MAXIMIZING RESIDUAL CAPACITY IN CONNECTION-ORIENTED NETWORKS

KRZYSZTOF WALKOWIAK

Received 5 October 2005; Revised 5 May 2006; Accepted 22 May 2006

The following problem arises in the study of survivable connection-oriented networks. Given a demand matrix to be routed between nodes, we want to route all demands, so that the residual capacity given by the difference between link capacity and link flow is maximized. Each demand can use only one path. Therefore, the flow is modeled as nonbifurcated multicommodity flow. We call the considered problem nonbifurcated congestion (NBC) problem. Solving NBC problem enables robust restoration of failed connections in a case of network failure. We propose a new heuristic algorithm for NBC problem and compare its performance with existing algorithms.

1. Introduction

The congestion problem arises in many practical applications encountered in computer networks. In this paper we concentrate on a special version of the congestion problem. Our objective is to maximize the minimum residual capacity of network links. The residual capacity is defined as the difference between capacity and flow of link and denotes the link capacity not used currently. Since we focus on connection-oriented high-speed networks using techniques like multiprotocol label switching (MPLS), asynchronous transfer mode (ATM), or optical networks each demand can use only one route. It means that the network flow is modeled as nonbifurcated multicommodity flow. The problem formulated in this work is called nonbifurcated congestion (NBC) problem.

We are interested in the NBC problem from the perspective of network survivability. Our objective is to route all demands in the network maximizing the minimum residual capacity of links. Such formulation of the optimization problem is important in network survivability, because resources of residual capacity, indispensable for rerouting of failed connections in a case of network failure, are located "proportionally" in the network. Survivability techniques require residual capacity to perform restoration of broken

Hindawi Publishing Corporation
Journal of Applied Mathematics and Decision Sciences
Volume 2006, Article ID 72547, Pages 1–18
DOI 10.1155/JAMDS/2006/72547

connections. The main idea of approach used to enable survivability in connection-oriented networks is as follows. Each circuit, that is, virtual path in ATM or label switched path in MPLS, has a primary route and a backup route. The primary route is used for transmitting of data in normal, failure-free state of the network. After a failure of the primary route the failed circuit is switched to the backup route. Less residual capacity in the network links means that fewer connections would be restored when a failure occurs. In other words, it is probable that more failed connections would be restored due to higher value of residual capacity. It should be noted that the residual capacity could be applied for assignment of both primary and backup routes in the network. Considerable information on issues of network survivability can be found in [10].

Minimum congestion/maximum throughput problems are common in networking applications. An issue of great significance is the fact that we are routing at the same time many commodities in a network. Moreover, considered problems are the min-max problems. Another common property is that these problems formulated as linear programs (bifurcated flows) or integer problems (nonbifurcated flows) are quite difficult.

In this paper we propose an heuristic algorithm for the NBC problem. The concept of approximation and heuristics has a significant role in all areas of science and engineering. It is expected to be possible to develop approximation algorithms for optimization problems with solid mathematical foundations and which can be efficiently implemented. According to experience with various optimization problems, it is evident that obtaining a partially accurate answer, quickly, can be much preferable to having to wait an excessively long time for an optimal solution. Therefore, approximation algorithms based on various heuristic approaches should be able to combine mathematical theory, computational efficiency, and practicality [2].

The remainder of this paper is organized as follows. In next section the nonbifurcated congestion problem is formulated as zero-one integer problem. Section 3 includes the discussion on related work on congestion problems and various optimization methods applied to problems similar to NBC. In Section 4 we introduce a congestion avoidance algorithm for NBC problem. In Section 5 we present some results of computational tests with a special focus on tuning of algorithm's parameters and comparison of CA against other algorithms. Finally, we conclude and suggest further research.

2. Nonbifurcated congestion problem

We are given a network (G,c) where $G=(N,A)$ is a directed graph with n nodes and m arcs, $c:A\to\mathbb{R}^+$ is a function that defines capacities of the arcs. We assume that all commodities (demands) included in a set P are numbered from 1 to p, where p denotes the number of all commodities. For kth commodity s_k denotes a source and t_k denotes a destination of the commodity. Each commodity of flow requirement Q_k must be routed from node s_k to node t_k through a given network. A multicommodity (m.c.) flow is a set of functions

$$f^k : A \longrightarrow \mathbb{R}^+ \cup \{0\}, \quad k = 1,\dots,p \tag{2.1}$$

for which flow of the kth commodity in arc (x,y) $f^k(x,y)$ for $k=1,\dots,p$ satisfies the

following conditions:

$$\sum_{y\in D(x)} f^k(x,y) - \sum_{y\in B(x)} f^k(y,x) = \begin{cases} Q_k & \text{for } x = s_k, \\ -Q_k & \text{for } x = t_k, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

$$f^k(x,y) \geq 0 \quad \text{for } (x,y) \in A,\ k \in P, \tag{2.3}$$

where $D(x) = \{y : y \in N \text{ and } (x,y) \in A\}$ is a set of destination nodes of edges leaving the node x and $B(x) = \{y : y \in N \text{ and } (y,x) \in A\}$ is a set of source nodes of edges entering the node x. The condition (2.2) is called the conservation of the flow at nodes. The condition (2.3) is a nonnegativity of flow in directed edges. The definition of m.c. flow given by (2.1)–(2.3) is called node-path notation.

$f(x,y)$ denoting flow of arc (x,y) is defined as

$$f(x,y) = \sum_{k\in P} f^k(x,y). \tag{2.4}$$

Multicommodity flows are of two types: bifurcated and nonbifurcated. The former one is a flow in which one commodity can be transported using many paths. Each path carries only a part of the commodity. For the nonbifurcated flow each commodity flows along one path only. An example of bifurcated flow is flow of the Internet using the TCP/IP protocols. Connection-oriented networks like ATM, MPLS, and Frame Relay are examples of networks applying the nonbifurcated m.c. flows. It should be noted that the nonbifurcated flow problem are called in the literature unsplittable flow problem (UFP) [13–15].

In this work we focus on a static flow assignment problem for nonbifurcated flows. The global m.c. flow denoted by $\underline{f} = [f_1, f_2, \ldots, f_m]$ is defined as a vector of flows in all arcs according to constraints (2.1)–(2.4).

In this work we apply a link-path formulation of the nonbifurcated m.c. flow [19]. It is obtained by providing for each commodity $i \in P$ a set of routes (paths) $\Pi_i = \{\pi_i^k : k = 1,\ldots,l_i\}$ from node s_k to node t_k. For nonbifurcated m.c. flow commodity, you can use only one path π_i^k. Let x_i^k denote a 0/1 variable, which equals 1 if π_i^k is the path for commodity i and is 0 otherwise. Another binary variable a_{ij}^k indicates whether or not path π_i^k uses the arc $j \in A$. Using this representation of m.c. flow the nonbifurcated congestion (NBC) problem is as follows:

$$\max z \tag{2.5}$$

such that

$$\sum_{\pi_i^k \in \Pi_i} x_i^k = 1 \quad \forall i \in P, \tag{2.6}$$

$$x_i^k \in \{0,1\} \quad \forall i \in P;\ \pi_i^k \in \Pi_i, \tag{2.7}$$

$$f_j = \sum_{i\in P} \sum_{\pi_i^k \in \Pi_i} a_{ij}^k x_i^k Q_i, \tag{2.8}$$

$$f_a \le c_a \quad \forall a \in A, \tag{2.9}$$

$$z \le c_a - f_a \quad \forall a \in A. \tag{2.10}$$

The objective is to maximize the minimum arc residual capacity of network function (2.5). Since we consider the nonbifurcated multicommodity flow, condition (2.6) states that each commodity can use only one primary route. Constraint (2.7) ensures that decision variables x_i^k are binary ones. Condition (2.8) is a definition of a link flow. Equation (2.9) is a capacity constraint. Finally, constraint (2.10) measures the residual capacity of each link. Variable z is the minimum value of the residual capacity over all arcs in the network. Therefore, the objective function can be also formulated as

$$\max z = \min_{a \in A} (c_a - f_a). \tag{2.11}$$

Note that if we change constraint (2.7) to

$$0 \le x_i^k \le 1 \quad \forall i \in P;\ \pi_i^k \in \Pi_i, \tag{2.12}$$

we will obtain the bifurcated multicommodity flow problem.

NBC is a 0/1 integer program but it is generally considered as a very hard problem. The first reason why this occurs is that the objective function (in the min-max formulation) is convex piecewise linear and not separable with respect to arcs. The second reason is that the solution space that includes all possible paths is extremely large.

3. Related work

Congestion problems formulated similarly to our formulation given by (2.5)–(2.10) can be found in the literature. The *maximum concurrent flow* problem consists in maximizing the throughput of the network, that is, as large a common percentage of each demand as possible should be routed while not exceeding the capacity constraint. In the *minimum congestion* problem full demands must be routed in order to minimize the maximum load of an arc and satisfy the capacity constraint. Also the *relative congestion* defined as the maximum ratio over all arcs of flow f_a divided by the capacity c_a can be applied as an objective function [2, 3, 6, 7, 14, 15].

As mentioned above, our main focus in this work is on the network survivability of connection-oriented networks. Therefore, in this work we apply formulation (2.5)–(2.10) of the congestion problem using nonbifurcated m.c. flows, which is more convenient for use in optimization of flows in survivable networks. Some of algorithms developed for the maximum concurrent flow and the minimum congestion problems can be also applied to our NBC problem.

Most literature we have found either concerns congestion problems of bifurcated m.c. flows, or some limited versions of UFP in the context of congestion.

A comprehensive treatment of various algorithms developed for maximum concurrent flow and minimum congestion problems is given in [2]. Author focuses only on bifurcated m.c. flows. Most of discussed algorithms are based on the flow deviation (FD) method [8]. FD has been proposed in the context of design of packet-switched networks and has been broadly used by the community of telecommunication networks.

Authors of [7] consider a routing problem of bifurcated multicommodity flows consisting in minimizing the maximal relative congestion on the arcs of the network with a bounded number of paths. An algorithm derived from the FD method is proposed. The main idea of Duhamel et al. is the use of a convex increasing congestion function separable on the arcs what tends to push flow on a subset of the active paths produced by the minimum congestion solution. Similar approach is proposed in [4]—an adaptation of the FD method based on Kleinrock's delay function is used to obtain a fully polynomial approximation scheme.

Chlamatac et al. proposed an algorithm for optimization of virtual paths in ATM networks that minimizes the load on networks links [6]. Since ATM is a connection-oriented technique, the network flow is modeled as nonbifurcated m.c. flow. The central idea behind the algorithm is to assign random weights to the links in a special way that assures, for any possible realization of the weights, that the minimum weight path between two nodes will be inevitably a minimum-hop path. Authors prove that for large networks the solution provided by the algorithm is within a small factor of the best possible solution. Numerical evaluation of the algorithm is provided.

Ott et al. consider various problems of flow allocation in the context of MPLS. A relative congestion is taken into account as an objective in one of optimization problems. Since, the problem is formulated as bifurcated m.c. flow, linear programming methods are used [18].

There are many papers concentrating on developing constant-factor approximation algorithms and bounds for various versions of unsplittable flow problems (UFP). Some of them consider a limited versions of UFP—the *single-source unsplittable flow problem*, which consists in allocating a set of commodities having the same source node to single paths [13, 14]. Approximation algorithms for multisource UFP are discussed in [1, 15]. However, works concentrating on constant-factor approximation algorithms lack numerical experiments presenting performance of algorithms in real networks. Main focus is reduced to mathematical properties of algorithms, while the way of how to use these algorithms for optimization of flows in real computer networks is not discussed, what limits significance of such papers for the telecommunication practitioners.

The NBC problem, like many other network design problems, is very complex and numerically intractable even for networks with a small number of nodes. Notice that the NBC problem is an integer 0-1 problem with linear constraints. However, size of the problem is very large even for relatively small networks. A popular method to solve 0-1 problems is the branch-and-bound (B & B) approach. Such algorithms were applied to many problems related to NBC [12, 19, 20]. Nevertheless, B & B algorithms are intractable for networks of medium and large sizes. The only way to solve the NBC problem by an exact algorithm that can produce the optimal solution is to reduce size of the problem and consider only a part of all possible routes. Such an approach, called path generation technique is discussed in [19]. Another possible method to reduce size of the considered problem is the hop-limit approach proposed in [11].

Lagrangian relaxation is a decomposition method that can be applied to 0/1 optimization problems. This approach consists in relaxation of the primal problem by incorporating a subset of constraints to Lagrangian function using dual multipliers. The new

optimization problem, called dual problem, in which the Lagrangian is the criterion function, is much more simpler than primal problem, because the set of constraint is reduced. Moreover, reducing of some constraints (by introducing them into the Lagrangian) eliminates dependency between variables and the Lagrangian problem can be divided into two (or more) separate subproblems. Consequently, we obtain a much less complicated optimization problem, which is tractable in a reasonable time. This is a highly desired feature of the Lagrangian relaxation that motivates the use of this approach. The most common method to solve the dual problem is subgradient optimization, which has long been popular among optimization practitioners, because of its ease of implementation, general applicability and (potential) computational success [19].

The Lagrangian relaxation and subgradient optimization technique is used in [9] to solve the nonbifurcated problem. The goal is to select a set of routes which minimizes the expected end-to-end delay function defined by Kleinrock. Since the objective function is separable with respect to arcs, the Lagrangian relaxation leads to two relatively simple kinds of subproblems. First subproblem can be separated into m (number of arcs) subproblems solved very simply. The second subproblem can be separated into p (number of demands) subproblems, which require calculation of the shortest path under a given metric. Results reported in [9] shows robustness of the Lagrangian approach.

However, the Lagrangian relaxation does not work efficiently for NBC and other min-max 0/1 problems. The main reason of this is that the objective function of NBC is not separable with respect to arcs. Therefore, Lagrangian approach cannot yield subproblems that could be solved effectively.

All references discussed above consider the static optimization of network flows. There are also related works that focus on dynamic, online optimization of network demands. In dynamic routing, requests arrive one-by-one and future demands are unknown. Traditional routing protocols applies a single metric such as hop count or delay that characterizes the network, and use the shortest-path algorithms for path computation [22]. However, in order to support bottleneck metrics like residual capacity or link utilization, other algorithms have been proposed. We focus on one of them, that is, the shortest-widest path (SWP) algorithm proposed in [16, 22]. SWP algorithm finds a path with the maximum residual capacity among all feasible paths. If there are several such paths, the one with minimum hop count is selected. Dynamic routing algorithms can be used also for static optimization of network flows by applying the same algorithm sequentially for all demands. Since all demands are known a priori in static optimization, sequence of processed demands can be changed to improve the results.

4. Congestion avoidance algorithm

In this section we present a new algorithm called congestion avoidance (CA) for the nonbifurcated congestion problem given by (2.5)–(2.10). The main idea of CA algorithm is derived from FD algorithm for nonbifurcated m.c. flows proposed in [8]. The FD algorithm and its modifications have proven its effectiveness in many network design problems [2, 5, 17, 21]. We present two versions of congestion avoidance algorithm.

For the sake of simplicity we define the residual capacity of arc j in the following way:

$$r_a = c_a - f_a \quad \forall a \in A. \tag{4.1}$$

We assume that X_j is a set of variables x_i^k, which are equal to one. The set X_j is called a selection. Each selection determines the unique set of selected routes for all demands. Let X_1 denote a feasible initial solution. In order to find X_1 we apply an algorithm based on the initial phase of the FD algorithm. Let $z(H)$ denote a value of the minimal arc's residual capacity obtained for routes allocation given by a selection H. We start with $j := 1$.

Algorithm CA1 (α,β)

Step 1. For a solution X_j find the minimal value of residual capacity $r_{\min} = \min_{a\in A} r_a$ and the maximal value of residual capacity $r_{\max} = \max_{a\in A} r_a$. Let Cong($\alpha$) be a set that includes all *α-congested* arcs $a \in A$ for which $r_a \le (r_{\min} + \alpha(r_{\max} - r_{\min}))$. Next, let P_{cong} be a set that consists of all demands that path of the demand uses at least one arc included in the set Cong(α).

Step 2. Find a selection SWP(X_j) of variables x_i^k associated with the widest-shortest route π_i^k for a selection X_j. To find a widest-shortest route for each demand $i \in P$ first remove the demand from the network, and next using the SWP algorithm calculate the path. Set $i := 1$ and go to Step 3.

Step 3. Let $H := X_r$.

(a) If $i \in P_{\text{cong}}$, then calculate a selection V from the selection H in the following way: $V := (H - \{x_i^m\}) \cup \{x_i^k\}$ where $x_i^m \in H$, $x_i^k \in \text{SWP}(X_j)$. Routes for other demands except demand i remain unchanged. Otherwise, if $i \notin P_{\text{cong}}$, go to Step 3(c).

(b) If $z(V) \ge z(H)$, then set $H := V$.

(c) If $i = p$, then go to Step 4. Otherwise set $i := i + 1$ and go to Step 3(a).

Step 4. If $j \ge \beta$, stop the algorithm. Otherwise set $j := j + 1$, $X_j := H$ and go to Step 1.

The central idea of the CA1 algorithm is as follows. The algorithm has two parameters that can be tuned. Parameter $\alpha \in (0,1]$ is used to find set Cong(α) including all *α-congested* arcs (Step 1). The second parameter of the CA algorithm, β, is a number of iterations for which the main loop of the algorithm is repeated.

We start with a feasible solution X_1, which defines all routes used by demands. Consequently, having these routes and bandwidth requirements of all demands, flow and residual capacity of each arc can be found. To find all *α-congested* arcs we first find the minimal value of residual capacity $r_{\min} = \min_{a\in A} r_a$ and the maximal value of residual capacity $r_{\max} = \max_{a\in A} r_a$ for an m.c. flow given by current selection. An arc a is *α-congested* if the following condition is satisfied:

$$r_a \le (r_{\min} + \alpha(r_{\max} - r_{\min})). \tag{4.2}$$

Note that if $\alpha = 1$, all arcs are included in the set Cong(α). If $\alpha = 0.1$, only arcs for which the residual capacity is between $r_{\min}$ and $(0.9r_{\min} + 0.1r_{\max})$ are included in the set Cong(α). The Cong(α) is applied to calculate set P_{cong}. The P_{cong} set includes all demands,

which routes use at least one α-*congested* arc (Step 1). The main idea of using parameter α is to concentrate the algorithm on the most congested arcs and try to increase the residual capacity of arcs included in Cong(α) by changing routes for demands included in the set P_{cong}. To improve the solution, we find set SWP(X_j) that comprises new routes for each demand $i \in P_{\text{cong}}$ calculated according to the shortest-widest path algorithm (Step 2). In particular, for each $i \in P_{\text{cong}}$ we remove the demand i from the network (decrease flow on each arc used by the demand i by the value of i capacity), and calculate a new route using SWP algorithm. Next, we try to improve the solution by deviation of one selected demand $i \in P_{\text{cong}}$ to the widest route (Step 3(a)). In Step 3(b) we evaluate the new solution denoted as V. If minimal residual capacity of V is greater or equal to the minimal residual capacity of previous selection, we accept the change of route for demand i. Note that in nonbifurcated flow deviation algorithm [8], solutions are compared using condition "less" ("less" because the problem is to minimize objective function, in our case we want to maximize the objective function). We use the "greater or equal" condition to enlarge the solution space analyzed by algorithm CA. Moreover, another difference between FD and CA is the stopping condition. The FD stops if the m.c. flow is not changed after the deviation. Since we apply the "greater or equal," we have to repeat the main loop of CA β times. Finally, FD uses a shortest path algorithm to obtain a new route for deviation while CA applies shortest-widest path algorithm because our objective is to maximize residual capacity.

Now we present a second version of congestion avoidance algorithm.

Algorithm CA2 (α, δ).

Steps 1–3. The same as in algorithm CA1.

Step 4. If $z(X_j) = z(X_{j-(\delta-1)})$, stop the algorithm. Otherwise set $j := j + 1$, $X_j := H$ and go to Step 1.

Algorithm CA2 has also two parameters that can be tuned. Parameter α has the same task as in CA1. The second parameter of the CA2-δ is used in the stopping condition. We suppose that if for a particular number of iterations the algorithm yields the same result, we can stop the algorithm because a local minimum is obtained. Therefore, if for δ consecutive iterations CA2 does not improve the network congestion given by formula (2.11), the algorithm terminates. Deployment of CA2 was motivated by initial results of CA1 computational tests. Comprehensive analysis of results can be found in the next section.

5. Numerical results

All tested algorithms were coded in C++, and the program was run on an IBM-compatible PC with 2 GHz Intel processor and 512 MB of RAM. To evaluate algorithms CA1, CA2 and other tested algorithms for various networks in terms of topology and density (average node degree) we selected to numerical experiments 7 networks. Three topologies are shown in Figure 5.1. Table 5.1 summarizes the parameters of all sample networks. The first column specifies the name of the parameter, next columns include values of these parameters for each network. Let bandwidth unit (BU) denote an arbitrary unit of

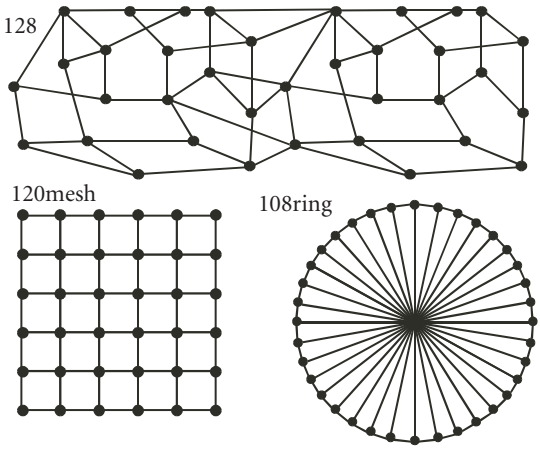

Figure 5.1. Topologies of tested networks 128, 120mesh, and 108ring.

Table 5.1. Parameters of tested networks.

Name of network	104	114	128	144	162	108ring	120mesh
Number of nodes	36	36	36	36	36	36	36
Number of links	104	114	128	144	162	108	120
Topology	Irregular mesh					Ring	Regular mesh
Node degree (average)	2.88	3.17	3.56	4.00	4.50	3.00	3.33
Number of tests experiment A	15	15	15	15	15	15	15
Number of tests experiment B	20	20	20	20	20	20	20
Number of tests experiment C	10	10	10	10	10	10	10

bandwidth, for instance, 1 Mb/s. We assume that for all networks capacity of each link is 5000 BU.

We run three sets of experiments. In experiment A it is assumed that there is a requirement to set up a connection for each direction of every node pair. Thus, the total number of demands (commodities) is 1260. Each demand is defined by the source node, destination node, and flow requirement. Several demand patterns are examined for each network. However, all demands are homogenous and flow requirement for each demand is the same. For instance, for network 104 we perform 15 simulations starting with flow requirement of each demand equal to 50 BU, the biggest value of flow requirement is 64 BU. In experiment B, there are also 1260 demands—one demand for each origin-destination node pair. However, bandwidth requirements are heterogeneous—a random value is chosen independently for each demand. Finally, in experiment C 2500

fully heterogeneous demands are generated, that is, origin node, destination node, and bandwidth requirement are chosen for each demand at random.

The first objective of experiments is tuning of the CA1 algorithm. As a starting solution we use the solution given by the initial phase of the FD for nonbifurcated flows [8]. Due to initial trial runs we decided to set the number of iterations (parameter β) to 50. We run simulations for the following values of parameter $\alpha = \{1; 0.75; 0.5; 0.25; 0.1; 0.05; 0.01; 0.005; 0.001\}$.

To compare results we apply the *competitive ration* performance indicator. The competitive ration, which indicates how well an algorithm performs for a given parameter α, is defined as the difference between result obtained for a particular parameter α and the maximum value of minimal residual capacity obtained in a considered simulation (unique in terms of network topology and demand pattern). For instance, if for the test consisting of 9 simulations of CA (9 different values of parameter α) maximum value of minimal residual capacity is 2500 and the minimal residual capacity of the considered simulation is 2000; the competitive ration is calculated as follows: $(2500 - -2000)/2500 = 20\%$. The competitive ration indicates quality of obtained result of given simulation compared to results of other simulations for a particular network and demand. Low value of competitive ration means that the result of current simulation is very close to the best results obtained in a given test. Notice that the competitive ration must be in the range $[0, 100\%]$. For presentation of aggregate results we apply the *aggregate competitive ration*, which is a sum of competitive rations over all considered experiments.

In Table 5.2, for each experiment, and network topology combination, we report aggregate competitive ration obtained for tested values of α. Generally, we can see that results obtained for irregular mesh topologies (104, 114, 128, 144, 162) are similar for various values of α. Only for networks 108ring and 120mesh more differences can be observed. However, for experiment C the difference between regular and irregular topologies is relatively smaller. The experimental data is reasonably well explained by the fact that for experiment C there are more demands (2500) and demands are heterogeneous. For experiments A and B there are 1260 homogeneous demands (one demand for each origin-destination node pair).

Table 5.3 shows the ranking of all tested values of parameter α. For instance, the second column of Table 5.3 demonstrates that setting $\alpha = 1$ provides the "first place" (best results) in 101 of 105 simulations run for experiment A, 89 of 140 simulations run for experiment B and 23 of 70 simulations run for experiment C. Columns 3–10 of Table 5.3 present detailed ranking for other values of α. In experiments A and B for many demand patterns algorithm CA1 yields the same result regardless of parameter α. As above, the results obtained for experiment C differ from other experiments. This follows from the heterogeneity of demands. More demands with random bandwidth requirements mean more combinations of routes and more possible solutions. Therefore, CA1 generates different results for various α.

Now we focus on the second parameter of algorithm CA1–number of iterations denoted as β. Recall, that in our simulations we set β to 50. However, for each simulation we record the number of iteration in which the final (best) solution is obtained. After this iteration the solution was not improved. Table 5.4 reports our results. As in Table 5.2,

Table 5.2. Aggregate competitive ration of various simulation scenarios for experiments A, B, C.

Network	$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0.5$	$\alpha = 0.25$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.005$	$\alpha = 0.001$
Experiment A									
All nets	56.5%	56.5%	56.5%	56.5%	56.5%	56.5%	11.5%	11.5%	11.5%
104	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	7.2%	7.2%	7.2%
114	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
128	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
144	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
162	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
108ring	16.0%	16.0%	16.0%	16.0%	16.0%	16.0%	4.3%	4.3%	4.3%
120mesh	40.6%	40.6%	40.6%	40.6%	40.6%	40.6%	0.0%	0.0%	0.0%
Experiment B									
All nets	81.5%	81.5%	81.5%	81.5%	79.7%	76.0%	94.6%	100.3%	80.5%
104	6.4%	6.4%	6.4%	6.4%	6.4%	1.8%	5.7%	13.3%	13.6%
114	8.3%	8.3%	8.3%	8.3%	8.3%	8.1%	11.4%	9.7%	6.6%
128	11.3%	11.3%	11.3%	11.3%	10.4%	11.6%	11.9%	13.0%	12.8%
144	11.3%	11.3%	11.3%	11.3%	10.3%	8.2%	6.7%	5.8%	4.4%
162	35.3%	35.3%	35.3%	35.3%	35.3%	37.4%	23.2%	15.7%	7.8%
108ring	1.2%	1.2%	1.2%	1.2%	1.2%	1.2%	18.8%	27.0%	19.6%
120mesh	7.7%	7.7%	7.7%	7.7%	7.7%	7.7%	16.9%	15.9%	15.7%
Experiment C									
All nets	20.2%	20.1%	19.1%	18.0%	21.2%	18.0%	12.9%	19.9%	34.7%
104	1.2%	1.5%	1.1%	1.1%	2.2%	2.3%	2.4%	2.3%	8.2%
114	0.5%	0.5%	0.5%	0.7%	0.8%	1.0%	0.6%	0.6%	1.0%
128	1.5%	1.4%	1.2%	1.5%	0.9%	1.2%	0.9%	0.7%	1.6%
144	1.9%	1.9%	1.8%	1.8%	1.6%	1.4%	1.5%	1.5%	1.1%
162	5.1%	5.0%	4.4%	4.1%	3.5%	3.0%	0.8%	4.3%	7.4%
108ring	5.0%	4.7%	4.2%	4.5%	4.5%	3.4%	2.1%	5.0%	7.7%
120mesh	5.1%	5.1%	6.0%	4.4%	7.9%	5.7%	4.7%	5.5%	7.8%

Table 5.3. Ranking of various values of parameter α.

Experiment	$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0.5$	$\alpha = 0.25$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.005$	$\alpha = 0.001$
A	101	101	101	101	101	101	103	103	103
B	89	89	89	89	90	93	96	97	101
C	23	24	22	20	27	25	31	25	23

analysis of Table 5.4 shows that the results depend on the experiment. When the number of demands grows, the CA1 algorithm needs more iterations to find a stable solution. Also of significance is that the number of iterations grows as the value of α decreases. This is also evident from the construct of algorithm CA1. In each iteration we try to change routes of demands that use α-congested arcs. If α is low, only few arcs are taken into account, and consequently relatively small number of demands are deviated. Therefore, CA1 needs more iteration for small values of α. However, since fewer paths are rerouted, the calculation time should be lower for low α. This can be observed in Table 5.5, which summarizes the decision time needed to run all 50 iterations of CA1. The time required to find the initial solution is not included, since it is the same for all tested α.

Figure 5.2 shows the convergence of the CA1 algorithm obtained in experiment C for network topology 128. Curves for five values of parameter α are presented. We can see that for $\alpha > 0.1$ the algorithm requires only few iterations to find a stable solution. For smaller values of α, the number of iterations grows.

Summarizing the discussion on the CA1 parameter tuning we conclude that the algorithm offer similar results for tested values of α. However, if we take into account also the number of iterations and decision time, we can see that for $\alpha = 0.01$ combination of both objectives: quality of result and calculation time is in our opinion the best. The parameter β defining the number of algorithm's iterations can be reduced to value 10 for experiment C and 5 for the two other experiments.

Our analysis of the CA1 performance suggests that the stopping criterion of the algorithm can be improved to minimize the decision time. Therefore, we developed a second algorithm, CA2, that examines results of consecutive iterations and if for a particular number of iterations the results are the same, the algorithm stops. Since CA2 is very similar to CA1, we apply results of CA1 tuning and decide to use the CA2 algorithm with the following parameters: $\alpha = 0.01$, $\delta = 3$.

For reference we selected several algorithms developed of nonbifurcated versions of the congestion problem. We implement the algorithm CFZ proposed by Chlamatac et al. [6]. CFZ minimizes the network load, however, the capacity constraint is not taken into account, that is, links have no limits on the load they can carry. Therefore, we propose a modification of the CFZ algorithm (called CFZMod), in which only feasible paths are considered in Step 4 of CFZ. By a feasible path for a particular demand we mean a path that has residual capacity of all arcs not lower than bandwidth requirement of the demand. Consequently, we try to avoid allocating demands to paths that cannot transport the whole capacity of the demand. If a feasible path does not exist for, it means that the algorithm cannot yield a feasible (in terms of capacity constraint) solution.

Another possible method of solving the NBC problem can be derived from works [4, 7] that propose applying a convex increasing congestion function separable on the arcs that leads to improvement of network congestion. As in [4] we implement the FD algorithm, however we use the nonbifurcated version of FD, while Bienstock and Raskina focuses on bifurcated flows. As objective we apply the average delay function that fulfills required conditions; it is a convex increasing function separable on the arcs.

We also use the shortest-widest path (SWP) algorithm [16, 22] for comparison of results. Since we consider a static problem, we can employ the SWP for various orderings

Table 5.4. Average number of CA1 iterations, after which the final solution is obtained.

Network	$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0.5$	$\alpha = 0.25$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.005$	$\alpha = 0.001$
Experiment A									
All nets	1.3	1.3	1.3	1.3	1.3	1.4	2.3	2.3	2.3
104	1.6	1.6	1.6	1.6	2.2	2.2	2.9	2.9	2.9
114	1.0	1.0	1.0	1.0	1.0	1.7	3.1	3.1	3.1
128	1.0	1.0	1.0	1.0	1.0	1.1	1.3	1.3	1.3
144	1.0	1.0	1.0	1.0	1.0	1.0	2.1	2.1	2.1
162	1.0	1.0	1.0	1.0	1.0	1.0	1.7	1.7	1.7
108ring	2.2	2.2	2.2	2.2	2.2	2.2	3.7	3.7	3.7
120mesh	1.0	1.0	1.0	1.0	1.0	1.0	1.3	1.3	1.3
Experiment B									
All nets	1.1	1.1	1.1	1.1	1.2	1.3	2.1	2.5	3.4
104	1.5	1.5	1.5	1.5	2.1	2.5	3.4	3.8	4.9
114	1.0	1.0	1.0	1.1	1.1	1.3	2.5	3.1	4.0
128	1.0	1.0	1.0	1.0	1.1	1.2	2.2	2.8	4.4
144	1.0	1.0	1.0	1.0	1.0	1.0	1.8	2.1	3.3
162	1.0	1.0	1.0	1.0	1.0	1.0	1.9	2.1	2.9
108ring	1.1	1.1	1.1	1.1	1.1	1.1	1.4	1.5	1.8
120mesh	1.3	1.3	1.3	1.3	1.3	1.3	1.7	2.0	2.9
Experiment C									
All nets	3.2	3.2	3.2	3.3	3.6	3.9	5.6	7.1	8.9
104	1.5	1.5	1.6	1.7	2.7	3.5	5.3	6.6	7.0
114	1.8	1.8	1.5	1.7	2.4	3.2	5.0	6.4	6.9
128	2.0	1.9	2.1	1.9	2.9	3.4	6.0	6.5	8.9
144	1.1	1.4	1.2	1.2	1.7	2.4	4.5	5.5	7.0
162	2.3	2.5	2.6	2.5	3.3	4.3	6.9	7.5	9.3
108ring	7.0	6.4	7.5	7.9	7.2	4.5	7.0	10.3	13.4
120mesh	6.6	6.6	5.7	5.9	5.0	6.1	4.7	6.6	9.8

Table 5.5. Average decision time of CA1 for various values of parameter α given in seconds.

Experiment	$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0.5$	$\alpha = 0.25$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.005$	$\alpha = 0.001$
A	76	74	69	62	53	51	38	38	38
B	81	78	73	65	56	53	25	13	6
C	324	319	299	261	210	156	100	64	27

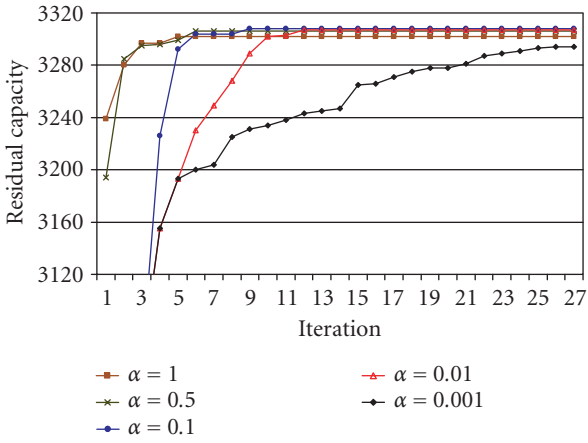

Figure 5.2. Convergence of CA algorithm for network 128 in experiment C.

of commodities. In particular, we develop three versions of algorithm using the shortest-widest path (SWP) method. In SWPNorm algorithm paths are processed without any sorting, that is, demands are allocated one-by-one as they are located in the set to paths found by SWP. SWPRand(n) method assumes that paths are sorted randomly, n various (in terms of demands' ordering) sets are generated. In SWPSort algorithm initially paths are allocated by the SWPNorm algorithm. Next, for each path the path's residual capacity (i.e., the miminal value of residual capacity over all arcs belonging to the path) is calculated. Finally, paths are sorted according to their residual capacity, starting from the path with the smallest value of residual capacity. If two paths have the same residual capacity, we compare the value of paths' bandwidth requirement.

As discussed in a related work, there are also many other methods developed for congestion problems for bifurcated m.c. flows. However, since we consider a network that uses a connection-oriented technique (e.g., MPLS, ATM), the algorithms developed for bifurcated flows cannot be applied in our case.

According to the discussion on the CA1 parameter setting we decided to use the following two combinations of CA1 parameters for comparison with other algorithms: CA1(0.01,5) ($\alpha = 0.01$; $\beta = 5$) and CA1(0.01,10) ($\alpha = 0.01$; $\beta = 10$). We also report results of CA2(0.01,3) ($\alpha = 0.01$; $\delta = 3$). As a starting solution of both CA algorithms we use the route combination yielded by the initial phase of nonbifurcated FD algorithm.

Table 5.6 shows the ranking of all tested algorithms. For instance, the seventh column of Table 5.6 demonstrates that CA1(0.01,5) algorithm provides the "first place" (best results) in 103 of 105 simulations run for experiment A, 137 of 140 simulations run for experiment B, and 42 of 70 simulations run for experiment C. In experiments A and B for many demand patterns algorithms CA1 and CA2 yield the same result regardless of parameter α. As above, the results obtained for experiment C differ from other experiments. This is due to heterogeneity of the demands. More demands with random bandwidth requirements mean more combinations of routes and more possible solutions. Therefore, CA1 and CA2 generate different results.

Table 5.6. Ranking of tested algorithms.

Experiment	SWP	SWPSort	SWPRand	CFZ	CFZMod	FD	CA1(0.01,5)	CA1(0.01,10)	CA2(0.01,3)
A	15	2	36	0	3	25	103	105	105
B	0	1	0	0	0	16	137	140	140
C	0	2	0	0	0	0	42	65	70

Table 5.7. Number of tests in which algorithm cannot find a feasible solution.

Experiment	SWP	SWPSort	SWPRand	CFZ	CFZMod	FD	CA1(0.01,5)	CA1(0.01,10)	CA2(0.01,3)
A	26	36	11	93	7	0	0	0	0
B	87	76	82	131	4	0	0	0	0
C	9	8	7	38	4	0	0	0	0

It should be noted that tested algorithms do not always yield a feasible solution. Detailed results are reported in Table 5.7. The worst performance yields the CFZ algorithm. In experiment A for only 12 of 105 tests (unique in terms of network topology and demand pattern) CFZ can find a feasible solution. This is because, as mentioned above, CFZ does not apply the capacity constraint. Furthermore, CFZ uses only the shortest routes for each demand. In congested networks to omit highly congested arcs, other longer routes should be used. We can observe that the modified CFZ produce much more feasible solutions. Only FD, CA1(0.01,5), CA1(0.01,10), and CA2(0.01,3) can find a feasible result for each test.

In Table 5.8, for each experiment and network combination, we report aggregate competitive ration obtained all for tested algorithms. Two versions of CA outperform other algorithms in all experiments. For the most heterogeneous case-experiment C-CA2 provides the best results. SWPRand and FD offer the best performance, regardless of CA. We do not show results of CFZ, because as mentioned above in most of tests, CFZ cannot find a feasible solution.

Table 5.9 summarizes the decision time needed to run tested algorithms. Recall that to run the CA algorithm we need an initial solution, which is provided by FD. SWP and SWPSort require only one execution of the shortest-widest path algorithm for each demand. Therefore, the decision time is relatively low. The CA algorithm itself needs only few seconds to find solution. However, when we add the decision time of FD the time grows substantially. Notice that, since in experiment C the number of demands (2500) is about twice the number of demands in experiments A and B (1260), calculation time is also larger. Generally, decision times obtained for various network topologies are comparable.

Concluding, both versions of CA outperform all other tested algorithms. Evaluating jointly objectives of solution quality and decision time CA2(0.01,3) offers the best performance among all tested algorithms. Another important observation is that network topology and demand pattern can affect the performance of CA algorithms in terms of decision time and quality of results.

Table 5.8. Aggregate competitive ration of various simulation scenarios for all tested algorithms.

Network	SWP	SWPSort	SWPRand	CFZMod	FD	CA1(0.01,5)	CA1(0.01,10)	CA2(0.01,3)
Experiment A								
All nets	5341%	6143%	2932%	8728%	3312%	18%	0%	32%
104	455%	929%	228%	1396%	754%	0%	0%	0%
114	461%	450%	0%	1408%	713%	0%	0%	0%
128	1354%	695%	513%	1346%	415%	0%	0%	0%
144	716%	458%	352%	1238%	702%	0%	0%	0%
162	0%	638%	0%	1178%	560%	0%	0%	0%
108ring	1341%	1473%	903%	801%	68%	18%	0%	32%
120mesh	1014%	1500%	934%	1361%	99%	0%	0%	0%
Experiment B								
All nets	11542%	10318%	10616%	12726%	4002%	80%	74%	74%
104	954%	1674%	1113%	1954%	909%	6%	0%	0%
114	1155%	774%	547%	1920%	566%	74%	74%	74%
128	1909%	791%	1859%	1935%	700%	0%	0%	0%
144	1666%	1273%	1282%	1845%	820%	0%	0%	0%
162	1858%	1806%	1815%	1424%	825%	0%	0%	0%
108ring	2000%	2000%	2000%	1920%	77%	0%	0%	0%
120mesh	2000%	2000%	2000%	1728%	106%	0%	0%	0%
Experiment C								
All nets	2284%	2020%	1980%	5130%	3220%	101%	6%	1%
104	208%	154%	159%	888%	513%	5%	0%	0%
114	48%	32%	29%	627%	500%	6%	0%	0%
128	328%	270%	301%	740%	509%	10%	0%	0%
144	281%	171%	191%	911%	443%	5%	1%	1%
162	105%	74%	80%	849%	693%	42%	0%	0%
108ring	706%	685%	623%	850%	247%	30%	4%	0%
120mesh	608%	634%	597%	265%	314%	3%	0%	0%

Table 5.9. Average decision time of tested algorithms given in seconds.

Experiment	SWP	SWPSort	SWPRand	CFZ	CFZMod	FD	CA1(0.01,5)	CA1(0.01,10)	CA2(0.01,3)
A	0.4	0.4	9.8	54.9	57.6	21.3	3.3	7.3	2.6
B	0.3	0.4	10.3	54.9	57.6	26.2	2.5	5.1	2.0
C	2.2	2.3	46.2	108.9	141.2	49.8	7.1	19.0	14.4

6. Concluding remarks

In this paper a new algorithm-congestion avoidance (CA) for the congestion problem in connection-oriented networks has been proposed. We have discussed main properties of two versions of CA. Next we have run simulations to calibrate two input parameters of CA1. According to obtained results, performance of CA depends on both parameters. However, the influence of α parameter is not substantial. If we take into account jointly the quality of results and computation time, we can conclude that the best results provide $\alpha = 0.01$. The second parameter denoting the number of CA1's iterations depends on traffic demands pattern. For heterogeneous demands the algorithm needs more iterations than for homogenous demands. Increasing the number of demands causes that more calculations steps are required to converge CA1 to a stable solution. Finally, we have evaluated CA1 and CA2 against existing heuristics. Results show supremacy of CA comparing to other algorithms. In future work we plan to apply CA to the problem of primary routes assignment in survivable connection-oriented networks like MPLS and ATM. In our opinion, allocation of demand paths that maximizes residual capacity can guarantee good restoration performance after a network failure.

Acknowledgment

This work was supported by a research project of the Polish State Committee for Scientific Research carried out in years 2005–2007.

References

[1] Y. Azar and O. Regev, *Strongly polynomial algorithms for the unsplittable flow problem*, Proceedings of Integer Programming and Combinatorial Optimization (Utrecht, 2001), Lecture Notes in Comput. Sci., vol. 2081, Springer, Berlin, 2001, pp. 15–29.

[2] D. Bienstock, *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*, International Series in Operations Research & Management Science, vol. 53, Kluwer Academic, Massachusetts, 2002.

[3] D. Bienstock and O. Günlük, *Computational experience with a difficult mixed-integer multicommodity flow problem*, Mathematical Programming, Series A **68** (1995), no. 2, 213–237.

[4] D. Bienstock and O. Raskina, *Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem*, Mathematical Programming, Series B **91** (2002), no. 3, 479–492.

[5] J. Burns, T. Ott, A. Krzesinski, and K. Müller, *Path selection and bandwidth allocation in MPLS networks*, Performance Evaluation **52** (2003), no. 2-3, 133–152.

[6] I. Chlamatac, A. Farago, and T. Zhang, *Optimizing the system of virtual paths*, IEEE/ACM Transactions on Networking **2** (1994), no. 6, 581–587.

[7] C. Duhamel, B. Vatinlen, P. Mahey, and F. Chauvet, *Minimizing congestion with a bounded number of paths*, Proceedings of Algorithmic of Telecommunications (ALGOTEL '03), Banyuls, May 2003, pp. 155–160.

[8] L. Fratta, M. Gerla, and L. Kleinrock, *The flow deviation method: an approach to store-and-forward communication network design*, Networks **3** (1973), 97–133.

[9] B. Gavish and S. Hantler, *An algorithm for optimal route selection in SNA networks*, IEEE Transactions on Communications **31** (1983), no. 10, 1154–1160.

[10] W. Grover, *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*, Prentice-Hall PTR, New Jersey, 2004.

[11] M. Herzberg, S. Bye, and A. Utano, *The hop-limit approach for spare-capacity assignment in survivable networks*, IEEE/ACM Transactions on Networking **3** (1995), no. 6, 775–784.

[12] A. Kasprzak, *Exact and approximate algorithms for topological design of wide area networks with non-simultaneous single commodity flows*, Computational Science—ICCS 2003. Part IV, Lecture Notes in Computer Science, vol. 2660, Springer, Berlin, 2003, pp. 799–808.

[13] J. M. Kleinberg, *Single-source unsplittable flow*, Proceedings of 37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996), IEEE Computer Science Press, California, 1996, pp. 68–77.

[14] S. Kolliopoulos and C. Stein, *Improved approximation algorithms for unsplittable flow problems*, Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS '97), 1997, pp. 426–435.

[15] P. Kolman and C. Scheideler, *Improved bounds for the unsplittable flow problem*, Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02), California, 2002, pp. 184–193.

[16] Q. Ma and P. Steenkiste, *On path selection for traffic with bandwidth guarantees*, Proceedings of IEEE International Conference on Network Protocols, Georgia, 1997, pp. 191–202.

[17] K. Murakami and H. Kim, *Virtual path routing for survivable ATM networks*, IEEE/ACM Transactions on Networking **4** (1996), no. 1, 22–39.

[18] T. Ott, T. Bogovic, T. Carpenter, K. Krishnan, and D. Shallcross, *Algorithms for flow allocation for multi protocol label switching*, Telcordia Technical Memorandum TM-26027, 2001.

[19] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*, Morgan Kaufman, California, 2004.

[20] K. Walkowiak, *A branch and bound algorithm for primary routes assignment in survivable connection oriented networks*, Computational Optimization and Applications **27** (2004), no. 2, 149–171.

[21] ———, *A new method of primary routes selection for local restoration*, Networking, Lectures Notes in Computer Science, vol. 3042, Athens, May 2004, pp. 1024–1035.

[22] Z. Wang and J. Crowcroft, *Quality-of-service routing for supporting multimedia applications*, IEEE Journal on Selected Areas in Communications **14** (1996), no. 7, 1288–1234.

Krzysztof Walkowiak: Chair of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
E-mail address: krzysztof.walkowiak@pwr.wroc.pl