

A Scalable Algorithm for Survivable Routing in IP-over-WDM Networks

Frederick Ducatelle
Luca M. Gambardella

*Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH-6928 Manno-Lugano, Switzerland*

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 hereby is to make the routing survivable. We call a routing survivable if the connectivity of the logical network is guaranteed in case of a failure in the physical network. In this paper we describe FastSurv, a local search algorithm which can provide survivable routing in the presence of physical link failures. The algorithm can easily be extended for the case of node failures and multiple simultaneous link failures. In a large series of test runs, we show that FastSurv is much more scalable with respect to the number of nodes in the network than current state-of-the-art algorithms, both in terms of solution quality and run time.

1 Introduction

Optical fiber connections are becoming an increasingly popular technology for high-speed Wide Area Networks (WANs). This is mainly because they offer an enormous bandwidth, and because their bandwidth can easily be shared among different channels through the use of Wavelength Division Multiplexing (WDM) [9, 12]. In IP-over-WDM networks, an IP network is placed as a logical topology on top of the physical topology of the optical network. Each logical IP link needs to be routed on a path in the optical network. Thanks to the WDM technology, one physical link can carry several logical links, each on a different wavelength. The problem of setting up logical links by routing them on the optical network and assigning wavelengths to them is called the routing and wavelength assignment (RWA) problem. See [18] for an overview. In what follows we will refer to logical IP

links as clear-channels and to physical (optical) links simply as links.

Using high capacity links carrying multiple clear-channels is not without danger: in case of just a single link failure, a huge amount of data can get lost. Therefore a lot of attention is being paid to network protection. There are two main approaches to protection in IP-over-WDM networks: WDM level protection and IP level restoration [14]. In the case of WDM protection, a physically disjoint backup path is reserved for each clear-channel. This can only be provided at the cost of hardware redundancy. Therefore IP restoration can be a cheaper option. In this approach, no action is taken at the optical layer: failures are detected by the IP routers, which adapt their routing tables. So when one link breaks, data traffic which used to go over clear-channels using this link is rerouted over other clear-channels which do not use this link.

An important problem with the use of IP restoration in WDM networks is caused by the fact that each link usually carries several clear-channels. This means that a single link failure normally causes a number of clear-channels to go down at the same time. It is then possible that the logical network becomes disconnected due to these concurrent failures, and IP restoration becomes impossible. This phenomenon is called failure propagation [3]. In order to avoid this, 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 the IP network of (B) need to be routed on the WDM network of (A). A first possible routing is given in (C). Clearly this routing 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 only node 4 and the other containing all the other

nodes. Routing (D) on the other hand is survivable: no matter which link breaks, the logical network will always remain connected. In particular, a failure of link (2, 5) disconnects the clear-channels d and f . However, their endpoints stay connected in what is left of the logical graph.¹

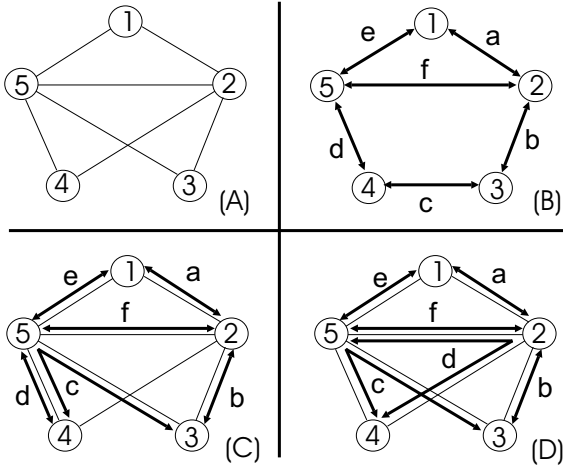


Figure 1. Mapping a logical topology onto a physical topology. (A) Physical topology. (B) Logical topology (C) An unsurvivable mapping. (D) A survivable mapping. In (C) and (D), the clear-channels of the logical graph (B) are placed on the physical topology of (A).

It is clear that the above problem description can be extended to node failures: route the clear-channels in such a way that no single node failure can disconnect the network. Also multiple simultaneous link failures are possible (e.g. because they run through the same conduit [17]) and could be taken into consideration when routing the clear-channels. In this paper we focus on survivable routing in the case of single link failures, but we will also indicate how the algorithm can be extended to take into account node failures and multiple simultaneous link failures. We show that our algorithm scales very well with respect to the number of nodes in the network. The rest of the paper is organized as follows. In section 2, a more detailed description of the problem is given, and related work is discussed. In section 3 the working of the algorithm is explained and in section 4 test results are presented.

¹2 and 5 stay connected using clear-channels a and e , whereas 4 and 5 stay connected using clear-channels c , b , a and e .

2 Problem definition and related literature

For this paper, we assume that a physical WDM topology and a logical IP topology are given. The logical topology has to be mapped onto the physical topology in such a way that it is survivable. This means that the clear-channels have to be routed on a physical path in such a way that no failure in the physical network can disconnect the logical network. In this section we will focus on survivability with respect to single physical link failures, but a mapping could also be made survivable with respect to node failures or multiple link failures. Note that in reality survivable routing would be 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 given physical topology (see [2, 11]). In the following we give a formal description of the problem, and an overview of existing related literature.

2.1 Formal problem definition

The input for the problem consists of two undirected² graphs: $G_{ph}(V, L)$ 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 source node s and destination node d . A solution to the problem 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 $R^{sd} = (r_{mn}^{sd})$ needs to be found, for which $r_{mn}^{sd} = 1$ if l_{mn} is on the path of c_{sd} . Clearly $r_{mn}^{sd} \leq l_{mn}$ must hold.

A mapping is evaluated by disconnecting the links l_{mn} of the physical graph G_{ph} one by one. For each l_{mn} , all clear-channels c_{sd} for which $r_{mn}^{sd} = 1$ are disconnected in G_{lo} , giving rise to a partial logical graph G_{lo}^{mn} . Using the clear-channels of this partial graph, an attempt is made to find a path connecting the endpoints s and d of each of the disconnected clear-channels c_{sd} . If no path is found (s and d are disconnected in G_{lo}^{mn}), we say that c_{sd} is unsurvivable on l_{mn} , and we call $\{c_{sd}, l_{mn}\}$ an unsurvivable pair. In this way an unsurvivability matrix $U = (u_{mn}^{sd})$ is obtained, in which u_{mn}^{sd} is 1 if $\{c_{sd}, l_{mn}\}$ is an unsurvivable pair and 0 other-

²Optical links are inherently unidirectional, so each undirected link is made up of two different optical fibers.

wise. The function to minimize is the total number of unsurvivable pairs, given in equation 1.

$$F(R) = \sum_{mn} \sum_{sd} u_{mn}^{sd} \quad (1)$$

The problem as it is described above does not take into account wavelength assignment. If we consider wavelengths, there are two extra constraints: the distinct wavelength constraint and the wavelength continuity constraint [13]. The distinct wavelength constraint states that each clear-channel c_{sd} routed on link l_{mn} should use a different wavelength. Since each optical fiber carries a limited number of wavelengths, this comes down to a capacity constraint for each link. To reflect this, the matrix of the links L can be extended to contain integer values (rather than binary values), indicating the capacity of the links. So, if a link between m and n exists, l_{mn} will contain the capacity of the link, and otherwise it will be 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 first present an algorithm for the survivable routing algorithm without taking into account link capacity constraints, and then we present an extension for the case with link capacity constraints. We do not consider the wavelength continuity constraint, assuming that all nodes are capable of full wavelength conversion.

2.2 Related literature

There have been a number of publications on survivable WDM routing as it is defined in subsection 2.1. In [1], a tabu search approach to find survivable routing is presented. This solution method does not take into account capacity constraints. [2] is the follow-up to the previous paper. It presents a similar algorithm, but with a more complicated evaluation function, which does take capacity constraints into account. The last paper from the same research group is [3]. It goes yet a step further, by considering other forms of failure propagation (not only connectivity problems). The authors of [7] point out that the logical topology can only get disconnected by a single link failure if that link carries a full cut set of the logical graph. They formulate an ILP using this idea, providing a method to find exact solutions. Even though they did not include the link capacity constraints in the ILP for their test runs, the algorithm still needs very long run times. In [8] the same authors provide some heuristic ILP methods, in

which only a subset of the cut sets is taken into account. This leads to good and faster results. In [15] a simple heuristic method is proposed to find survivable WDM routing. None of the papers described here take wavelength continuity, node failures or multiple simultaneous link failures into account.

Some other papers consider related problems, or special cases of the survivable WDM routing problem. The algorithm proposed in [11] attempts to look at the complete logical topology design problem: the input for the problem is a physical topology, an average traffic matrix and a routing algorithm. A tabu search algorithm generates a logical topology as an intermediate result, which is routed on the physical network using a simple heuristic. To find a survivable routing the logical topology is optimized, rather than the mapping of the logical topology onto the physical topology. [6] and other papers by the same author treat the special case where the physical topology is a ring. This specification allows to deduct certain theorems about survivable designs, which can be used to make heuristic solution methods. Finally, in [16] the authors describe a solution for the special case where the logical topology is a ring. This specification can be justified by the fact that many protection mechanisms in the higher layers use ring structures to obtain survivability.

3 FastSurv: A local search algorithm for survivable WDM routing

This section gives an overview of FastSurv, a local search algorithm for survivable routing in WDM networks. In the first subsection the basic algorithm is presented: it tries to find a survivable routing solution without taking into account link capacity constraints. In the second subsection this algorithm is extended so that it tries to find a survivable routing which does observe the link capacities. Finally, in the third subsection we show how the algorithm can be adapted to provide survivability in the presence of node failures and multiple simultaneous link failures. Part of the description below appeared in [4].

3.1 The basic algorithm

An overview of the FastSurv algorithm is given in figure 2. It starts from an initial solution obtained with a simple heuristic method and tries to improve it in a number of iterations. Solutions are evaluated as explained in subsection 2.1, and the algorithm reroutes all clear-channels which were unsurvivable on at least one of the links. While rerouting FastSurv tries to avoid

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 combinations of 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

Figure 2. The basic FastSurv algorithm

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 a random order. The routing mechanism uses shortest path routing with the cost of a path depending on the clear-channels which were already routed on the network before: for every next clear-channel which is routed on the physical graph, the cost of each link is equal to the number of other clear-channels which are already placed on the link. So if we have routed i clear-channels, indicating by $C^i \subset C$ the set of these i clear-channels, then the cost for the $(i + 1)^{th}$ clear-channel to use link l_{mn} is given in formula 2. In this way, the shortest path algorithm leads to a more or less even spread of the clear-channels. The idea is that a link which carries many clear-channels has a higher probability of leaving the logical graph disconnected after a failure. So spreading the load should improve survivability, a strategy which can also be found in the heuristic of [15].

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

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. So if $U(t) = [u_{mn}^{sd}(t)]$ represents the unsurvivability matrix after evaluation of the solution obtained in iteration t , then in iteration $t + 1$ FastSurv reroutes all clear-channels c_{sd} for which $\sum_{mn} u_{mn}^{sd}(t) > 0$. While doing this rerouting, the algorithm tries to avoid routing together clear-channels which were in the previous solutions unsurvivable when routed together over the same link. The

information necessary for this is kept in a $|C| \times |C|$ matrix $P(t) = [p_{ij}(t)]$, where the indices i and j indicate clear-channels $c_{s_i d_i}$ and $c_{s_j d_j}$. The entries $p_{ij}(t)$ of this matrix are updated according to the formulas 3-5. In formula 3 $a_{ij}(t)$ is defined as the number of times that clear-channels i and j were routed together on the same link in the solution of iteration t (in this formula, $R^i(t) = [r_{mn}^i(t)]$ is the mapping matrix for $c_{s_i d_i}$ obtained in iteration t), and in formula 4 $b_{ij}(t)$ is defined as the number of times that clear-channels i and j were both unsurvivable when they were routed together on the same link. 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. $p_{ij}(t)$ is then defined in formula 5 as the exponential average of these probability estimates.

$$a_{ij}(t) = \sum_{mn} r_{mn}^i(t) r_{mn}^j(t) \quad \forall i, j \in C \quad (3)$$

$$b_{ij}(t) = \sum_{mn} u_{mn}^i(t) u_{mn}^j(t) \quad \forall i, j \in C \quad (4)$$

$$p_{ij}(t) = \begin{cases} \alpha p_{ij}(t-1) + (1-\alpha) \frac{b_{ij}(t)}{a_{ij}(t)} & \text{if } a_{ij}(t) > 0 \\ p_{ij}(t-1) & \text{if } a_{ij}(t) = 0 \end{cases} \quad (5)$$

So in $p_{ij}(t)$ the algorithm keeps a moving estimate of the probability that i and j become unsurvivable when they are routed together. These probabilities are then used to reroute clear-channels: a shortest path algorithm is applied in which the cost of a path is the probability that the clear-channel will be unsurvivable on at least one link of the path. The probability that the clear-channel will be unsurvivable on one link is calculated as the probability that it will be unsurvivable when routed together with any of the clear-channels already routed on the link (remember that we only reroute a few of the clear-channels, and leave the rest routed as they were). Using the estimates $p_{ij}(t)$, formula 6 gives the probability that clear-channel i will be unsurvivable when routed on link l_{mn} and formula 7 gives the probability that clear-channel i will be unsurvivable over its whole *path*. In these formulas independence of probabilities is assumed, even though this is often not the case. For our heuristic solution method this rough approach is a good enough guideline towards better solutions.

$$Prob_{mn}^i(t) = 1 - \prod_{j \text{ on } l_{mn}} (1 - p_{ij}(t)) \quad (6)$$

$$Prob^i(t) = 1 - \prod_{l_{mn} \text{ on path}} (1 - Prob_{mn}^i(t)) \quad (7)$$

The algorithm described here is based on the observation made in [7] 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 is defined by a cut of the network: a cut is a partition of the set of nodes V in 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 [7, 8], 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, which makes this method impractical for large networks. In [8], the authors propose two relaxations to their ILP, in which they do not include all cut sets. This considerably speeds up the algorithm, but sometimes leads to suboptimal solutions.

In FastSurv, we approximate the notion of the cut sets by the probability estimates $p_{ij}(t)$. A look at the example logical graph given in part (B) of figure 1 could shed a bit more light on the meaning of $p_{ij}(t)$. In this graph, the pair of clear-channels $\{b, d\}$ forms a cut set of size two. b and d will both be unsurvivable every time they are routed together, and therefore $p_{bd}(t)$ will be 1. $p_{be}(t)$ on the other hand will normally be lower than 1, since b and e together do not form a cut set. $\{b, e\}$ is part of the larger cut set $\{b, e, f\}$ though, and therefore b and e will both be unsurvivable if they are routed together with the third clear-channel f . So, when two clear-channels i and j do not form a cut set of their own, the probability $p_{ij}(t)$ depends on which other clear-channels are also routed together with them.³ 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 $p_{be}(t)$, $p_{bf}(t)$ and $p_{ef}(t)$, and the cut set will be taken into account by the algorithm. If on the other hand f is hardly ever routed together with b and e , all three probabilities will be low: the cut set will be considered as unimportant and the algorithm will not take it into account. Also for very large cut sets, it will hardly ever happen that all the elements of the cut set are routed together on the same link, and therefore the individual pairwise probabilities $p_{ij}(t)$ among the elements of the cut set will be very low. So the pairwise probabilities can be

³That is also why we need the discount factor α in formula 5: the routing of the other clear-channels changes from iteration to iteration, and therefore also the probabilities.

seen as a simple way to focus only on important cut sets.

3.2 Extending FastSurv to take link capacities into account

The FastSurv algorithm as it is described in the previous subsection can find a survivable routing, but does not take into account link capacity constraints. In this subsection we present an extension for this purpose. The full algorithm is presented in figure 3. A full iteration of the algorithm consists of a number of iterations of the previously described survivable routing algorithm (which we will call survivability iterations) and then a number of iterations to decrease the number of link capacity constraint violations (which we will 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 are run 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 8. The full iterations (the combination of survivability and capacity iterations) are repeated until the routing is survivable and at the same time observes all link capacity constraints, or until a maximum number of iterations is reached.

$$F_c(R) = \sum_{mn} \max\left(\left(\sum_{sd} r_{mn}^{sd}\right) - l_{mn}, 0\right) \quad (8)$$

In each of the capacity iterations, the clear-channels to be rerouted are chosen randomly from among the clear-channels which are routed over at least one link with overcapacity. More formally, a clear-channel c_{sd} is considered for rerouting if $\sum_{mn} (r_{mn}^{sd} * \max((\sum_{s'd'} r_{mn}^{s'd'}) - l_{mn}, 0)) > 0$. The maximal number of clear-channels to be rerouted in one iteration step is empirically set to 10 percent of the total number of clear-channels in the graph. Too high a number would make a step in the local search too large, whereas too low a number would not allow the algorithm to escape from local optima.

All the chosen clear-channels are removed from the routing matrix and they are rerouted one by one in random order. The rerouting is again done with shortest path routing. 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 total number of clear-channels on the link is lower than the link's capacity, this cost is now divided by the link's capacity. In this way overfull links get a much higher cost, and hence they are

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 constraint violations
12. Return to step 9

Figure 3. The extended FastSurv algorithm

avoided. Formally, if $C^i \subset C$ is the set of the i clear-channels which are already routed (either 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 9. Also for the initial routing we now use cost formula 9 rather than 2. The capacity iterations have some similarities with a local search for routing and wavelength assignment described by Nagatsu et Al. in [10].

$$cost^{i+1}(l_{mn}) = \begin{cases} \frac{\sum_{c_{sd} \in C^i} r_{mn}^{sd}}{l_{mn}} & \text{if } \sum_{c_{sd} \in C^i} r_{mn}^{sd} < l_{mn} \\ \sum_{c_{sd} \in C^i} r_{mn}^{sd} & \text{if } \sum_{c_{sd} \in C^i} r_{mn}^{sd} \geq l_{mn} \end{cases} \quad (9)$$

3.3 Adapting the algorithm for node failures and multiple simultaneous link failures

So far we have only considered survivability in the case of single link failures. Equally important are survivability with respect to node failures and multiple simultaneous link failures. In this subsection we explain how FastSurv can be adapted for these scenarios.

A routing is survivable with respect to node failures if no single node failure leaves the logical topology disconnected. The main difference with respect to the problem description in subsection 2.1 is that a solution is not evaluated by disconnecting the links l_{mn} one by one, but by disconnecting the nodes n . Just like before, one then removes all clear-channels c_{sd} going over

n and tries to find a path between s and d in the remaining partial graph G_{lo}^n . If this proves to be impossible, $u_n^{sd} = 1$. One difference is that clear-channels c_{nd} or c_{sn} , which are incident on n , can never be routed survivably with respect to a failure of n , and u_n^{nd} and u_n^{sn} are therefore left out of consideration. Adapting the algorithm description of subsection 3.1 is straightforward: the probability that a clear-channel will be unsurvivable on a path is the probability that it will be unsurvivable on a node in its path, and the probability that it will be unsurvivable on a node is the probability that it will be unsurvivable when routed together with any of the other clear-channels on the node. Clear-channels incident on a node are not taken into account for the probability calculations.

For the case of multiple simultaneous link failures, one usually defines shared risk groups [5]. These are groups of links which are likely to fail together (e.g. because they share a conduit). The problem description of subsection 2.1 can easily be adapted to take this into account. We define a number of shared risk groups srt_k , which are sets of links. A solution is evaluated by considering the shared risk groups srt_k one by one, and disconnecting all the links $l_{mn} \in srt_k$ simultaneously. Then all clear-channels c_{sd} which run over the links of srt_k are disconnected, giving rise to a partial logical graph G_{lo}^k . If there is no path between s and d in G_{lo}^k , $u_k^{sd} = 1$. Adapting the algorithm of subsection 3.1 is again very straightforward: the probability that a clear-channel will be unsurvivable on a link l_{mn} in its path is now the probability that it will be unsurvivable with any of the clear-channels routed over any of the links in the shared risk group(s) to which l_{mn} belongs.

4 Test results

In this section we describe test results we obtained with FastSurv. We compare it to current state of the art algorithms on different physical and logical network topologies. Using test problems of increasing sizes, we show that FastSurv is much more scalable than existing algorithms. Subsection 4.1 contains results for the basic survivable routing algorithm, and subsection 4.2 for the algorithm with link capacities. Our programs are implemented in C++ and all tests were run on a PC with a 2.4 GHz Intel Pentium 4 processor.

4.1 Test results for the basic algorithm

For the case without link capacities, we made comparisons with the full and the relaxed ILP methods presented in [8], and with the tabu search algorithms

Table 1. Test results on NSFNET.

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

presented in [1] and [2]. For every test problem, FastSurv is given 100 iterations, with a random restart after every 10^{th} iteration in order to avoid that it gets 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 the comparison with the ILP methods, we ran FastSurv on the same test problems which were used in [8]. The physical network used for these tests is the 14-node 21-link NSFNET. The logical networks were randomly generated by the authors of [8]. There are logical networks of degree 3, 4, and 5, where a network of degree n is defined as one in which all nodes have degree n . For each degree there are 100 logical networks, and each network is at least two-connected. The algorithms are run once for each test problem. The results are summarized per network degree in table 1, where *ILP* refers to the full ILP solution method, and *relax-1* and *relax-2* to the two relaxations of the ILP, which use only a subset of the survivability constraints. *Unsurvivable* is the number of problems for which no survivable solution was found, and *Time* is the average run time per problem in CPU seconds. The results show that FastSurv gives good results in short times. The other approaches are much slower (especially ILP), and relax-1 is not able to solve all problems.

For the comparison with the tabu search algorithms, we could not obtain the original test problems used by the authors, nor the source code, so the results presented here are obtained with our own implementation of the algorithms presented in [1] and [2]. Strictly speaking, only the tabu search of [1] (which we refer to as TS97) addresses exactly the same problem as FastSurv, since the tabu search of [2] (TS98) takes link capacities into account. Nevertheless, there are differences in TS98 which make it more efficient. Therefore we made an adaptation of TS98 which does not take link capacities into account (TS*03).

As a first series of tests we ran the algorithms on the same physical network which was used in [1] and [2]:

Table 2. Test results on ARPA2.

		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

the 21-nodes, 26-links ARPA2 network. As logical networks we used 4 randomly generated graphs: 2 of average node degree 3 and 2 of average node degree 4. We ran each algorithm 10 times for each logical network. The results are given in table 2, where *Time* is the average run time per problem in CPU seconds, and *Uns* is the number of runs in which no solution was found. All three algorithms always find an optimal solution, except TS*03 in two cases. It is quite clear however that FastSurv has much shorter run times.

We then ran a much larger series of tests, with networks of increasing sizes, in order to compare the scalability of the algorithms. Physical networks were randomly generated with number of nodes ranging from 20 to 50 with a step size of 5, and with average node degree 3. Logical graphs of the same number of nodes were generated with average degrees 3 and 4. For each size and each degree there are 10 different physical networks and 10 different logical networks. We ran each algorithm once for each combination of physical network and logical network, giving 100 results per network size and node degree. The results are summarized in figures 4, 5, 6 and 7. The bars in figures 4 and 5 indicate for each network size how many problems (out of 100) each algorithm managed to route survivably. TS97 and TS*03 show comparable performance over this wide range of problems, while FastSurv consistently performs better. The differences on degree 4 problems are smaller than on degree 3 problems because these problems are easier: they have more clear-channels, and therefore better connectivity, which makes survivable routing easier. The graphs of figures 6 and 7 indicate the run times in CPU seconds needed by the different algorithms. They confirm that FastSurv is much more scalable with respect to the network size. For degree 3, we ran FastSurv on more problems, up to 150 nodes. The run times of the other algorithms were prohibitively large for these problem sizes. FastSurv shows continued short run times for these larger problems. For degree 4, we can see that TS*03 is better scalable than for degree 3. This is again due to the fact that these problems are easier. TS97 does not profit from this to the same extent, because of the way

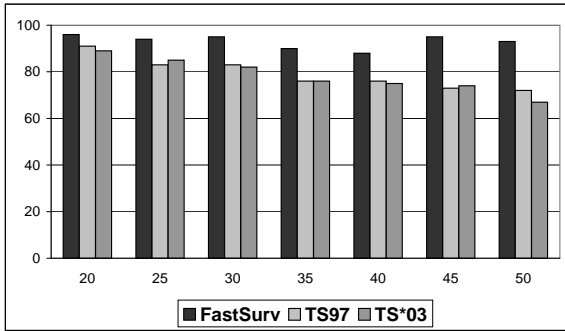


Figure 4. Problems solved to optimality for degree 3 networks of different sizes.

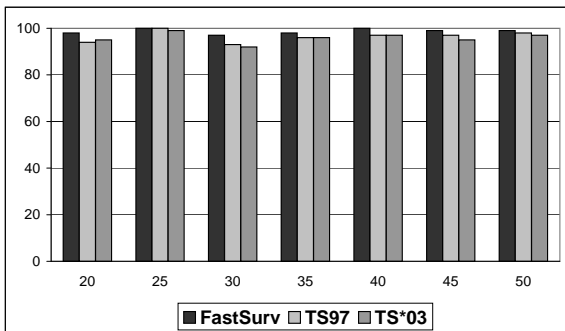


Figure 5. Problems solved to optimality for degree 4 networks of different sizes.

it goes about the problem (see [1]).

An explanation for the faster run times of FastSurv 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 maximally $O(N^4)$ (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^4 , the complexity of an iteration is maximally $O(\alpha N(N^2 + N^4)) = O(N^5)$. In FastSurv 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 maximal complexity of $O(\alpha N \cdot N^2 + N^4) = O(N^4)$. The fact that the complexities are polynomial is confirmed in figure 8, where we compare the average run time per iteration for Fast-

⁴ If the average node degree is fixed, the number of clear-channels $|C|$ is a linear function of N .

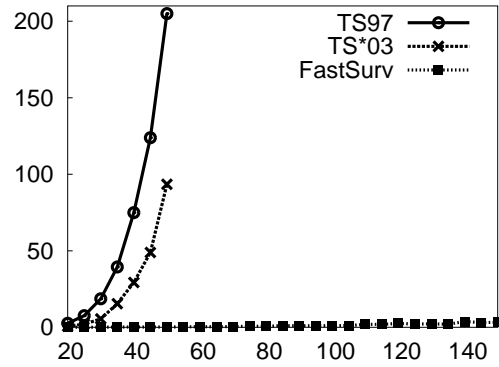


Figure 6. Run times in CPU seconds on degree 3 problems summarized by network size.

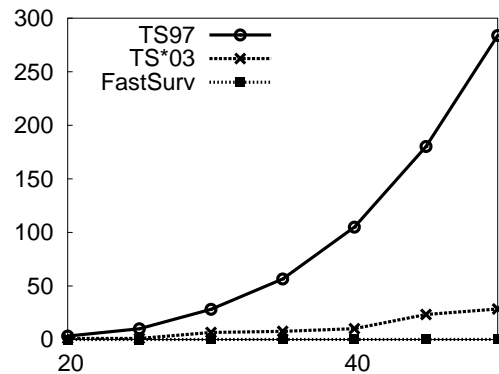


Figure 7. Run times in CPU seconds on degree 4 problems summarized by network size.

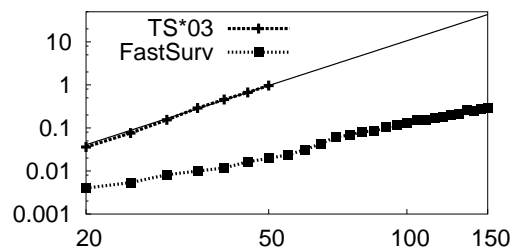


Figure 8. Average run time per iteration in CPU seconds on degree 3 problems.

Surv and TS*03: the run times per iteration plotted against the number of nodes in log-log scale follow more or less a straight line for both algorithms (for TS*03 values above 50 nodes were extrapolated). The gradi-

ents are lower than the theoretically predicted maxima though: 2.15 for FastSurv and 3.45 for TS*03.

4.2 Test results for the extended algorithm

For the extended algorithm we only compare to TS98 [2], since it is the only one which takes link capacities into account. We use the same physical topologies as in 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 they were set differently for different logical graphs. For the ARPA2 network, link capacities were set to 7 for degree 3 logical graphs and to 8 for degree 4 graphs. For the randomly generated networks, capacities were set to 6 for degree 3 networks up to 40 nodes, to 7 up to 55 nodes, and to 8 up to 150 nodes. For degree 4, capacities were set to 7 up to 30 nodes and 8 up to 50 nodes. The logical graphs are the same as in the previous section.

The results for the arpa2 network are presented in table 3, where *Time* is the average run time per problem in CPU seconds, *Au* the average number of unsurvivable pairs (according to formula 1) and *Ac* the average overcapacity (according to formula 8). *Best* is the best solution over 10 runs, with on the left its number of unsurvivable pairs, and on the 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. The results on the random graphs are shown in figures 9-12, where figures 9 and 10 present the number of test problems solved to optimality (meaning survivable solutions without any overcapacity), and figures 11 and 12 present the average run times. These results confirm FastSurv’s advantage both in solution quality and run time, and show that also the extended FastSurv algorithm scales much better than TS98 with respect to the number of nodes. For the degree 3 problems we again ran FastSurv for networks up to 150 nodes to show the continued good performances.

Table 3. Test results on ARPA2 with link capacities.

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

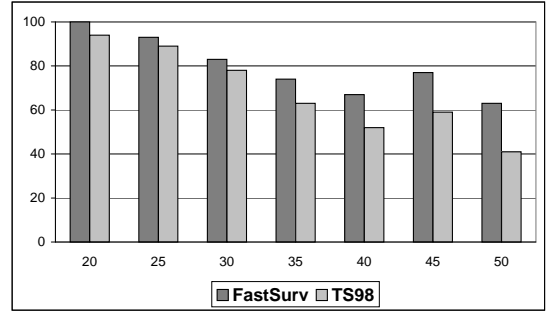


Figure 9. Problems solved to optimality for degree 3 networks of different sizes.

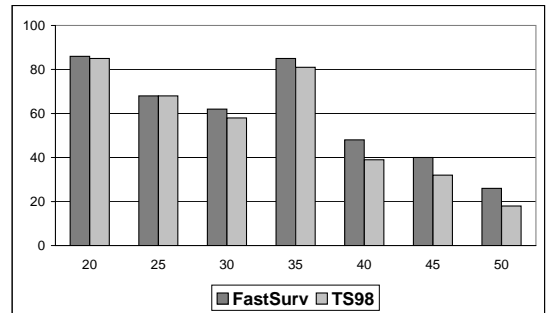


Figure 10. Problems solved to optimality for degree 4 networks of different sizes.

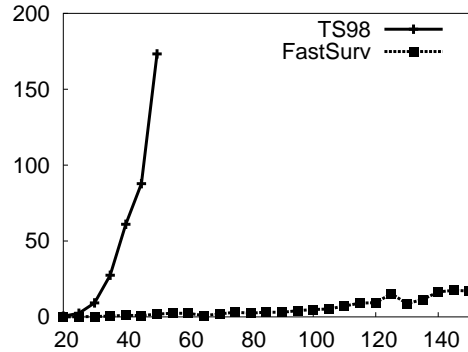


Figure 11. Run times in CPU seconds on degree 3 problems summarized by network size.

5 Conclusions

In this paper we have described FastSurv, a local search algorithm for survivable routing in WDM networks. In an extensive series of test runs, we have

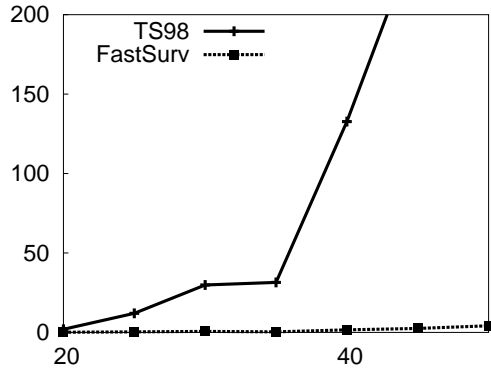


Figure 12. Run times in CPU seconds on de-gre 4 problems summarized by network size.

compared FastSurv to other algorithms for survivable WDM routing. It gave better results than current state of the art algorithms, while using much shorter run times. Moreover, these advantages in terms of solution quality and run times became larger for increasing network sizes. The property of giving high quality results in very short time can be important. As was pointed out in section 2, routing a logical topology survivably on a physical network is normally part of a larger logical topology design algorithm, and a time-consuming algorithm there could considerably slow down the larger algorithm. For example, in [11] the authors need to do survivable routing as part of their algorithm, but have to use a greedy heuristic since existing state of the art algorithms take too long.

6 Acknowledgements

We thank Gianni Di Caro and Roberto Montemanni for useful advice and Patrick Thiran for pointing out this problem area. We thank Maciej Kurant for useful discussions, and Aradhana Narula-Tam for sharing the test problems used in her work. This work is 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] F. Ducatelle and L.M. Gambardella. FastSurv: A new efficient local search algorithm for survivable routing in WDM networks. In *Proceedings of GlobeCom*, 2004.
- [5] I.P. Kaminow and T.L. Koch. *Optical Fiber Telecommunications IIIA*. Academic Press, 1997.
- [6] 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.
- [7] E. Modiano and A. Narula-Tam. Survivable routing of logical topologies in WDM networks. In *Proceedings of INFOCOM '01*, 2001.
- [8] E. Modiano and A. Narula-Tam. Survivable lightpath routing: a new approach to the design of WDM-based networks. *IEEE JSAC*, 2002.
- [9] B. Mukherjee. *Optical Communication Networks*. McGraw-Hill, 1997.
- [10] 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.
- [11] 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 '01*, 2001.
- [12] R. Ramaswami and K.N. Sivarajan. *Optical Networks: a Practical Perspective*. Morgan Kaufmann Publishers Inc., 1998.
- [13] G. Rouskas. *Wiley Encyclopedia of Telecommunications*, chapter Routing and Wavelength Assignment in Optical WDM Networks. John Wiley and Sons, 2001.
- [14] L. Sahasrabudde, S. Ramamurthy, and B. Mukherjee. Fault management in IP-over-WDM networks: WDM protection versus IP restoration. *IEEE JSAC*, 2002.
- [15] G.H. Sasaki, C.-F. Su, and D. Blight. Simple layout algorithms to maintain network connectivity under faults. In *Proc. of the Annual Allerton Conf.*, 2000.
- [16] 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 International Communication Conference ICC02*, 2002.
- [17] J. Strand, A.L. Chiu, and R. Tkach. Issues for routing in the optical layer. *IEEE Comm. Magazine*, 2001.
- [18] 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.