FastSurv: A New Efficient Local Search Algorithm for Survivable Routing in WDM Networks

Frederick Ducatelle and Luca M. Gambardella
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH-6928 Manno-Lugano, Switzerland
{frederick,luca}@idsia.ch

***Abstract*— In IP-over-WDM networks, a logical IP network has to be routed on top of a physical optical fiber network. An important challenge is to make this routing survivable. We call a routing survivable if no single physical link failure can disconnect the logical topology. In this paper we present FastSurv, a local search algorithm for survivable routing. FastSurv works in an iterated way: after each iteration it learns more about the structure of the logical graph and in the next iteration it uses this information to improve its solution. We also extend the algorithm to take link capacity constraints into account. We show that our simple algorithm can produce better and faster results than current state-of-the-art algorithms.**

I. Introduction

In IP-over-WDM networks, an IP network is placed as a logical topology on top of the physical topology of an optical network. Each logical IP link is routed on a path in the optical network. Using wavelength division multiplexing (WDM) [7], one physical link can carry several logical links, each on a different wavelength. The problem of routing logical links on the optical network and assigning wavelengths to them is called the routing and wavelength assignment (RWA) problem [14]. In what follows we refer to logical IP links as clear-channels and to physical links simply as links.

There are two main approaches to protection in IP-over-WDM networks: WDM protection and IP restoration [11]. In WDM protection, a physically disjoint backup path is reserved for each clear-channel. This can only be done at the cost of hardware redundancy. IP restoration can be a cheaper option. In this approach failures are detected by IP routers, which adapt their routing tables. So when a link breaks, data traffic which used to go over clear-channels using this link is rerouted over other clear-channels which do not use the link.

An important problem for IP restoration in WDM networks is caused by the fact that each link usually carries several clear-channels. This means that a link failure causes a number of clear-channels to go down at the same time. It is possible that the logical network becomes disconnected due to these concurrent failures and IP restoration becomes impossible. This is called failure propagation [3]. To avoid it, the clear-channels should be routed in such a way that no single link failure can disconnect the logical network. This NP-complete combinatorial problem is often called survivable routing.

An example is given in figure 1. The clear-channels of IP network (B) need to be routed on WDM network (A). The routing given in (C) is not survivable: when link $(4, 5)$ breaks, both clear-channels c and d get disconnected, leaving the logical network in two parts: one containing node 4 and the other containing the other nodes. Routing (D) on the other hand is survivable: no matter which link breaks, the logical network always remains connected.

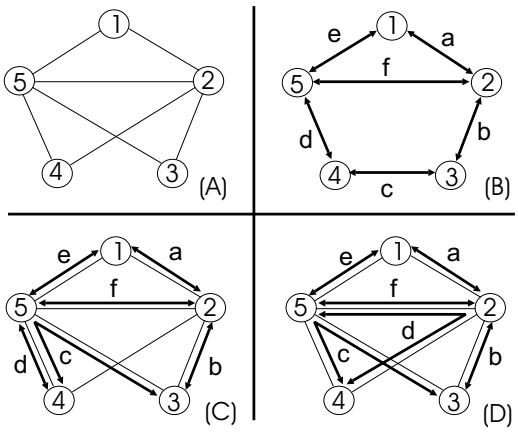

Fig. 1. (A) Physical topology. (B) Logical topology (C) An unsurvivable mapping. (D) A survivable mapping.

In this paper we present FastSurv, a fast local search algorithm for survivable WDM routing. This paper is organized as follows. In section II, a detailed problem description is given, and related work is discussed. In section III the algorithm is explained and in section IV test results are presented.

II. Problem definition and related literature

We assume that a physical WDM topology and a logical IP topology are given. The clear-channels of the logical topology have to be routed on physical paths in such a way that no single link failure can disconnect the logical network. Note that this is a subproblem of the overall logical topology design problem, in which the logical topology needs to be generated before it can be routed onto the physical topology [2], [9].

A. Formal problem definition

The input for the problem consists of two undirected graphs (each undirected link is made up of two unidirectional optical fibers): $G_{ph}(V, L)$ representing the physical topology, and $G_{lo}(V, C)$ representing the logical topology. V is the set of nodes in the network, and N is the number of nodes $|V|$. L is the set of links of the physical network. It is represented as an $N \times N$ matrix, for which entry $l_{mn} = 1$ if there is a link

between nodes m and n and $l_{mn} = 0$ otherwise. C is the set of clear-channels. $c_{sd} = 1$ if there is a clear-channel between nodes s and d. A solution is obtained when each clear-channel of C is mapped onto a path in the physical graph. So, for each $c_{sd} \in C$, an $N \times N$ mapping matrix $R^{sd} = (r^{sd}_{mn})$ needs to be found, for which $r^{sd}_{mn} = 1$ if l_{mn} is on the path of c_{sd}.

A mapping is evaluated by disconnecting the links l_{mn} of physical graph G_{ph} one by one. For each l_{mn}, all clear-channels c_{sd} for which $r^{sd}_{mn} = 1$ are disconnected in G_{lo}, giving rise to a partial logical graph G^{mn}_{lo}. Using the clear-channels of this partial graph, we try to find a path connecting the endpoints s and d of each of the disconnected clear-channels c_{sd}. If s and d are disconnected in G^{mn}_{lo}, we say that c_{sd} is unsurvivable on l_{mn}, and call $\{c_{sd}, l_{mn}\}$ an unsurvivable pair. In this way an unsurvivability matrix $U = (u^{sd}_{mn})$ is obtained, in which u^{sd}_{mn} is 1 if c_{sd} is unsurvivable on l_{mn} and 0 otherwise. The function to minimize is given in equation 1.

$$F(R) = \sum_{mn} \sum_{sd} u^{sd}_{mn} \tag{1}$$

If we consider wavelength assignment, there are two extra constraints: the distinct wavelength constraint and the wavelength continuity constraint [10]. The distinct wavelength constraint states that each clear-channel c_{sd} routed on link l_{mn} should use a different wavelength. Since each link can carry a limited number of wavelengths, this comes down to a capacity constraint. To reflect this, we adapt the matrix of the links L: if there is a physical link between m and n, l_{mn} contains the capacity of the link (instead of just 1), and otherwise it is 0. The wavelength continuity constraint states that a clear-channel c_{sd} should use the same wavelength on all the links along its path from s to d. This constraint can be relaxed if nodes in the network are capable of wavelength conversion.

In this paper we present an algorithm for the survivable routing algorithm without link capacity constraints, and an extension for the case with link capacity constraints. We do not consider the wavelength continuity constraint.

B. Related literature

There have been a number of publications on survivable WDM routing as it is defined in subsection II-A. In [1]–[3], tabu search solutions are presented. In [5] an ILP formulation of the problem is proposed, and in [6] two heuristic solution methods are derived using relaxations of the ILP. A simple heuristic method is presented in [12]. Other papers consider related problems, or special cases of the survivable WDM routing problem. The algorithm proposed in [9] attempts to look at the complete logical topology design problem: given a physical topology and a traffic demand matrix, it generates a logical topology and tries to route it survivably using a very simple heuristic. Also the work in [11] looks at the full logical topology design. Reference [4] and other papers by the same authors treat the special case where the physical topology is a ring. In [13] the authors describe a solution for the special case where the logical topology is a ring.

III. The FastSurv algorithm

In the first subsection we present the basic FastSurv algorithm for survivable routing. In the second subsection this algorithm is extended to account for link capacity constraints.

A. The basic FastSurv algorithm

An overview of the algorithm is given in figure 2. It starts from an initial solution obtained with a simple heuristic and tries to improve it in a number of iterations. Solutions are evaluated as explained in subsection II-A, and the algorithm reroutes all clear-channels which were unsurvivable on at least one of the links. While rerouting the algorithm tries to avoid routing together clear-channels which were unsurvivable when they were routed together on the same link in previous iterations. The algorithm ends when the full mapping is survivable or when a maximum number of iterations is reached.

To obtain an initial solution, the clear-channels are routed on the physical graph one by one in random order, using a shortest path algorithm in which the cost of a path depends on the clear-channels which are already placed on the physical graph. Suppose we have routed i clear-channels, and are calculating a shortest path for the $(i+1)^{th}$ clear-channel. If we indicate by $C^i \subset C$ the subset of the i first routed clear-channels, then the cost $cost^{i+1}(l_{mn})$ for clear-channel $i+1$ to use link l_{mn} is equal to the number of clear-channels c_{sd} from C^i which use l_{mn}, as indicated in formula 2. This simple algorithm avoids that some links carry a lot more clear-channels than others, which would make them more vulnerable with respect to survivability (a similar strategy is used in [12]).

$$cost^{i+1}(l_{mn}) = \sum_{c_{sd} \in C^i} r^{sd}_{mn} \tag{2}$$

After the initial solution is constructed, FastSurv reroutes at every new iteration all clear-channels which were unsurvivable on at least one link in the solution of the previous iteration. In order to explain the algorithm, we first define a number of variables. For notational purposes, we use in this subsection indices i and j to refer to clear-channels $c_{s_i d_i}$ and $c_{s_j d_j}$. $R^i(t) = [r^i_{mn}(t)]$ is the mapping matrix for clear-channel i in the solution of iteration t and $U(t) = [u^i_{mn}(t)]$ represents the unsurvivability matrix of the solution obtained in iteration t. So the clear-channels to be rerouted after iteration t are the ones for which $\sum_{mn} u^i_{mn}(t) > 0$. $a_{ij}(t)$ is the number of times that clear-channels i and j were routed together on the same link in the solution of iteration t, formally defined as $\sum_{mn} r^i_{mn}(t) r^j_{mn}(t)$. $b_{ij}(t)$ is the number of times that clear-channels i and j were both unsurvivable when they were routed together on the same link, formally defined as $\sum_{mn} u^i_{mn}(t) u^j_{mn}(t)$. Dividing $b_{ij}(t)$ by $a_{ij}(t)$, one obtains a ratio which can be seen as an estimate (based on the experience of iteration t) of the probability that clear-channels i and j both become unsurvivable when they are routed together. Based on this ratio, we finally define $p_{ij}(t)$, which is the running average of these probability estimates. It is updated after each iteration t according to formula 3 (in this formula, $\alpha \in [0, 1]$).

1) Create an initial solution
2) Evaluate the solution, indicating which clear-channels were unsurvivable on at least one link
3) Finish if the solution is survivable or the maximum number of iterations is reached
4) Update the information about which clear-channels were unsurvivable when routed together on the same link
5) Reroute the unsurvivable clear-channels of the previous solution, using the information of step 4
6) Go back to step 2

Fig. 2. An overview of the basic FastSurv algorithm

$$p_{ij}(t) = \begin{cases} \alpha p_{ij}(t-1) + (1-\alpha)\frac{b_{ij}(t)}{a_{ij}(t)} & \text{if } a_{ij}(t) > 0 \\ p_{ij}(t-1) & \text{if } a_{ij}(t) = 0 \end{cases} \quad (3)$$

So in $p_{ij}(t)$ the algorithm keeps a moving estimate of the probability that i and j will be unsurvivable when they are routed together. These probability estimates are used to reroute clear-channels: a shortest path algorithm is applied in which the cost of a path for a clear-channel i is the probability that i will be unsurvivable along the path. The probability $Prob^i_{path}(t)$ that i will be unsurvivable on a path is the probability that it will be unsurvivable on at least one link of the path. The probability $Prob^i_{mn}(t)$ that i will be unsurvivable on a link l_{mn} of the path is the probability that i will be unsurvivable when routed together with any of the other clear-channels which use l_{mn} (clear-channels which were not chosen for rerouting or which are already rerouted). $Prob^i_{mn}(t)$ is estimated according to formula 4 and $Prob^i_{path}(t)$ according to formula 5.

$$Prob^i_{mn}(t) = 1 - \prod_{j \text{ on } l_{mn}} (1 - p_{ij}(t)) \quad (4)$$

$$Prob^i_{path}(t) = 1 - \prod_{l_{mn} \text{ on } path} (1 - Prob^i_{mn}(t)) \quad (5)$$

The algorithm described here is based on the observation made in [5] that a routing is survivable if and only if no link is shared by all clear-channels belonging to a cut set of the logical network. A cut set of a network if defined by a cut of the network: a cut of the network is a partition of the nodes V into two sets S and $V - S$, and the cut set defined by this cut is the set of edges which have one endpoint in S and one in $V - S$. In [5], this observation is used to formulate an ILP for the survivable routing problem: for each clear-channel and for each cut set of the logical network, a constraint is added to the ILP. This leads to exact solutions, but also to very long run times. In [6], the authors propose two relaxations to their ILP, in which they do not include all cut sets. This speeds up the algorithm, but sometimes leads to suboptimal solutions.

In FastSurv, we approximate the notion of the cut sets by the probability estimates p_{ij}. A look at the example logical graph of figure 1(B) can help explain the meaning of p_{ij}. In this graph, $\{b, d\}$ forms a cut set of size two. b and d are both unsurvivable each time they are routed together, and therefore p_{bd} will be 1. p_{be} on the other hand will normally be lower than 1, since $\{b, e\}$ does not form a cut set. $\{b, e\}$ is part of the larger cut set $\{b, e, f\}$, and therefore b and e are both unsurvivable if they are also routed together with f. So, when two clear-channels i and j do not form a cut set of their own, p_{ij} depends on which other clear-channels are also routed together with them. If, going back to the example, the situation is such that f is often routed together with b and e (e.g. because of the structure of the physical graph), this will be reflected in high values for $p_{b,e}$, $p_{b,f}$ and $p_{e,f}$, and FastSurv will take the cut set into account. If on the other hand f is rarely routed together with b and e, all three values will be low: the cut set will be considered unimportant and FastSurv will ignore it. So the pairwise probability estimates can be seen as a simple way to focus only on important cut sets.

1) Create an initial solution
2) Evaluate survivability and link capacity constraint violations
3) Finish if a solution is found or the maximum number of full iterations is reached
4) Reroute unsurvivable clear-channels
5) Evaluate survivability
6) Go to step 8 if the solution is survivable or the maximum number of survivable routing iterations is reached
7) Return to step 4
8) Evaluate link capacity constraint violations
9) Reroute clear-channels which are on overfull links
10) Evaluate link capacity constraint violations
11) Go to step 2 if there is no reduction in capacity violations
12) Return to step 9

Fig. 3. An overview of the extended FastSurv algorithm

B. The extended FastSurv algorithm

The extended FastSurv algorithm takes link capacity constraints into account. An overview is given in figure 3. A full iteration of the algorithm consists of a number of iterations of the basic FastSurv algorithm (which we call survivability iterations) and then a number of iterations to decrease the number of capacity constraint violations (which we call capacity iterations). The survivability iterations are run until the routing is survivable or the maximum number of iterations (empirically set to 2) is reached, and the capacity iterations until there is no further reduction in the number of capacity constraint violations. The capacity constraint violations are evaluated by taking the sum of the overcapacity over all links, as indicated in formula 6. The full iterations (the combination of survivability and capacity iterations) are repeated until a survivable routing without any overcapacity is found, or until a maximum number of iterations is reached.

$$F_c(R) = \sum_{mn} max\Big(\Big(\sum_{sd} r^{sd}_{mn}\Big) - l_{mn}, 0\Big) \quad (6)$$

In the capacity iterations, the clear-channels to be rerouted are chosen randomly from among the clear-channels which are routed over links with overcapacity. More formally, a clear-channel c_{sd} is considered for rerouting if $\sum_{mn}(r^{sd}_{mn} \times max((\sum_{s'd'} r^{s'd'}_{mn}) - l_{mn}, 0)) > 0$. The maximal number of clear-channels rerouted in one iteration is empirically set to 10 percent of the total number of clear-channels in the graph.

The chosen clear-channels are removed from the routing matrix and rerouted one by one in random order. The rerouting is done with shortest path routing where, like in the heuristic used to obtain the initial solution for the basic algorithm, the cost of a link is equal to the total number of clear-channels on the link. However, if the number of clear-channels on the link is lower than the link's capacity, this cost is divided by the link's capacity. In this way overfull links get a much higher cost, and hence they are avoided. Formally, if $C^i \subset C$ is the set of the i clear-channels which are already routed (because they were not chosen for rerouting or because they have already been rerouted), the cost of using link l_{mn} for the $(i+1)^{th}$ clear-channel is given in formula 7. Also for the initial routing we now use formula 7 rather than 2. The capacity iterations have similarities with a local search for RWA described in [8].

$$cost^{i+1}(l_{mn}) = \begin{cases} \frac{\sum_{c_{sd} \in C^i} r^{sd}_{mn}}{l_{mn}} & \text{if } \sum_{c_{sd} \in C^i} r^{sd}_{mn} < l_{mn} \\ \sum_{c_{sd} \in C^i} r^{sd}_{mn} & \text{if } \sum_{c_{sd} \in C^i} r^{sd}_{mn} \geq l_{mn} \end{cases} \quad (7)$$

IV. Test results

In this section we describe test results we obtained with FastSurv. IV-A contains results for the basic FastSurv algorithm and IV-B for the extended FastSurv algorithm. Our programs are implemented in C++ and all tests were run on a PC with a 2.4 GHz Intel Pentium 4 processor.

A. Test results for the basic FastSurv algorithm

For the basic FastSurv algorithm, we made comparisons with the full and relaxed ILP methods of [6] and the tabu search algorithms of [1], [2]. In every test run FastSurv is given 100 iterations, with a random restart after every 10^{th} iteration in order to avoid getting stuck in a local optimum (since it is a local search algorithm). The tabu search algorithms are run with the parameters given in their respective papers.

For comparison with the ILP methods, we ran FastSurv on the problems which were used in [6]. The physical network is the 14-node, 21-link NSFNET. The logical topologies were randomly generated by the authors of [6]. There are networks of degree 3, 4, and 5, where a network of degree n is a network in which all nodes have degree n. For each degree 100 logical networks were generated. The algorithm is run once for each test problem, and the results are summarized per network degree in table I, where *Uns* is the number of problems for which no survivable solution was found, and *Time* is the average run time per problem in CPU seconds. ILP is the full ILP solution method, and relax-1 and relax-2 are the two relaxations of ILP, which use only a subset of the survivability constraints. The results show that FastSurv gives good results in very short times. The other approaches are much slower, and relax-1 is not able to solve all problems.

For comparison with the tabu search algorithms, we could not obtain the test problems used by the authors, nor the source code, so we use our own implementation of the algorithms presented in [1], [2]. Strictly speaking, only the tabu search of [1] (TS97) addresses the basic survivable routing problem, since the tabu search of [2] (TS98) also takes link capacities into account. Nevertheless, there are some differences in TS98 which make it more efficient. Therefore we made an adaptation of the algorithm which ignores link capacities (TS*03).

TABLE I
Results on NSFNET

Network Degree		3	4	5
FastSurv	Uns	0	0	0
	Time	0.0117	0.0155	0.0166
ILP	Uns	0	0	0
	Time	8.3	173	1157
Relax-1	Uns	10	0	0
	Time	1.3	1.5	2.0
Relax-2	Uns	0	0	0
	Time	1.3	1.5	2.0

TABLE II
Results on ARPA2 without link capacities

		TS97		TS*03		FastSurv	
		Time	Uns	Time	Uns	Time	Uns
Degree 3	1	4.34	0	0.31	0	0.07	0
	2	4.91	0	1.04	0	0.05	0
Degree 4	1	5.38	0	2.37	2	0.02	0
	2	4.75	0	0.33	0	0.02	0

TABLE III
Results on the 28-node random network without capacities

		TS97		TS*03		FastSurv	
		Time	Uns	Time	Uns	Time	Uns
Degree 3	1	10.37	0	1.21	0	0.04	0
	2	10.14	10	5.62	4	0.09	0
Degree 4	1	15.75	0	0.75	0	0.03	0
	2	15.08	0	0.69	0	0.04	0

TABLE IV
Results on the 49-node lattice networks without capacities

	TS97		TS*03		FastSurv	
	Time	Uns	Time	Uns	Time	Uns
1	304.13	10	512.40	10	2.30	10
2	463.07	0	37.23	0	0.15	0
3	323.96	10	513.36	10	2.26	10
4	455.22	10	896.05	9	0.29	0

We used the following physical networks: the 21-node, 26-link ARPA2 network and a 28-node 42-link random network, both used in [2], and four randomly generated 49-node 67-link partial lattice graphs.[1] Both for the ARPA2 and the 28-node random network, we randomly generated four logical graphs: two with average node degree 3 and two with average node degree 4. For the partial lattice networks, we used one logical graph per physical graph, randomly generated with an approximate average node degree of 4. For each combination of physical and logical graph we ran each algorithm 10 times.

The results are given in tables II-IV, where *Uns* is the number of runs in which no survivable solution was found and *Time* is the average run time per problem in CPU seconds. For the small ARPA2 network, all three algorithms always find an optimal solution (except TS*03 in two cases), but FastSurv has shorter run times. For the 28-node random graph and the 49-node partial lattice graphs this time difference gets bigger, and there are more differences in solution quality. These

[1]The partial lattice graphs are constructed by building a 7 by 7 full lattice graph, and taking away links randomly until 67 (40 %) are left.

TABLE V
RESULTS ON ARPA2 WITH LINK CAPACITIES

		TS98				FastSurv			
		Time	Au	Ac	Best	Time	Au	Ac	Best
Dg 3	1	2.69	0.0	0.3	0/0	0.13	0.0	0.0	0/0
	2	1.89	0.0	0.2	0/0	0.18	0.2	0.0	0/0
Dg 4	1	4.79	0.8	0.0	0/0	0.05	0.0	0.0	0/0
	2	4.90	0.0	0.4	0/0	0.31	0.0	0.3	0/0

TABLE VI
RESULTS ON THE 28-NODE RANDOM NETWORK WITH CAPACITIES

		TS98				FastSurv			
		Time	Au	Ac	Best	Time	Au	Ac	Best
Dg 3	1	16.64	0.0	2.5	0/2	1.07	0.0	2.0	0/2
	2	12.98	0.8	0.3	0/0	0.71	0.0	0.2	0/0
Dg 4	1	1.10	0.0	0.0	0/0	0.05	0.0	0.0	0/0
	2	2.75	0.0	0.0	0/0	0.10	0.0	0.0	0/0

TABLE VII
RESULTS ON THE 49-NODE LATTICE NETWORKS WITH CAPACITIES

	TS98				FastSurv			
	Time	Au	Ac	Best	Time	Au	Ac	Best
1	487.98	2.2	0.0	2/0	7.87	2.0	0.0	2/0
2	631.35	0.0	3.0	0/3	6.38	0.0	3.0	0/3
3	464.7	2.0	0.0	2/0	7.59	2.0	0.0	2/0
4	638.83	1.0	0.0	0/0	2.44	0.4	0.0	0/0

results suggest that FastSurv is faster, better and more scalable with respect to the number of nodes. One explanation why FastSurv is more scalable can be found in the complexity of the iterations. The tabu search algorithms try in each iteration to reroute each clear-channel separately, and pick the rerouting which improves the solution most. Routing a clear-channel has a complexity of $O(Dijkstra)$ (which is maximally $O(N^2)$), and evaluating a solution has a complexity of $O(N^4)$ (see [2]). Since this (a rerouting and an evaluation) needs to be done for each clear-channel in each iteration, then, if we express $|C|$ as αN,[2] the complexity of an iteration is $O(\alpha N(N^2 + N^4)) = O(N^5)$. In FastSurv, on the other hand, the clear-channels are rerouted using the probabilities, and the evaluation is done only once, at the end of each iteration, which means that one iteration has a complexity of $O(\alpha N \cdot N^2 + N^4) = O(N^4)$.

B. Test results for the extended FastSurv algorithm

For the extended algorithm we only compare to TS98, since it is the only one which takes link capacities into account. In these tests, FastSurv is given 150 iterations, without random restarts. We use the physical topologies of the previous section, but add to them a maximum capacity for each link. The link capacities are the same for each link of the network, but were set differently for different logical graphs. For ARPA2, link capacities were set to 7 for degree 3 graphs and to 8 for degree 4 graphs. For the 28-node network capacities were set to 5 for degree 3 graphs and to 6 for degree 4 graphs. For the partial lattice graphs capacities were set to 20. The logical graphs are the same as before. Results are presented in tables V-VII, where *Time* is the average run time per problem in CPU seconds, *Au* the average number of unsurvivable pairs (using eq. 1) and *Ac* the average overcapacity (using eq. 6). *Best* is the best solution over 10 runs, with left its number of unsurvivable pairs, and right its overcapacity. The best solution is the one with the lowest sum of these two values. FastSurv gives comparable to better results than TS98, and again its run times are much shorter. Also, FastSurv seems again more scalable with respect to the number of nodes.

[2] If α is the average connectivity per node, $|C|$ is a linear function of N.

V. CONCLUSIONS AND FUTURE WORK

We have presented FastSurv, a new local search algorithm for survivable routing in WDM networks, and compared it to existing state-of-the-art algorithms for this problem. FastSurv gave better results while using much shorter run times. The property of giving high quality results in a very short time can be important. As was pointed out in section II, routing a logical topology survivably on a physical network is part of a larger logical topology design algorithm, and a time-consuming algorithm there can considerably slow down the larger algorithm. In future work we plan to extend FastSurv to take into account multiple failures and wavelength continuity constraints. We also want to investigate the performance of the current algorithm better. In particular, we want to verify the scalability suggested by the test results in this work.

VI. ACKNOWLEDGEMENTS

We thank Gianni Di Caro, Patrick Thiran, Maciej Kurant and Aradhana Narula-Tam. This work is partially supported by the Swiss HASLER Foundation project DICS-1830.

REFERENCES

[1] J. Armitage, O. Crochat, and J.-Y. Le Boudec. Design of a survivable WDM photonic network. In *Proceedings of INFOCOM*, 1997.
[2] O. Crochat and J.-Y. Le Boudec. Design protection for WDM optical networks. *IEEE JSAC Special Issue on High-Capacity Optical Transport Networks*, 1998.
[3] O. Crochat, J-Y. Le boudec, and O. Gerstel. Protection interoperability for WDM optical networks. *IEEE Transactions on Networking*, 2000.
[4] H. Lee, H. Choi, S. Subramaniam, and H.-A. Choi. Survival embedding of logical topology in WDM ring networks. *Information Sciences : An International Journal, Special Issue on Photonics, Networking and Computing*, 2002.
[5] E. Modiano and A. Narula-Tam. Survivable routing of logical topologies in WDM networks. In *Proceedings of INFOCOM*, 2001.
[6] E. Modiano and A. Narula-Tam. Survivable lightpath routing: a new approach to the design of WDM-based networks. *IEEE JSAC*, 2002.
[7] B. Mukherjee. *Optical Communication Networks*. McGraw-Hill, 1997.
[8] N. Nagatsu, Y. Hamazumi, and K. I. Sato. Number of wavelengths required for constructing large-scale optical path networks. *Electronics and Communications in Japan, Part I - Communications*, 1995.
[9] A. Nucci, B. Sanso, T. G. Crainic, E. Leonardi, and M. A. Marsan. Design of fault-tolerant logical topologies in wavelength-routed optical IP networks. In *Proceedings of Globecom*, 2001.
[10] G. Rouskas. *Wiley Encyclopedia of Telecommunications*, chapter Routing and Wavelength Assignment in Optical WDM Networks. John Wiley and Sons, 2001.
[11] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee. Fault management in IP-over-WDM networks: WDM protection versus IP restoration. *IEEE JSAC*, 2002.
[12] G. H. Sasaki, C.-F. Su, and D. Blight. Simple layout algorithms to maintain network connectivity under faults. In *Proceedings of the 2000 Annual Allerton Conference*, 2000.
[13] A. Sen, B. Hao, B. H. Shen, and G. H. Lin. Survivable routing in WDM networks logical ring in arbitrary physical topology. In *Proceedings of the IEEE International Communication Conference ICC*, 2002.
[14] H. Zang, J. P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *SPIE Optical Networks Magazine*, 2000.