hops	Max Link Utilization [2]	CPU Time (hh:mm:ss)		
					Max Link Utilization	Paths Used	Max Hops	Max Link Utilization [1]							CPU Time (hh:mm:ss)
P110	6	9	3	1	0.146	4	5	0.155	00:00:01	5	3	3	0.155	00:00:01	0.0%
P120	6	9	3	2	0.122	6	5	0.146	00:00:01	16	6	5	0.146	00:00:01	0.0%
P130	6	9	3	3	0.117	5	4	0.154	00:00:01	17	5	4	0.154	00:00:01	0.0%
P140	6	9	3	4	0.269	8	3	0.345	00:00:40	23	7	3	0.345	00:00:01	0.0%
P150	6	9	3	5	0.453	10	3	0.525	00:00:01	32	10	3	0.525	00:00:01	0.0%
P160	6	9	3	6	0.311	7	2	0.362	00:00:01	38	6	2	0.362	00:00:03	0.0%
Q210	7	11	3.14	1	0.106	4	3	0.138	00:00:01	6	3	3	0.138	00:00:01	0.0%
Q220	7	11	3.14	2	0.113	8	5	0.185	00:00:01	20	8	5	0.185	00:00:28	0.0%
Q230	7	11	3.14	3	0.178	7	2	0.203	00:00:01	18	7	2	0.203	00:00:01	0.0%
Q240	7	11	3.14	4	0.174	8	4	0.225	00:00:01	28	6	3	0.234	00:00:01	4.0%
Q250	7	11	3.14	5	0.267	8	4	0.396	00:00:02	36	8	3	0.396	00:00:07	0.0%
Q260	7	11	3.14	6	0.343	7	5	0.350	00:54:01	46	9	3	0.350	00:00:01	0.0%
R310	8	13	3.25	1	0.071	6	6	0.082	00:00:01	13	6	6	0.082	00:00:01	0.0%
R320	8	13	3.25	2	0.184	8	4	0.215	00:00:01	15	7	3	0.215	00:00:01	0.0%
R330	8	13	3.25	3	0.171	9	3	0.196	00:00:08	22	9	3	0.196	00:00:01	0.0%
R340	8	13	3.25	4	0.256	8	6	0.311	00:00:01	33	8	5	0.311	00:00:07	0.0%
R350	8	13	3.25	5	0.386	6	3	0.424	00:00:04	39	6	3	0.424	00:00:01	0.0%
R360	8	13	3.25	6	0.286	12	3	0.332	00:00:14	49	12	3	0.332	00:00:05	0.0%
S410	9	15	3.33	1	0.101	3	6	0.111	00:00:01	14	3	6	0.111	00:00:01	0.0%
S420	9	15	3.33	2	0.138	5	7	0.180	00:00:01	16	4	3	0.188	00:00:01	4.0%
S430	9	15	3.33	3	0.125	7	4	0.222	00:00:34	27	8	4	0.222	00:00:43	0.0%
S440	9	15	3.33	4	0.276	11	4	0.443 <sup>1</sup>	01:00:00	35	10	4	0.318	00:00:21	-39.4%
S450	9	15	3.33	5	0.288	17	6	0.330	00:00:03	40	6	4	0.465	00:00:02	29.1%
S460	9	15	3.33	6	0.290	10	4	0.339	00:01:26	48	10	4	0.339	00:06:21	0.0%
<b>Total</b>						184	101		01:57:26	636	167	87		00:08:32	

1 Terminated due to 1 hour limit with mip gap = 37.6%

**Table 6: Minimizing the Maximum Utilization - Node-Arc Vs. Arc-Path**

( Time Limit Equals 1 Hour,Demand Range is [10,30], Capacity Range is [100,150], Optimality Gap equals 5% )

Name	Nodes	Links	Avg. Node Deg.	# of (o,d) Demand Pairs	Node-Arc Model					Arc-Path OSPF Algorithm					% Diff (Node-Arc vs. Arc-Path)
					LP Relaxation	IP Version				Total Paths	Paths Used	Max Hops	Max Link Utilization [2]	CPU Time (hh:mm:ss)	
					Max Link Utilization	Paths Used	Max Hops	Max Link Utilization [1]	CPU Time (hh:mm:ss)						
R510	10	17	3.4	1	0.107	5	5	0.111	00:00:01	4	2	3	0.118	00:00:01	6.0%
R520	10	17	3.4	2	0.069	6	6	0.076	00:00:01	8	3	3	0.114	00:00:01	33.3%
R530	10	17	3.4	3	0.103	11	5	0.117	00:00:26	12	5	3	0.119	00:00:01	1.6%
R540	10	17	3.4	4	0.114	11	5	0.162	00:05:53	18	8	4	0.177	00:00:01	8.4%
R550	10	17	3.4	5	0.139	8	5	0.165	01:00:00	33	8	5	0.151	00:00:16	-9.3%
R560	10	17	3.4	6	0.151	13	5	0.225	01:00:00	24	13	4	0.230	00:00:01	2.1%
R570	10	17	3.4	7	0.175	24	6	0.208	01:00:00	32	14	4	0.231	00:00:01	10.0%
R580	10	17	3.4	8	0.212	-	-	NFS <sup>2</sup>	01:00:00	36	13	5	0.260	00:00:01	-
R590	10	17	3.4	9	0.207	15	4	0.247	01:00:00	36	11	4	0.252	00:00:01	1.8%
R600	10	17	3.4	10	0.181	18	4	0.242	01:00:00	61	15	5	0.242	00:00:23	0.0%
<b>Total</b>						111	45		06:06:21	264	92	40		00:00:47	
U410	11	23	4.18	10	0.138	-	-	NFS <sup>2</sup>	01:00:00	54	23	4	0.197	00:00:00	-
U415	11	23	4.18	15	0.174	-	-	NFS <sup>2</sup>	01:00:00	80	23	3	0.199	00:01:20	-
U420	11	23	4.18	20	0.272	-	-	NFS <sup>2</sup>	01:00:00	143	25	4	0.329	00:04:44	-
U425	11	23	4.18	25	0.281	-	-	NFS <sup>2</sup>	01:00:00	120	36	4	0.337 <sup>3</sup>	00:02:11	-
U430	11	23	4.18	30	0.333	-	-	NFS <sup>2</sup>	01:00:00	139	39	4	0.366 <sup>3</sup>	00:02:03	-
U435	11	23	4.18	35	0.340	-	-	NFS <sup>2</sup>	01:00:00	147	40	3	0.429 <sup>3</sup>	00:00:41	-
U440	11	23	4.18	40	0.328	-	-	NFS <sup>2</sup>	01:00:00	165	43	3	0.438	00:02:45	-
U445	11	23	4.18	45	0.470	-	-	NFS <sup>2</sup>	01:00:00	185	52	4	0.497 <sup>3</sup>	00:45:22	-
U450	11	23	4.18	50	0.579	-	-	NFS <sup>2</sup>	01:00:00	203	55	4	0.600 <sup>3</sup>	00:06:08	-
U455	11	23	4.18	55	0.476	-	-	NFS <sup>2</sup>	01:00:00	225	59	3	0.586 <sup>4</sup>	01:00:00	-
U460	11	23	4.18	60	0.613	-	-	NFS <sup>2</sup>	01:00:00	242	62	3	0.768 <sup>5</sup>	01:00:00	-

2 No feasible solution with 1 hour time limit

3 Convergence achieved with CPLEX probe option set to 3

4 Terminated due to 1 hour limit with mip gap = 18.8%

5 Terminated due to 1 hour limit with mip gap = 20.19%

**Table 7: Minimizing the Maximum Utilization - Large Networks**

( Time Limit Equals 1 Hour, Optimality Gap equals 5% )

Name	Nodes	Links	Avg. Node Deg.	# of (o,d) Demand Pairs	Demand Range	Capacity Range	Node-Arc Modified Procedure			Arc-Path Modified Procedure # of paths per (o,d) = 8					% Diff (Node-Arc vs. Arc-Path)  ((1)-[2])/[1]
							Max Hops	Max Link Utilization [1]	CPU Time (hh:mm:ss)	Total Paths	Paths Used	Max Hops	Max Link Utilization [2]	CPU Time (hh:mm:ss)	
EU	18	35	3.9	100	[10,80]	[1000,1500]	6	0.232	00:00:01	800	107	4	0.232	00:00:01	0.0%
US	28	42	3.0	250	[10,80]	[1000,1500]	11	0.881	00:00:01	2000	251	8	0.881	00:00:01	0.0%
NA	36	67	3.7	250	[10,80]	[1000,1500]	11	0.407	00:00:01	2000	265	6	0.407	00:00:01	0.0%
N0	20	31	3.1	300	[10,80]	From Set <sup>6</sup>	8	0.653	00:00:01	2400	313	6	0.653	00:00:01	0.0%
V	236	467	4.0	923	From Set <sup>7</sup>	From Set <sup>8</sup>	17	0.343	00:14:22	7384	966	7	0.343	00:00:02	0.0%
V1	236	467	4.0	1000	[10,50]	[10000,100000]	19	0.026	00:09:12	8000	1004	8	0.026	00:00:01	0.0%
V2	236	467	4.0	2000	[10,50]	[10000,100000]	21	0.045	01:00:00 <sup>9</sup>	16000	2012	8	0.047	00:00:01	-4.3%
V3	236	467	4.0	3000	[10,50]	[10000,100000]	21	0.097	01:00:00 <sup>9</sup>	24000	3010	9	0.104	00:00:01	-6.6%
V4	236	467	4.0	4000	[10,50]	[10000,100000]	22	0.063	01:00:00 <sup>9</sup>	32000	4004	9	0.122	00:00:01	-93.2%

6 From the set {622, 2488}

7 From the set {5, 6, 32, ..., 486}

8 From the set {435, 1991, 89580}

9 Terminated due to 1 hour time limit

**Table 8: Minimizing the Maximum Utilization - Dense Random Networks**

( Time Limit Equals 1 Hour, Demand Range is [100,500], Optimality Gap equals 5% )

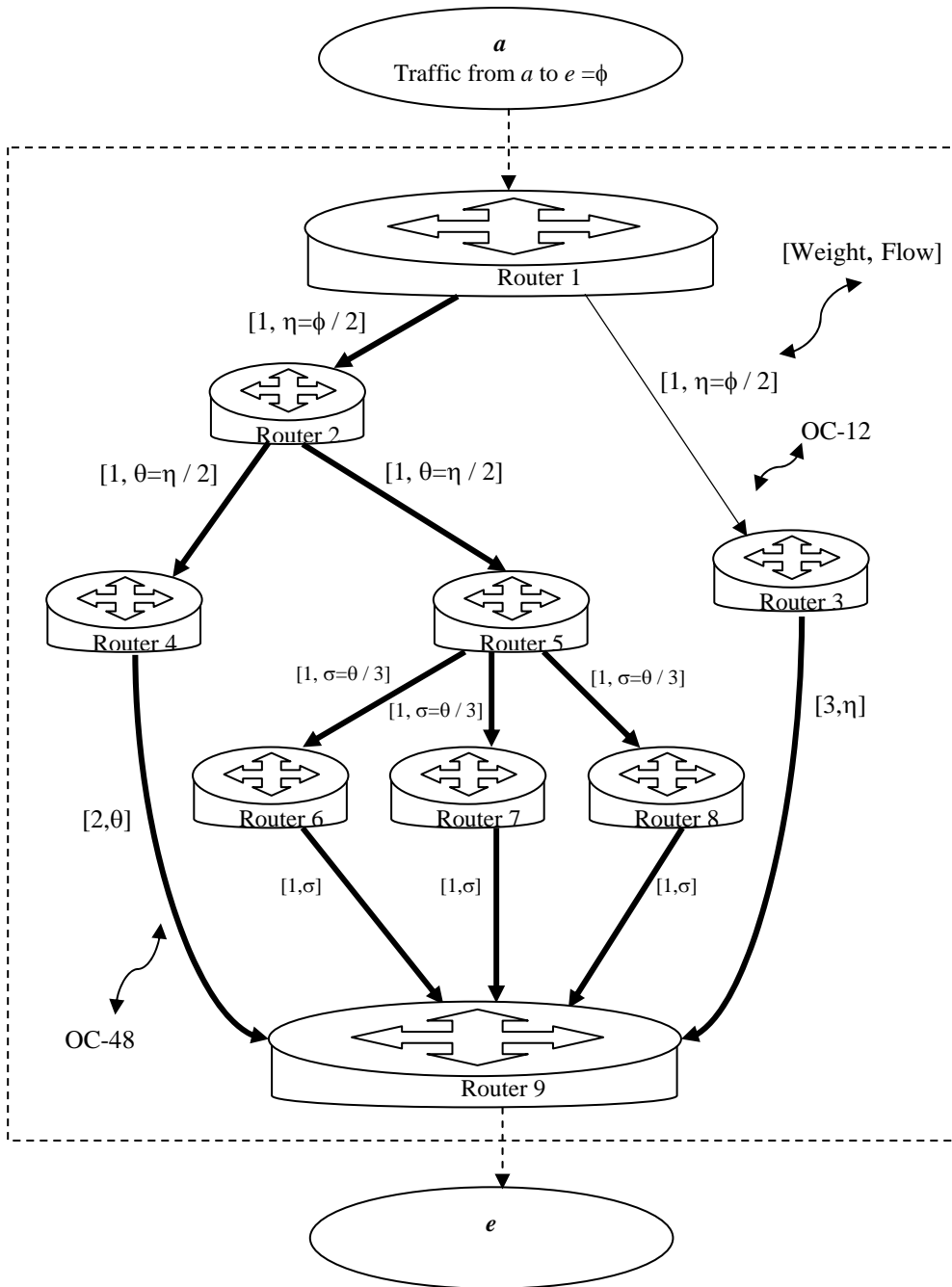
Name	Nodes	Links	Avg. Node Deg.	# of (o,d) Demand Pairs	Capacity Range	Node-Arc Modified Procedure			Arc-Path Modified Procedure # of paths per (o,d) = 8					% Diff (Node-Arc vs. Arc-Path) ((1)-[2])/[1]
						Max Hops	Max Link Utilization [1]	CPU Time (hh:mm:ss)	Total Paths	Paths Used	Max Hops	Max Link Utilization [2]	CPU Time (hh:mm:ss)	
R110	35	184	10.5	200	[1000,50000]	8	0.099	00:00:04	1600	229	5	0.109	00:00:01	-10.2%
R120	35	184	10.5	400	[1000,50000]	11	0.141	00:00:22	3200	435	5	0.216	00:00:01	-53.5%
R130	35	184	10.5	600	[1000,50000]	10	0.155	00:01:10	4800	661	5	0.155	00:00:01	0.0%
R140	35	184	10.5	800	[1000,50000]	8	0.203	00:03:55	6400	878	5	0.203	00:00:01	0.0%
R150	35	184	10.5	1000	[1000,50000]	8	0.247	00:04:11	8000	1069	5	0.247	00:00:01	0.0%
R210	45	332	14.8	350	[1000,50000]	8	0.049	00:02:21	2800	453	4	0.049	00:00:01	0.0%
R220	45	332	14.8	400	[1000,50000]	8	0.084	00:51:07	5600	875	4	0.084	00:00:01	0.0%
R230	45	332	14.8	600	[1000,50000]	9	0.379	01:00:00 <sup>10</sup>	8400	1241	4	0.103	00:00:02	72.7%
R240	45	332	14.8	800	[1000,50000]	7	1.027	01:00:00 <sup>10</sup>	11200	1607	4	0.137	00:00:03	86.7%
R250	45	332	14.8	1000	[1000,50000]	9	0.288	00:29:09	14000	1825	4	0.288	00:00:01	0.0%
R310	55	534	19.4	500	[1000,50000]	9	0.072	00:01:58	4000	556	4	0.072	00:00:01	0.0%
R320	55	534	19.4	1000	[1000,50000]	12	0.166	00:05:45	8000	1064	5	0.166	00:00:01	0.0%
R330	55	534	19.4	1500	[1000,50000]	10	0.219	00:20:29	12000	1580	5	0.219	00:00:01	0.0%
R340	55	534	19.4	2000	[1000,50000]	10	0.570	00:07:50	15993	2011	4	0.570	00:00:01	0.0%
R350	55	534	19.4	2500	[1000,50000]	12	1.151	00:08:08	19993	2501	5	1.151	00:00:01	0.0%
R410	65	381	11.7	800	[10000,50000]	12	NFS <sup>11</sup>	01:00:00 <sup>10</sup>	6400	1135	4	0.029	00:00:01	-
R420	65	381	11.7	1600	[10000,50000]	10	0.067	01:00:00 <sup>10</sup>	12800	1896	4	0.059	00:00:01	11.7%
R430	65	381	11.7	2400	[10000,50000]	9	0.060	01:00:00 <sup>10</sup>	19200	3047	3	0.067	00:01:15	-12.2%
R440	65	381	11.7	3200	[10000,50000]	10	0.040	01:00:00 <sup>10</sup>	25600	3481	4	0.116	00:00:02	-188.6%
R450	65	381	11.7	4000	[10000,50000]	17	0.199	01:00:00 <sup>10</sup>	32000	4465	4	0.127	00:00:28	36.3%

10 Terminated due to 1 hour time limit

11 No Feasible Solution

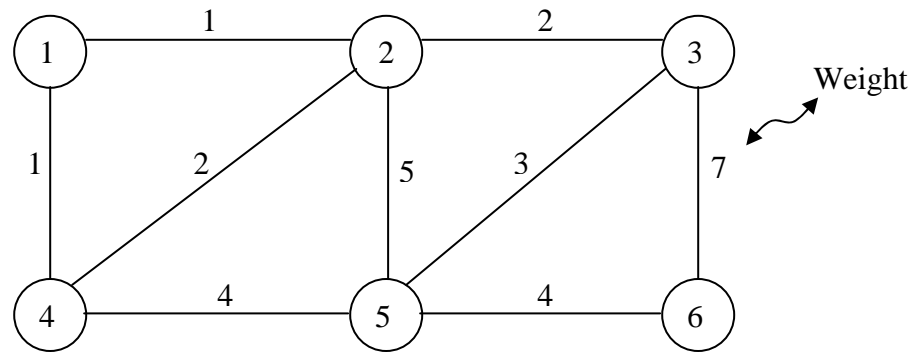
**Table A1: Shortest Paths Used in Problem S450**

<i>(o, d)</i> Pairs	Node-Arc		Arc-Path	
	Paths Used	Flow on Paths	Paths Used	Flow on Paths
(2,4)	2-4	11	2-4	11
(1,6)	1-2-4-6 1-3-5-6 1-3-4-5-6 1-2-4-5-6 1-3-2-4-5-6	11.25 4.5 4.5 2.25 4.5	1-3-4-6	27
(5,3)	5-3 5-4-3 5-4-2-3	11.5 5.75 5.75	5-3	23
(8,3)	8-6-5-3 8-6-4-3 8-6-5-4-3 8-6-4-2-3 8-6-5-4-2-3	5.25 5.25 2.625 5.25 2.625	8-6-4-3	21
(7,3)	7-5-3 7-6-5-4-3 7-9-8-6-4-2-3	8 8 8	7-5-3 7-8-6-4-3	12 12

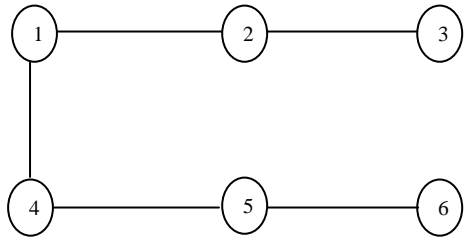


- Path 1:  $a \rightarrow$  Router 1  $\rightarrow$  Router 3  $\rightarrow$  Router 9  $\rightarrow e$ , Weight = 4
- Path 2:  $a \rightarrow$  Router 1  $\rightarrow$  Router 2  $\rightarrow$  Router 4  $\rightarrow$  Router 9  $\rightarrow e$ , Weight = 4
- Path 3:  $a \rightarrow$  Router 1  $\rightarrow$  Router 2  $\rightarrow$  Router 5  $\rightarrow$  Router 6  $\rightarrow$  Router 9  $\rightarrow e$ , Weight = 4
- Path 4:  $a \rightarrow$  Router 1  $\rightarrow$  Router 2  $\rightarrow$  Router 5  $\rightarrow$  Router 7  $\rightarrow$  Router 9  $\rightarrow e$ , Weight = 4
- Path 5:  $a \rightarrow$  Router 1  $\rightarrow$  Router 2  $\rightarrow$  Router 5  $\rightarrow$  Router 8  $\rightarrow$  Router 9  $\rightarrow e$ , Weight = 4

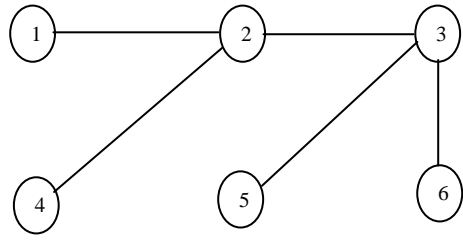
**Figure 1: Percentage Flow Distribution in a OSPF Network**



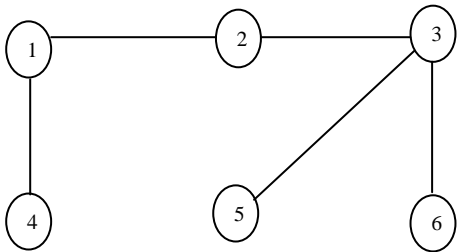
**Figure 2: Six Router Example Network**



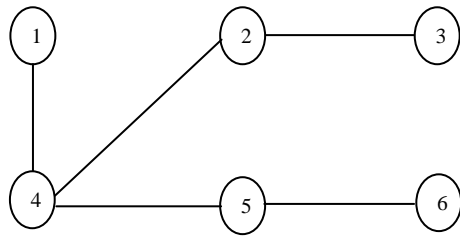
**a: Root Router 1**



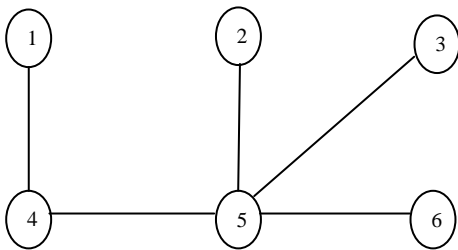
**b: Root Router 2**



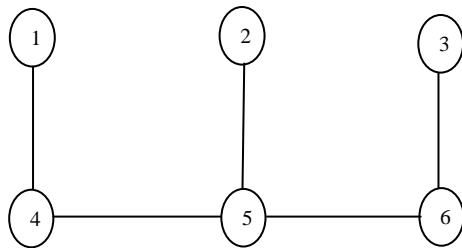
**c: Root Router 3**



**d: Root Router 4**

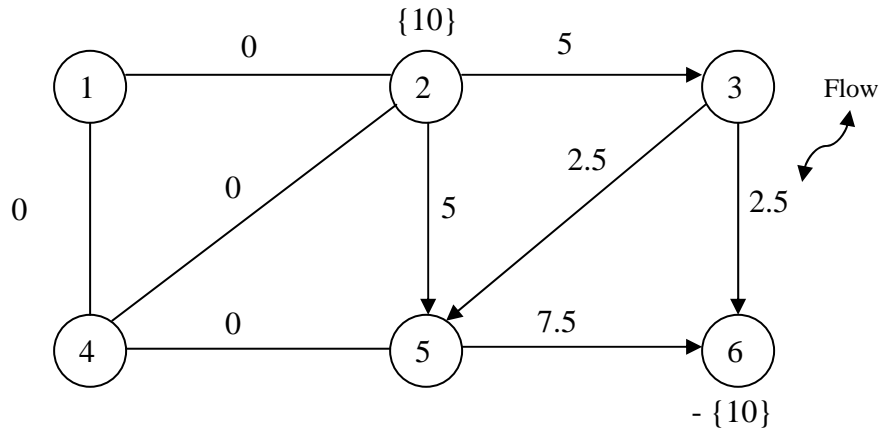


**e: Root Router 5**



**f: Root Router 6**

**Figure 3: Shortest Path Trees for the Network in Figure 2**



**Figure 4: Flow Distribution in ECMP Procedure**  
 (Shortest paths are 2-3-6, 2-3-5-6, and 2-5-6)

**procedure** Modified Dijkstra( $o, d, V, E, w, \rho$ )

Inputs:  $o, d, V, E, w$

/\*  $V, E$  denotes the network with  $V$  routers and  $E$  arcs \*/

/\*  $(o,d)$  denotes the origin and destination routers \*/

/\*  $w_{ij}$  denotes the weight on arc  $(i,j) \in E$  \*/

Output:  $\rho$

/\*  $\rho_{ji}$  set to 1 implies that router  $i$  is a parent of router  $j$  \*/

```

begin
   $count \leftarrow 0, BigL \leftarrow (|E| \max \{w_{ij} : (i, j) \in E\}) / 2$ 
   $\forall m \in V$  do
     $count \leftarrow count + 1, list_{count} \leftarrow m, dist_m \leftarrow BigL$ 
     $\forall v \in V$  do  $\rho_{mv} \leftarrow 0$ 
  end do
   $dist_o \leftarrow 0$ 
  repeat
    /* Find router with smallest distance */
     $small \leftarrow 2BigL$ 
     $\forall m \in \{1, \dots, count\}$  do
      if  $dist_{list_m} < small$  then
         $savem \leftarrow m, small \leftarrow dist_{list_m}$ 
      end if
    end do
     $s \leftarrow list_{savem}$ 
    if  $s = d$  then break
     $list_{savem} \leftarrow list_{count}, count \leftarrow count - 1$ 
    /* Process router  $s$  */
     $\forall (s, j) \in E$  do
      if  $dist_s + w_{sj} = dist_j$  then  $\rho_{js} \leftarrow 1$ 
      if  $dist_s + w_{sj} < dist_j$  then
         $dist_j \leftarrow dist_s + w_{sj}$ 
         $\forall m \in V$  do  $\rho_{jm} \leftarrow 0$ 
         $\rho_{js} \leftarrow 1$ 
      end if
    end do
  end repeat
end

```

**Figure 5: Multiple Shortest Path Algorithm**

**procedure** Equal Flow Distribution( $o, d, V, E, h, T, f$ )

Inputs:  $o, d, V, E, h, T$

/\*  $V, E$  denotes the network with  $V$  routers and  $E$  arcs \*/

/\*  $(o,d)$  denotes the origin and destination routers \*/

/\*  $h^{od}$  denotes the demand value for the demand pair  $(o,d)$  \*/

/\*  $T$  denotes the set of all arcs in the shortest paths for demand pair  $(o,d)$  \*/

Output:  $f$

/\*  $f_{ij}$  denotes the flow on arc  $(i,j) \in E$  \*/

```

begin
   $\forall (i, j) \in V \times V$  do
     $M_{ij} \leftarrow 0, f_{ij} \leftarrow 0$ 
  end do
   $\forall (i, j) \in T$  do  $M_{ij} \leftarrow 1$ 
   $\forall j \in V$  do  $Supply_j \leftarrow 0.0$ 
   $Supply_o \leftarrow h^{od}, slist \leftarrow \{o\}, t \leftarrow 0, i \leftarrow 0$ 
  repeat while  $|slist| > 0$  do
    /* find a col with Supply > 0 and no entries in col of M */
     $\forall u \in slist$  do
      if  $Supply_u = 0.0$  then continue
       $count \leftarrow 0$ 
       $\forall v \in V$  do  $count \leftarrow count + M_{vu}$ 
      if  $count = 0$  then
         $i \leftarrow u$ 
         $\forall v \in V$  do  $count \leftarrow count + M_{iv}$ 
         $t \leftarrow Supply_i / count, Supply_i \leftarrow 0.0$ 
        break
      end if
    end do
     $slist \leftarrow slist \setminus \{i\}$ 
     $\forall v \in V$  do
      if  $M_{iv} = 1$  then
         $f_{iv} \leftarrow f_{iv} + t, Supply_v \leftarrow Supply_v + t, M_{iv} \leftarrow 0$ 
        if  $v \neq d$  then  $slist \leftarrow slist \cup \{v\}$ 
      end if
    end do
  end repeat
end

```

**Figure 6: Equal Flow Distribution Procedure**

**procedure** Equal Cost Multi-Path Split( $V, E, D, w, h, c, \mathbf{U}$ )

Inputs:  $V, E, D, w, h, c$

/\*  $V, E$  denotes the network with  $V$  routers and  $E$  arcs \*/

/\*  $D$  denotes the set of demand pairs \*/

/\*  $w_{ij}$  denotes the weight on arc  $(i,j) \in E$  \*/

/\*  $h^{od}$  denotes the demand value for the demand pair  $(o,d)$  \*/

/\*  $c_{ij}$  denotes the capacity of arc  $(i,j) \in E$  \*/

Output:  $\mathbf{U}$

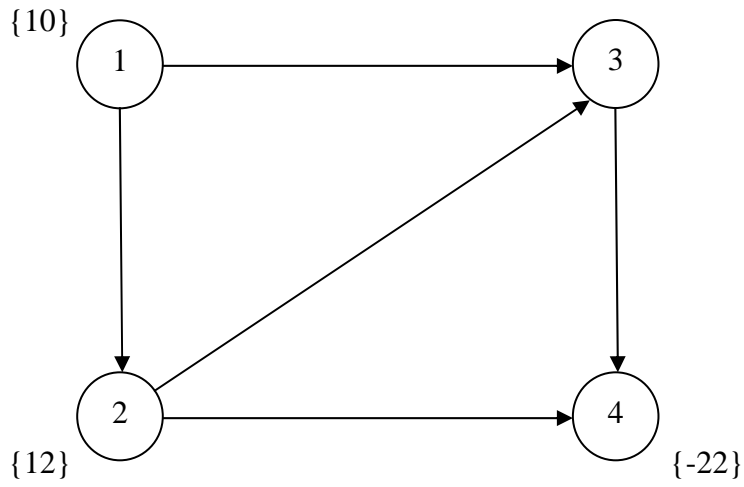
/\*  $\mathbf{U}$  denotes the maximum utilization \*/

```

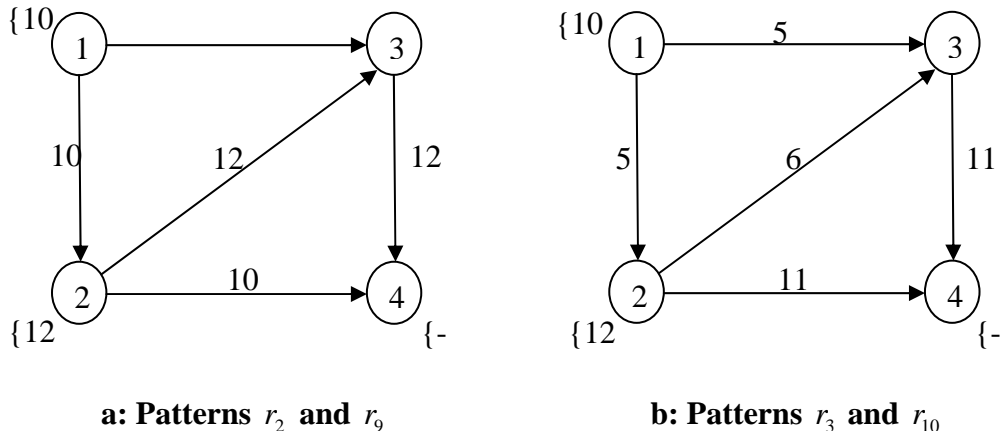
begin
   $\forall (o,d) \in D$  do
     $\forall (j,i) \in V \times V$  do  $parent_{ji} \leftarrow 0$ 
    call Modified Dijkstra( $o, d, V, E, w, \rho$ )
     $\forall (j,i) \in V \times V$  do  $parent_{ji} \leftarrow \rho_{ji}$ 
    /* Construct shortest path graph*/
     $slist \leftarrow \{d\}, T \leftarrow \phi$ 
    repeat while  $|slist| > 0$  do
      let  $s \in slist$ 
       $slist \leftarrow slist \setminus \{s\}$ 
       $\forall v \in V$  do
        if  $parent_{sv} = 1$  then  $T \leftarrow T \cup \{(v,s)\}, slist \leftarrow slist \cup \{v\}$ 
      end do
    end repeat
     $\forall (i,j) \in V \times V$  do  $flow_{ij} \leftarrow 0$ 
    call Equal Flow Distribution( $o, d, V, E, h, T, f$ )
     $\forall (i,j) \in V \times V$  do  $flow_{ij} \leftarrow f_{ij}$ 
  end do
  let  $\mathbf{U} \leftarrow \max\{flow_{ij} / c_{ij} : (i,j) \in E\}$ 
end

```

**Figure 7: Pseudocode for ECMP split procedure**



**Figure 8: Four Router Example Network**



**Figure 9: Examples of Flows on Arcs for Invalid Pattern Combinations**

**procedure** Arc-Path OSPF Weight Setting Algorithm ( $V, L, E, D, h, c, n, P, J, R, M, b, \mathbf{U}$ )

Inputs:  $V, L, E, D, h, c, n, P, J, R, M, b$

/\*  $V, E$  denotes the network with  $V$  routers and  $E$  arcs \*/  
 /\*  $L$  denote the set of links connecting the routers \*/  
 /\*  $D$  denotes the set of demand pairs \*/  
 /\*  $h^{od}$  denotes the demand value for the demand pair  $(o,d)$  \*/  
 /\*  $c_{ij}$  denotes the capacity of arc  $(i,j) \in E$  \*/  
 /\*  $n$  is the maximum number of paths generated initially between a demand pair  
 /\*  $P_{od}$  denotes the set of paths between demand pair  $(o,d) \in D$   
 /\*  $J_p$  denotes the sets of arcs  $(i,j) \in E$  in path  $p \in \mathbf{P}$   
 /\*  $R_{od}$  denotes the set of patterns for demand pair  $(o,d) \in D$   
 /\*  $M_p$  denotes the set of patterns that include path  $p \in P$   
 /\*  $b_{rij}$  denotes the flow distribution of pattern  $r$  that includes arc  $(i,j)$

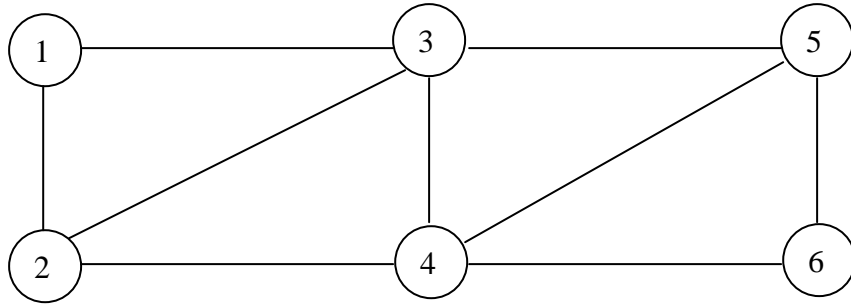
Output:  $\mathbf{U}$

/\*  $\mathbf{U}$  denotes the maximum utilization \*/

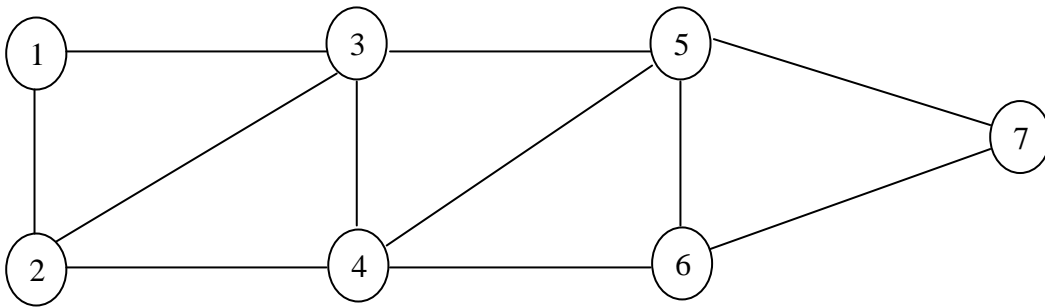
```

begin
  repeat
     $\mathbf{P} \leftarrow \bigcup_{(o,d) \in D} P_{od}, \mathbf{R} \leftarrow \bigcup_{(o,d) \in D} R_{od}, C \leftarrow \phi$ 
     $\forall (o_1, d), (o_2, d) \in D \times D$  such that  $(o_1 \neq o_2)$  do
       $C \leftarrow$  contains pairs of incompatible patterns
    end do
    Solve Arc-Path Model
     $U\_Arcpath \leftarrow \mathbf{U}, tolerance \leftarrow 1.0e^{-6}$ 
    /* The weights from the arc-path model serve as input weights in the equal cost
       multi-path split procedure below */
    call Equal Cost Multi-Path Split( $V, E, D, w, h, c, U\_ECMP$ )
    if  $(|U\_ECMP - U\_Arcpath| / U\_ECMP) \leq tolerance$  then quit
     $\forall (o,d) \in D$  do
       $P_{od} \leftarrow P_{od} \cup$  {paths from equal cost multi-path split procedure}
       $R_{od} \leftarrow R_{od} \cup$  {patterns due to paths from equal cost multi-path split procedure}
    end do
  end repeat
end
  
```

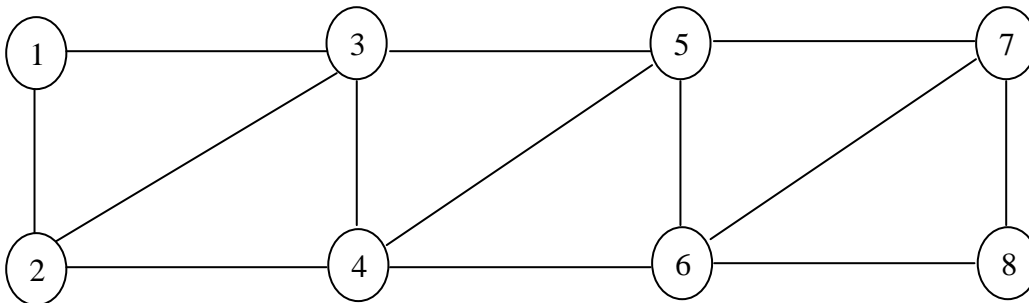
**Figure 10: Pseudocode for Arc-Path OSPF Weight Setting Algorithm**



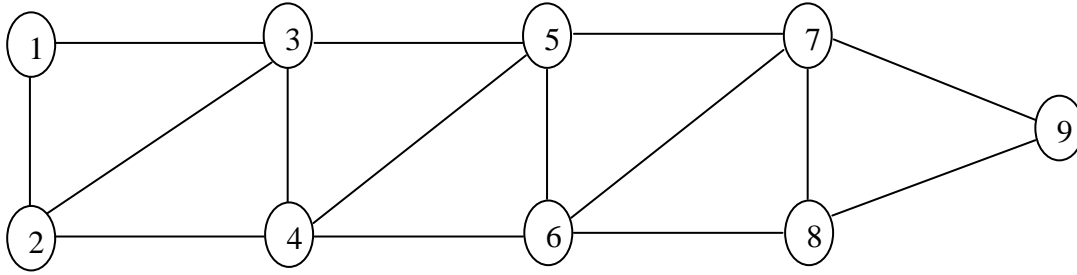
**Figure 11: Six Router Example Graph**



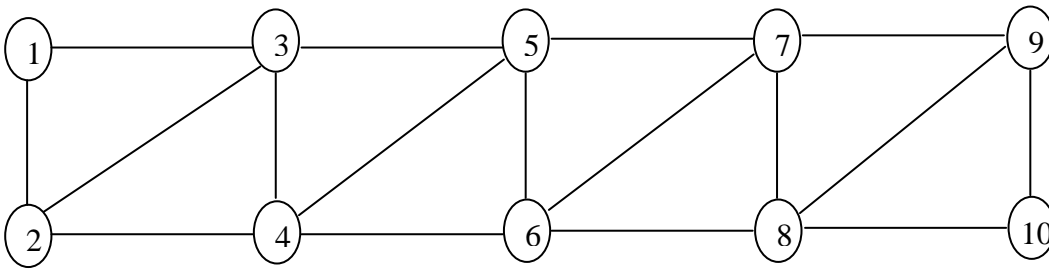
**Figure 12: Seven Router Example Graph**



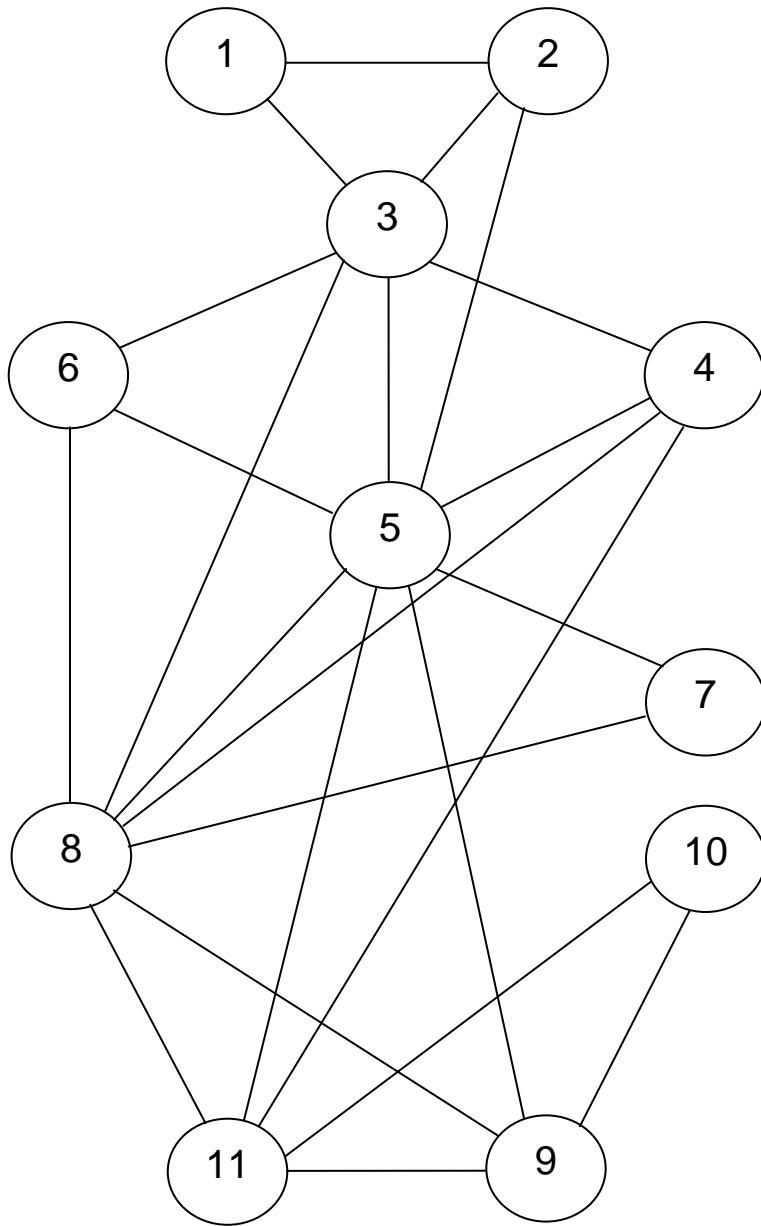
**Figure 13: Eight Router Example Graph**



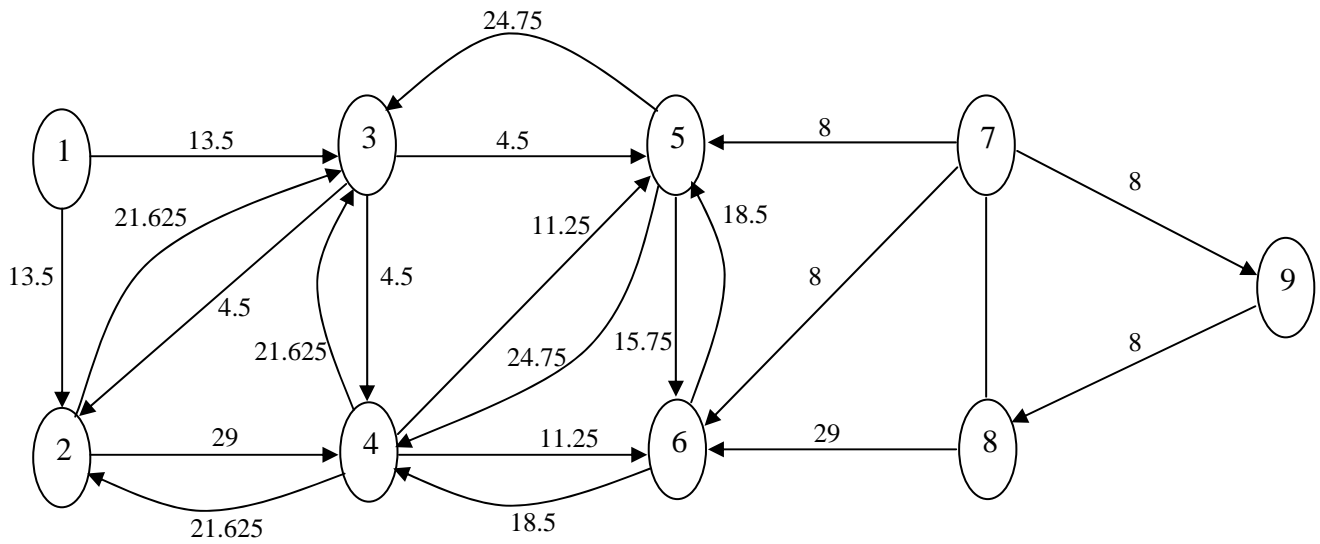
**Figure 14: Nine Router Example Graph**



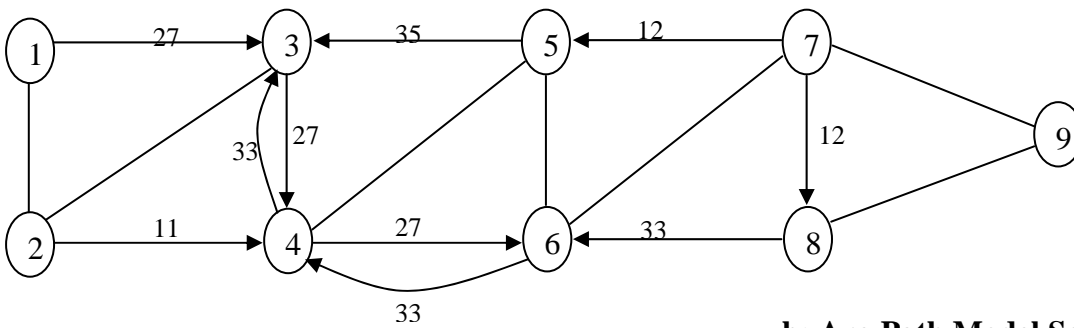
**Figure 15: Ten Router Example Graph**



**Figure 16: 11 Routers, 23 Links Graph**



**a: Node-Arc Model Solution**



**b: Arc-Path Model Solution**

**Figure A1: Flows Distribution for Problem S450**