A New Method of Primary Routes Selection for Local Restoration

Krzysztof Walkowiak

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

Abstract. We consider the problem of non-bifurcated multicommodity flows rerouting in survivable connection-oriented networks. We focus on local restoration, in which the backup route is reroutes the flow only around the failed arc. The origin node of the failed arc is responsible for rerouting. We introduce and discuss a new convex function for optimisation of primary routes. We propose a heuristic algorithm for assignment of primary routes employing the developed function. During numerical experiments we examine performance of the new function compared to other functions proposed previously in the literature.

1 Introduction

Recently, issues of network survivability become more important due to the growing requirements for QoS and traffic engineering. Connection-oriented network technologies like Asynchronous Transfer Mode (ATM), MultiProtocol Label Switching (MPLS) use comparable approaches to enable network survivability. The main idea of this approach is as follows. Each circuit, i.e. 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. The process of switching is easy, i.e. the circuit's identifier numbers are changed in network nodes. All backup routes have zero bandwidth. After activation there are assigned with necessary bandwidth. In this work we focus on local restoration (called also rerouting or repair) [1], [8]. The backup route is found only around the failed arc. The origin node of the failed arc is responsible for rerouting.

In modern computer networks a single-link failure is the most common and frequently reported failure event [6]. Therefore, in most of optimization models a single-link failure is considered as the basic occurrence. Spare capacity is computed to provide full restoration in case of a failure of any single-link. In networks having limited resources of spare capacity 100% restoration is not always possible and routes are designed to minimize effects of the failure, i.e. to minimize to amount of flow lost due to a failure.

For the context of this work we concentrate on an existing facility network, i.e. we do not consider facility capacity planning and topological design. Joint optimization of primary and backup routes must be carried out to find a globally optimal solution of the lost flow due to a network failure for a projected traffic demand. Since the optimization is conducted jointly over primary and backup routes, the complexity of the problem grows tremendously. The main idea of our approach is to partition the problem into two simpler problems: first optimize primary routes and next find backup routes for already established primary routes. Since there is mutual dependency between the primary routes and the backup routes assignment, the obtained solution cannot be claimed to be an optimum if these problems are treated separately. However, obtained results prove robustness of this approach [9], [10]. A key problem of our approach is to define an objective function for primary routes' assignment. Such a function must indicate preparation of the network to the rerouting process. Since there are many algorithms for convex multicommodity flow problem, it would be convenient if the function were convex.

In this work we define and discuss a new convex function for optimisation of primary routes in a network applying local rerouting. We also propose a heuristic algorithm for assignment of primary routes using this function. Furthermore, we present and discuss results of extensive simulations.

2 Definition of a New Function for Local Rerouting

Our objective in this section is twofold: Firstly, we want to define a function that reflects the preparation of the network to the local rerouting. Secondly, we plan to discuss main characteristics of this function.

Various network or transportation problems can be modeled as multicommodity (m.c.) flow problem [4], [5]. Multicommodity flows are of two types: bifurcated and non-bifurcated. In this work we focus on non-bifurcated m.c., in which each commodity flows along one route only.

We are given a network (G, c) where $G = (V, A)$ is a directed graph with *n* vertices representing routers or switches and *m* arcs representing links, $c : A \to R^+$ is a function that defines capacities of the arcs. We denote by $o : A \to V$ and $d : A \to V$ functions defining the origin and destination node of each arc. For each $a \in A$ we call $\text{in}(a) = \{k \in A \mid d(k) = d(a), k \neq a\}$ the set of incoming arcs of $d(a)$ except *a*, and $\text{out}(a) = \{k \in A \mid o(k) = o(a), k \neq a\}$ the set of outgoing arcs of $o(a)$ except *a*.

To mathematically represent the problem we introduce the following notations

f_a	represents the total flow on arc *a*.
c_a	capacity of arc *a*.
$g_v^{\text{out}} = \sum_{i:o(i)=v} f_i$	aggregate flow of outgoing arcs of *v*.
$g_v^{\text{in}} = \sum_{i:d(i)=v} f_i$	aggregate flow of incoming arcs of *v*.
$e_v^{\text{out}} = \sum_{i:o(i)=v} c_i$	aggregate capacity of outgoing arcs of *v*.

$e_v^{\text{in}} = \sum_{i:d(i)=v} c_i$ The aggregate capacity of incoming arcs of *v*.

Definition 1. *The global non-bifurcated m.c. flow denoted by* $\underline{f} = [f_1, f_2, ..., f_m]$ *is defined as a vector of flows in all arcs. We call a flow* $\underline{f}$ feasible *if for every arc* $a \in A$ *the following inequality holds*

$$\forall a \in A : f_a \le c_a \qquad \textbf{(1)}$$

Inequality (1) *ensures that in every arc flow is not greater then capacity. This inequality is called a capacity constraint.*

For the sake of simplicity we introduce the following function

$$\varepsilon(x) = \begin{cases} 0 & \text{for} \quad x \le 0 \\ x & \text{for} \quad x > 0 \end{cases} \qquad \textbf{(2)}$$

To analyze properties of the local restoration we consider an arc $k \in A$. We assume failure of *k*. Recall that in the local rerouting flow on the arc *k* must be rerouted by the source node of the arc *k*. Therefore, spare capacity of outgoing arcs of *o*(*k*) except *k* is a potential bottleneck of the restoration process. Notice that if

$$f_k \le \sum_{i \in \text{out}(k)} (c_i - f_i) \qquad \textbf{(3)}$$

then flow of the failed *k* can be restored using spare capacity of other links leaving the origin node of *k*. Otherwise if

$$f_k > \sum_{i \in \text{out}(k)} (c_i - f_i) \qquad \textbf{(4)}$$

then some flow of the failed link *k* cannot be restored because spare capacity of other arcs leaving the origin node of *k* is too small. It means that those arcs block the 100% restoration and some flow of *k* is lost. In formulas (3-4) we assume that reroutable flow can be split to a number of different routes. It means that in order to define a function of lost flow we relax the non-bifurcated flow to bifurcated flow. It is, according to [6], a reasonable approach for backbone networks where vary large volume of various calls are transmitted. Furthermore, such assumption makes easier the analysis. Recalling definition of $g_{o(k)}^{\text{out}}$ and $e_{o(k)}^{\text{out}}$ and applying formulas (3-4) we define the function L_k^{out} of the arc *k* flow lost in the node *o(k)* in the following way

$$L_k^{\text{out}}(\underline{f}) = \varepsilon\left(g_{o(k)}^{\text{out}} - (e_{o(k)}^{\text{out}} - c_k)\right) \qquad \textbf{(5)}$$

Note that L_k^{out} denotes lost flow that cannot be restored using arcs leaving the node *o*(*k*) due to limited spare capacity of these arcs. The L_k^{out} function depends on the flow $g_{o(k)}^{\text{out}}$ leaving the node *o*(*k*). It is not dependent directly on the flow f_k. Correspondingly, we define the function L_k^{in} that denotes lost flow that cannot be restored using arcs entering the node *d*(*k*).

$$L_k^{\text{in}}(\underline{f}) = \varepsilon\left(g_{d(k)}^{\text{in}} - (e_{d(k)}^{\text{in}} - c_k)\right) \qquad \textbf{(6)}$$

Definition 2. *We call an arc k* adjacent *to an arc a if o*(*k*)*=o*(*a*) *or d*(*k*)*=d*(*a*)*. If k is not adjacent to a we call it* remote *to a.*

Function L_k defined below is a linear combination of flow lost in arcs outgoing *o*(*k*) and arcs incoming *d*(*k*). This function considers only arcs adjacent to the failed arc *a*. Arcs remote to *k*, which can block some flow of the failed arc during the rerouting process, aren't taken into account. Therefore, L_k only estimates the flow of arc *k* lost after local rerouting. Function L_k is a lower bound of the flow of arc *k* lost due a failure of *k*.

$$L_k(\underline{f},\alpha) = \alpha L_k^{\text{in}}(\underline{f}) + (1-\alpha)L_k^{\text{out}}(\underline{f}) \qquad 0 \le \alpha \le 1 \qquad \textbf{(7)}$$

Theorem 1. *Function* $L_k(\underline{f},\alpha)$ (7) *is a convex function for any feasible flow* $\underline{f}$ *and any* α *such that* $0 \le \alpha \le 1$.

Due to limited space of the paper we do not present the formal proof.

Using $L_k(\underline{f},\alpha)$ we can define a function $L(\underline{f},\alpha)$ that shows preparation of the whole network to the local rerouting after a failure of any single arc. We assume that probability of the arc failure is the same for all arcs. Therefore, probability is not included in this function.

$$L(\underline{f},\alpha) = \sum_{k\in A} L_k(\underline{f},\alpha) = \alpha \sum_{k\in A} L_k^{\text{in}}(\underline{f}) + (1-\alpha)\sum_{k\in A} L_k^{\text{out}}(\underline{f}) \qquad 0 \le \alpha \le 1 \qquad \textbf{(8)}$$

Theorem 2. *Function* $L(\underline{f},\alpha)$ (8) *is a convex function for any feasible flow* $\underline{f}$ *and any* α *such that* $0 \le \alpha \le 1$.

Proof. It is sufficient to notice that the function $L(\underline{f},\alpha)$ is a sum of convex functions $L_k(\underline{f},\alpha)$ over all arcs $k \in A$. According to Theorem 1 the function $L_k(\underline{f},\alpha)$ is convex for any feasible flow $\underline{f}$ and any α such that $0 \le \alpha \le 1$. Consequently, the function $L(\underline{f},\alpha)$ is convex. This completes the proof. □

Corollary 1. $L(\underline{f},\alpha)$ *is a continuous, non-decreasing, piece-wise linear function for any feasible flow* $\underline{f}$ *and any* α *such that* $0 \le \alpha \le 1$*. The function* $L(\underline{f},\alpha)$ *is differentiable except points for which one of the following condition holds*

$$g_{o(k)}^{\text{out}} = (e_{o(k)}^{\text{out}} - c_k) \qquad k \in A \qquad \textbf{(9)}$$

$$g_{o(k)}^{\text{in}} = (e_{o(k)}^{\text{in}} - c_k) \qquad k \in A \tag{10}$$

3 Related Work

In this section we present two functions proposed in the literature for optimization of primary routes using the local restoration and compare these functions with the function developed in previous section. In local restoration after a failure of the *k*-th arc all circuits using the failed arc must be rerouted around the arc *k*. In order to estimate the amount of the restored flow the maximum flow algorithm can be applied. The maximum flow criterion denotes the theoretical maximal rerouting capacity. The failed arc *k* is removed from the network. Next, the maximum flow between the origin and destination node of the failed arc is calculated taking into account spare capacity of network's arcs. Let *MF*(*k*) denote flow of the failed arc a restored by the maximum flow method. The flow lost using the maximum flow rerouting is given by $\varepsilon\big(f_k - MF(k)\big)$. If the *MF*(*k*) is greater than f_k, no flow is lost. Otherwise, $\big(f_k - MF(k)\big)$ flow is lost.

Another function applied for calculation of lost flow after the local rerouting is based on k-shortest paths (KSP) algorithm. The failed arc *k* is removed from the network. Next, the KSP algorithm finds k-successively shortest disjoint paths between the origin and the destination node of the failed arc. These paths one by one are saturated with flow of the failed arc and are used for restoration of flow f_k. The fraction of the flow f_k not restored during running KSP algorithm is lost. Let *KSP*(*k*) denote flow of the failed arc a restored by the KSP method. The flow lost after failure of arc *k* using the KSP rerouting is given by $\varepsilon\big(f_k - KSP(k)\big)$. Consequently, the lost flow after a failure of any single arc using KSP approach is calculated similarly to (8)

$$L_{KSP}(\underline{f}) = \sum_{k \in A} \varepsilon(f_k - KSP(k)) \tag{11}$$

Some previous authors have introduced similar approach for local rerouting of ATM network and formulated a problem of primary routes assignment with the objective function of lost flow using the KSP rerouting [6].

Authors of [3] compared maximum flow and KSP strategies using simulation methods. KSP restoration offers performance 99.9% of that from Max Flow. The advantage of KSP is time complexity of O(*n*log*n*) obtained for one link compared to maximum flow O(n^3) using the centralized restoration by a single processor computation. Function L_k (7) requires only O(*n*) time complexity. However, as mentioned above, the function (8) is a lower bound of the lost flow calculated using the maximum flow or KSP method. The function *L* takes into consideration only arcs adjacent to the failed arc *k*. Other arcs remote to *k* are not considered in this function.

Another advantage of the function $L(\underline{f},\alpha)$ is convexity. There are many algorithms developed for nonlinear convex multicommodity flow problems. Since the function L is convex, it can be easily applied for optimization of network flows in survivable networks. According to [5], the most popular algorithm for optimization of nonlinear convex m.c. flow problems is Flow Deviation (FD) - method proposed in [4] and applied to comparable problem in [2]. A comprehensive list of other algorithms can be found in [7].

4 Algorithm for Flow Assignment

In order to solve the problem of static primary routes assignment using as objective function $L(\underline{f},\alpha)$ we develop a heuristic algorithm based on the non-bifurcated FD algorithm proposed in [4].

With the purpose of making easier the consideration we define a new function

$$\varpi(x)=\begin{cases}0 & \text{for} \quad x\le 0\\ 1 & \text{for} \quad x>0\end{cases} \tag{12}$$

The FD algorithm uses an arc metric, which is a derivative of the objective function. Since according to the Corollary 1, the function $L(\underline{f},\alpha)$ is not differentiable everywhere, we introduce the following metrics of arc k

$$l_k^{\text{out}}(\underline{f})=\varpi\left(g_{o(k)}^{\text{out}}-(e_{o(k)}^{\text{out}}-c_k)\right) \tag{13}$$

$$l_k^{\text{in}}(\underline{f})=\varpi\left(g_{o(k)}^{\text{in}}-(e_{o(k)}^{\text{in}}-c_k)\right) \tag{14}$$

$$l_k(\underline{f},\alpha)=1+\alpha l_k^{\text{in}}(\underline{f})+(1-\alpha)l_k^{\text{out}}(\underline{f}) \qquad 0\le\alpha\le 1 \tag{15}$$

Note that $\alpha l_k^{\text{in}}(\underline{f})+(1-\alpha)l_k^{\text{out}}(\underline{f})$ is a derivative of the function $L_k(\underline{f},\alpha)$ (7) except points for which $g_{o(k)}^{\text{out}}=(e_{o(k)}^{\text{out}}-c_k)$ or $g_{o(k)}^{\text{in}}=(e_{o(k)}^{\text{in}}-c_k)$. In these points the function $l_k(\underline{f},\alpha)$ is equal to the left-sided derivative of $L_k(\underline{f},\alpha)$. Note that for less loaded networks metrics (13) and (14) are equal to 0. Therefore, we introduce to the formula (15) the hop number.

Algorithm FDP

Let $\underline{f}^1$ denote a feasible flow containing routes for all p circuits to be established. In order to find $\underline{f}^1$ we can apply an algorithm based on the initial phase of the FD algorithm [4]. Let $L(\underline{g},\alpha)$ denote value of the L (8) function for a feasible flow $\underline{g}$. We start with $r:=1$.

Step 1. Find a flow $SR(\underline{f}^r)$ defined as the set of shortest routes under the metric $l_k(\underline{f},\alpha)$ for all circuits. Set $i := 1$ and go to step 2.

Step 2. Let $\underline{g} = \underline{f}^r$.

a) Find $\underline{v}$ from $\underline{g}$ by deviating flow of circuit i to the shortest route given by $SR(\underline{f}^r)$. Routes for other circuits except circuit i remain unchanged.

b) If $\underline{v}$ is a feasible flow and $L(\underline{v},\alpha) < L(\underline{g},\alpha)$ set $\underline{g} = \underline{v}$.

c) If $i = p$ go to step 3. Otherwise, set $i := i+1$ and go to step 2a.

Step 3. If $\underline{g} = \underline{f}^r$ stop the algorithm, since the solution cannot be improved. Otherwise, set $r := r+1$, $\underline{f}^r = \underline{g}$ and go to step 1.

Theorem 3. *Algorithm FDP converges in a finite number of steps and constitutes a feasible solution.*

Proof. The main idea of the FDP algorithm is as follows. We start with a feasible flow $\underline{f}^1$. For each considered flow $\underline{f}^r$ we calculate $SR(\underline{f}^r)$ containing the shortest routes according to the metric $l_k(\underline{f},\alpha)$ (Step 1). Next, we try to improve the solution by deviation of one selected circuit to another route (Step 2). Since there are a finite number of non-bifurcated flows, the algorithm converges in a finite number of steps. Repetitions of the same flow are impossible due to the stopping condition (Step 3). We assume that the initial flow $\underline{f}^1$ is feasible. Next, in Step 2b we check whether the new flow $\underline{v}$ is feasible. If $\underline{v}$ is not feasible it is not analyzed further. Therefore, the algorithm FDP constitutes a feasible solution. □

We use the FDP algorithm also for optimization using other functions: lost flow using the KSP rerouting given by (11) and the overall network flow given by the sum of all arcs' flows. Clearly, we must modify the FDP according to these functions. For the former function we apply the following metric

$$l_k(\underline{f}) = 1 + \omega(KSP(k)) \tag{16}$$

For the flow function we use the traditional hop number metric

$$l_k(\underline{f}) = 1 \tag{17}$$

5 Results

The algorithm proposed in previous section was coded in C, and the program was run on an IBM-compatible PC with 2GHz Intel processor and 512 MB of RAM. Throughout the experiments, three objective functions were examined: lost flow in link given by (8), lost flow using the KSP rerouting (11) and the function of overall network flow. For the sake of simplicity in presentation of results, we refer to these functions in this section LFL, KSP and Hop, respectively. Also in tables and figures we use these names. The FDP algorithm is run for the same network and demand pattern three times with different objective functions. Labels FDP_LFL, FDP_KSP and FDP_Hop are used to denote the FDP algorithm applying functions LFL, KSP and Hop, respectively. In all cases the same starting solution found by the initial phase of the FD algorithm is applied. We assume that in function (8) α =0.5. We also show results given by the initial phase of the FD algorithm. Results presented in this section are obtained from simulations on 6 sample networks. Name of each network indicates the number of links in the network.

Table 1. Parameters of tested networks

Name of network	**114**	**128**	**144**	**162**	**180**	**200**
Number of nodes	36	36	36	36	36	36
Number of links	114	128	144	162	180	200
Node degree (average)	3.17	3.56	4.00	4.50	5.00	5.56
Node degree (minimum)	2	3	3	3	4	4
Node degree (maximum)	5	6	6	6	7	7
Number of tests	20	18	16	15	14	14
Flow requirement (minimum)	45	75	77	92	108	138
Flow requirement (maximum)	64	92	92	106	121	151

Table 1 summarizes the parameters of all sample networks. The first column specifies the name of the parameter, next columns includes values of these parameters for each network. Let bandwidth unit (BU) denote an arbitrary unit of bandwidth, for instance 1 Mb/s. We assume that for all networks capacity of each link is 5000 BU. Since, according to theoretical analysis presented above, the function of link lost flow depends on the node degree; we selected to numerical experiments networks with different values of the average node degree. In the experiment 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 is 1260. Each demand is defined by: the source node, destination node and flow requirement. To make clear the effectiveness of function LFL in response to a varying traffic demand, several demand patterns are examined for each network. The flow requirement for all demands is the same. For instance, for network 114 we perform 20 simulations starting with flow requirement of each demand equal to 45 BU, the biggest value of flow requirement is 64 BU.

We introduce the following parameters to present results. Since the simplest performance indicator to show preparation of the network to the local rerouting is the KSP function given by (11) defined in section 4, in order to compare performance of three tested functions we use the *normalized loss* (NL) function calculated using the KSP function. NL is defined as a unit of a normalized flow where 100 NL is equal to the total flow in the network. For instance, if L_{KSP}=100 and the total flow in the network is 500, the normalized flow equals 20 NL. Also the concept of *average link utilization* (AVLU) is used to describe the simulation results. The AVLU parameter, which indicates the network load, is defined as the proportion of the total flow in the network summed over all links and the total capacity of all networks links.

Table 2. The aggregate normalized loss obtained for various functions

Networks	FDInit	LFL	KSP	Hop
All	12.47	12.19	11.99	13.36
114	12.68	10.88	10.97	13.80
128	28.16	27.95	26.81	30.33
144	15.80	15.72	15.59	16.90
162	12.52	12.44	12.38	13.49
180	7.55	7.55	7.55	7.70
200	6.71	6.71	6.71	6.96

In Table 2 we report performance of the initial phase of FD and three functions: LFL, KSP and Hop in terms of the normalized loss. Results are aggregated over all tests performed for a given network. Generally, we found the KSP approach to be superior to the other approaches. However, for the network 114 the LFL function gives better results. Summary for all networks shows that the difference between LFL and KSP is lower then 1.7%. FDInit and Hop yield much worse solutions.

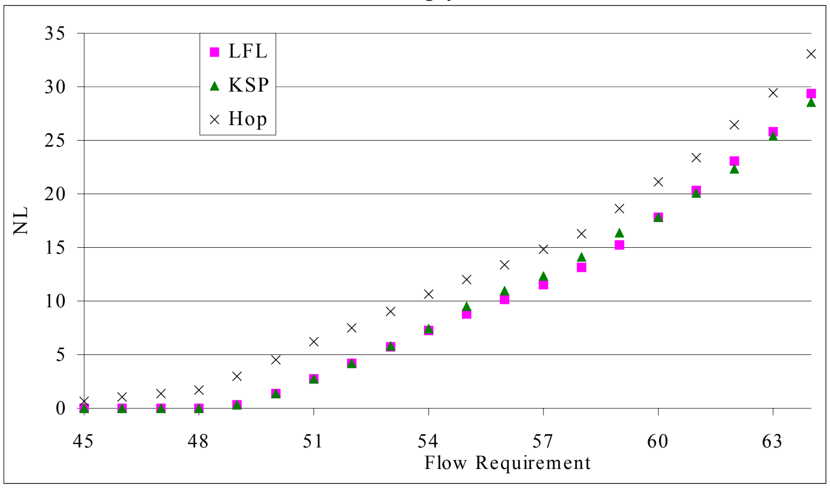

Fig. 1. Performance of various approaches in terms of the normalized flow for network 114

Fig. 1 shows the detailed performance of the three functions for network 114. The *x*-axis represents the flow requirement and the *y*-axis represents the normalized loss. The general trend is that both the LFL and KSP functions yield similar results while the Hop function provides much worse performance. Apparently, when the flow requirement increases, more flow is lost due to failure of any single arc.

In Table 3 we report CPU time taken for running algorithm FDP for all three functions. The time does not include time to do I/O for input of various files and design output. For each network we sum times over for all tested traffic patterns. Times are given in seconds and do not include the calculation time of the initial phase of the FD. We can see that the FDP_LFL performs 37 times faster then the FDP_KSP. This can be easily explained by the time complexity of both approaches discussed in Section 4.

To calculate the KSP function we must find k-shortest paths, which is much more time consuming then the calculation of the LFL function.

Table 3. The aggregated decision times of the FDP algorithm obtained for various functions

Networks	FDP_LFL [s]	FDP_KSP [s]	FDP_Hop [s]
All	222	8119	207
114	60	1057	33
128	70	1585	39
144	30	1198	31
162	28	1432	34
180	17	1195	32
200	18	1653	39

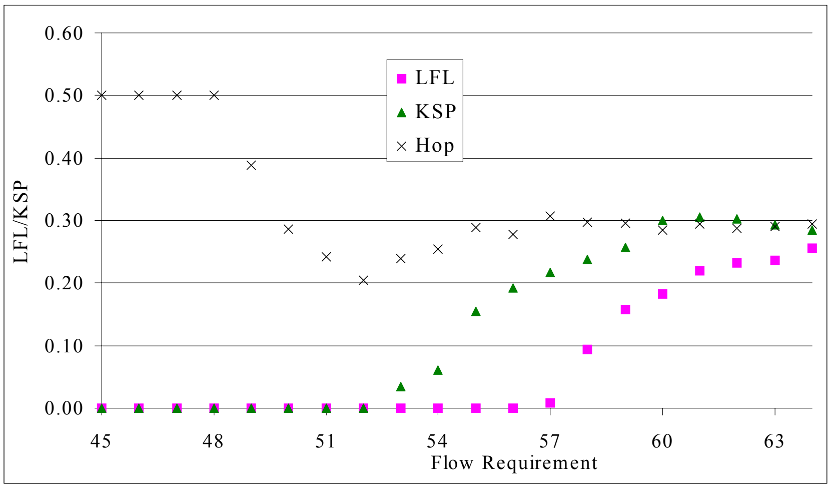

Fig. 2. The LFL/KSP ratio obtained for various functions for the network 114

One of the most interesting questions raised during the simulation on all networks was how the LFL function approximates the KSP function. As mentioned above, the LFL function given by (8) is a lower bound of the lost flow using the KSP rerouting (11). In Fig. 2 we compare performance of functions LFL, KSP and Hop. The *x*-axis represents the flow requirement. The *y*-axis represents the LFL/KSP ratio calculated for each traffic demand pattern. The data for each point in the figure are obtained by running the FDP algorithm using one of three functions for a given demand pattern in network 114. We studied the performance of the functions for increasing traffic load, examining the evolution of the network status toward a saturation condition. We can see that for the FDP_Hop algorithm values of LFL/KSP are the largest. It is due to the fact that the Hop function does not optimize the network in terms of the lost flow for local rerouting. Therefore, the FDP_HOP yields values of LFL and KSP functions far from minimal values. Since there is a correlation between these two functions, the ratio LFL/KSP for FDP_Hop function is relatively large. Under low load conditions, algorithm FDP_LFL and FDP_KSP give much lower values of LFL/KSP then the FDP_Hop. This implies that optimization of network flow according to LFL or KSP functions significantly reduces the lost flow, especially for low congested networks. Under high, near saturation, loads, all functions tested have similar performance. This is because for all functions, the FDP algorithm uses the same starting solution and for high loaded networks only a small part all possible solutions are feasible. Consequently, all three functions produce similar results.

Generally the experimental results are consistent with theoretical analysis. The LFL function is a lower bound of the KSP function. However, when the average node degree of tested network grows, the ratio LFL/KSP decreases. In networks 180 and 200 for all traffic demand patterns the LFL is 0, while values of KSP are larger than 0. Hence, the LFL function should be rather applied for networks with average node degree lower than 4.5. For highly connected networks, the LFL function does not estimate KSP function properly.

Due to limited paper space, other important results of our study are described briefly as follows:

1. Each function tested gives the best results calculated by the FDP algorithm applying this function. Values of KSP and Hop functions obtained for various versions of FDP are quite similar. However, the FDP_LFL can find values of LFL function 50% better in average than the FDP applying one of two other functions.
2. Minimizing overall network flow does not guarantee good restoration performance; sometimes demands should use longer routes in order to omit highly congested areas of the network.
3. If we split each demand to 2, 3 or 4 demands, the performance gain of any of three functions is relatively small.
4. The curves of the LFL function obtained during simulations are similar to analytically plotted curves.

6 Conclusion

The two main contributions in this paper are the definition of a new function for optimization of m.c. flows in survivable connection-oriented networks and experimental simulations performed to examine this function. We have studied how to simplify the optimization of m.c. flows in survivable networks using the LFL function given by (8). Moreover, we have compared this function with other functions proposed by previous authors.

We found the KSP approach to be superior to the LFL and Hop functions. However, the calculation time for the KSP function is much greater then the calculation time for two other functions. In addition, the gap between results of KSP and LFL is very small. Concluding, the function (8) developed in this work can be effectively applied for design of primary routes in order to prepare the network for local rerouting. The performance evaluation reveals that the LFL function yields results close to the KSP approach proposed in previous works. However, the time complexity is much lower and consequently, the calculation time is shorter.

There are also several shortcomings of the LFL that are worth to be mentioned. First, the presented approach relies on the information adjacent to the failed link. Therefore, it is a local metric that does not take into account global information on the network flows. Second, as shown in previous section, the LFL performs well for networks with the average node degree lower than 4.5 .

For design of computer networks we can use offline or online algorithms. The function $L(\underline{f},\alpha)$ is applicable in both types of algorithms. The application for offline algorithms is shown above. In online algorithms, e.g. dynamic routing algorithms, we can use $l_k(\underline{f},\alpha)$ (15) as an arc metric for computation of shortest routes.

References

1. Ayanoglu, E., Gitlin, R.: Broadband Network Restoration. IEEE Comm. Magazine, 7 (1996) 110-119
2. Burns, J., Ott, T., Krzesinski, A., Muller, K.,: Path selection and bandwidth allocation in MPLS networks. Performance Evaluation, 52 (2003) 133-152.
3. Dunn, A., Grover, W., MacGregor, M.: Comparison of k-Shortest Paths and Maximum Flow Routing for Network Facility Restoration. IEEE JSAC, 1 (1994) 88-99
4. Fratta, L., Gerla, M., Kleinrock, L.: The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design. Networks (1973) 97–133
5. Kasprzak, A.: Exact and Approximate Algorithms for Topological Design of Wide Area Networks with Non-simultaneous Single Commodity Flows. Lectures Notes In Computer Science, LNCS 2660, (2003) 799-808
6. Murakami, K., Kim, H.: Virtual Path Routing for Survivable ATM Networks. IEEE/ACM Transactions on Networking, 2 (1996) 22-39
7. Ouorou, A., Mahey, P., Vial, J.-Ph.: A survey of algorithms for convex multicommodity flow problems. Management Science, 1 (2000) 126-147
8. Sharma, V., Hellstrand, F. (ed.): Framework for MPLS-based Recovery. RFC 3469 (2003)
9. Walkowiak, K.: A New Approach to Survivability of Connection Oriented Networks. Lectures Notes In Computer Science, LNCS 2657, (2003) 501-510
10. Walkowiak, K.: A Branch and Bound Algorithm for Primary Routes Assignment in Survivable Connection Oriented Networks. Computational Optimization and Applications 2 (2004) 149-171