Survivable Mapping Algorithm by Ring Trimming (SMART) for large IP-over-WDM networks. [*]

Maciej Kurant, Patrick Thiran
LCA - School of Communications and Computer Science
Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland
Email:{maciej.kurant, patrick.thiran}@epfl.ch

Abstract

We develop a fast and efficient algorithm that finds a survivable (i.e., robust to single fiber failures) mapping of IP topology on the mesh of fibers in IP-over-WDM networks; we call it SMART. A number of algorithms solving this problem can be found in the literature. Since ILP solutions are highly complex, many heuristics were proposed. They usually start with some initial mapping and then try to gradually improve it. This involves the evaluation of the entire *topology at each iteration, which is costly for large topologies. We propose a different approach. The SMART algorithm breaks down the task into a set of independent and very simple subtasks. The combination of solutions of these subtasks is a survivable mapping. This is why SMART is orders of magnitude faster than other proposals, especially when dealing with large topologies. We also extend the SMART algorithm to obtain a mapping resilient to fiber span failures, node failures and double–link failures. Finally, we show that the scalability of the standard heuristic approaches is additionally limited (contrary to SMART) when applied to double–link failures.*

1. Introduction

An important issue in an IP-over-WDM network is its robustness to failures, i.e., survivability. Generally there are two approaches to providing survivability. *Protection* uses the pre–computed backup paths; in the case of a failure, a backup path is applied. *Restoration* finds a new path dynamically, once the failure has occurred. Therefore protection is less resource efficient (the resources are reserved without knowledge of the failure) but fast, whereas restoration is more resource efficient and slower. Protection and restoration mechanisms can be provided at different layers. *IP layer* mechanisms can handle the failures that occur at both layers (IP and WDM), contrary to *WDM layer* mechanisms that are transparent to the IP topology. It is not obvious which combination (mechanism/layer) is the best; each has pros and cons [1]. In this paper we consider exclusively the *IP restoration approach*, because, it was shown in real networks to be an effective and cost–efficient solution (see e.g., Sprint network [2]).

Each IP link is mapped on the physical topology as a *lightpath*. Usually a fiber is used by more than one lightpath (in Sprint the maximum number is 25 [3]). Therefore a single physical failure will usually bring down a number of IP links. With the IP restoration mechanism, those IP link failures are detected by IP routers, and alternative routes in the IP topology are found. In order to enable this, the IP topology should remain connected after a failure; this in turn, may be guaranteed by an appropriate mapping of IP links on the physical topology. We call such a mapping a *survivable mapping*.

Clearly, no mapping can protect against *any* type and number of physical failures. Therefore in order to call a mapping 'survivable,' we need to specify the failures it has to survive. There are several types of failures that may be encountered in the WDM layer. The most common is a *single physical link failure*. This might be caused by a fiber (physical link) cut, a fault of a single interface card in the optical switch, or a fault of an optical amplifier. If we allow for the physical location of the fibers we will extend single link failures to single *span failures* [4]. A *span* is a collection of all fibers co-located in the same conduit, between two consecutive points of access (such as a manhole or an amplifier site). To limit costs, different physical links are sometimes put in the same span (e.g., along railway and electricity lines); then a single cut can break all of them at once. So a span failure might be either a failure of a single physical link or simultaneous failure of all physical links that are put in the same span. We can also encounter *node failures*; they are the consequence of a failure of equipment at nodes, such as switches. Finally, we will consider the *double–link failures*, i.e., independent failures of any two physical links.

* The work presented in this paper was financially supported by grant DICS 1830 of the Hasler Foundation, Bern, Switzerland.

Usually such a situation takes place when the second failure occurs before the first one is repaired. This is not very common, but possible. For example, in the Sprint network, the time between two successive optical failures ranges from 5.5 sec to 7.5 days with a mean of 12 hours [3]. Most of them are repaired automatically within several minutes, but those requiring human intervention (e.g., after a fiber cut) may last hours or days. It is quite probable that during that period another physical failure occurs.
The approach we introduce in this paper is suitable to deal with every type of failure listed above.

1.1. Related work

The problem of survivable mapping is not new; it was first defined in [5]. Since then, many algorithms solving this NP-complete problem (with different variations) were proposed. In general, they can be divided into two groups: greedy search based on Integer Linear Programming (ILP), and heuristics. The ILP solutions can be found for example in [6, 7]. However, this approach leads to unacceptably high complexity for networks of a non-trivially small size [8] (larger than few tens of nodes). The second approach uses various heuristics, like Tabu Search [5, 9, 10, 7], Simulated Annealing [11] and others [1, 12]. Most of them start with some initial mapping (e.g., shortest path) and try to improve it at subsequent iterations. Therefore the-time complexity of each iteration is dominated by the time it takes to evaluate the candidate mapping of the *entire* logical topology, which is costly for large topologies. We will show that, in practice, the application of those heuristics is limited to the topologies of a few hundred nodes, or even fewer for more sophisticated failure scenarios. The algorithm we propose in this paper opens a third group. It is based on a breakdown of the problem into a set of independent smaller problems, which are easy to solve. Each of them is solved separately, and then the solutions are combined to obtain a survivable mapping of the entire topology. That makes SMART fast and scalable. One equally fast solution was proposed in [12]. However, we show later that it does not work well for larger graphs.

Failure scenarios more complex then a single link failure were also addressed before, e.g., in [4] (span failures), in [13] (node failures) and in [14, 15, 16] (double-link failures). But they were approached only with WDM layer protection and restoration mechanisms. To the best of our knowledge, this is the first time IP restoration is applied to these failure scenarios.

Most of the approaches mentioned above take as a parameter the number of wavelentgths in each fiber, i.e., take fiber capacities into account. Clearly, this better reflects the real-life scenarios. The version of SMART presented in this paper, like the approaches in [5, 6, 12], releases the capacity constraints. We believe that the adaptation of SMART to capacity constraints is possible and easy; this is the main point in the scope of our future work.

1.2. Contributions

The contributions of this paper are the following:

- a novel approach to construction of a survivable mapping of the logical (IP) topology on the physical (WDM) topology, resulting in an algorithm 2-3 orders of magnitude faster than previous proposals;
- a low complexity allowing to run it on large topologies (hundreds and thousands of nodes) in reasonable time (in contrast to previous proposals);
- efficiency under various failure scenarios, namely span failures, node failures and double-link failures.

1.3. Organization of this paper

In Section 2 we introduce notations and formalize the problem. In Section 3 we describe the SMART algorithm for a single-fiber failure case and we give an example run. In Section 4 we present the adaptations of SMART to span/node/double–link failures. In Section 5 we compare by simulation the SMART algorithm with other algorithms and demonstrate SMART performance for large topologies. In Section 6 we conclude the paper.

2. Notation and problem formulation

Physical and logical topologies are represented by undirected graphs $G^{\phi} = (V, E^{\phi})$ and $G^{L} = (V, E^{L})$, respectively. V is the set of vertices, E^{ϕ} and E^{L} are the sets of undirected edges. Note that for simplicity we assume $V^{\phi} \equiv V^{L} \equiv V$. Moreover, G^{ϕ} and G^{L} are 2-edge-connected. (A graph G is *k-edge-connected* if G is connected and every set of edges disconnecting G has at least k edges [17].) This implies that, on both levels separately, one link failure will not disconnect the graph, which is clearly the necessary condition for the existence of a survivable mapping.
Since we have assumed unlimited capacities of physical links, we do not consider the wavelengths assigned to lightpaths and wavelength converters placement. Hence a *lightpath* will be no more than a loop–free path in the physical topology.

Definition 1 (Mapping) *Let P^{ϕ} be a set of all possible lightpaths (physical paths) in the physical topology, and let $A \subset E^{L}$ be a set of logical links. A* mapping *M_A is a function $M_A : A \rightarrow P^{\phi}$ associating each logical link from the set A with a corresponding lightpath in the physical topology.*

For arguments from beyond A, M_A is not defined. In the particular case $A = E^{L}$, M_{E^L} is defined for all links in the logical topology. Two examples of mappings are given in Fig. 1.

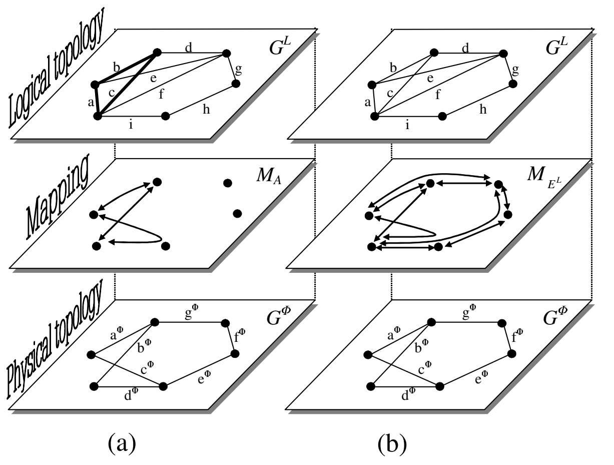

Figure 1. A mapping M_A of the logical topology $G^L = (V, E^L)$ on the physical topology $G^\phi = (V, E^\phi)$. The domain A of the mapping is $A = \{a, b, c\} \subset E^L$ (set in bold) in (a) and $A = E^L$ (set of all logical edges) in (b). In both figures the lightpaths associated with logical links a, b and c are $M_A(a) = \{c^\phi, d^\phi\}$ and $M_A(b) = \{a^\phi\}$ and $M_A(c) = \{b^\phi\}$, respectively.

Definition 2 (Disjoint mapping) *A mapping $M_A, A \subset E^L$, is* disjoint *if within the image $M_A(A)$ of its entire domain A, each physical link is used at most once.*

The mapping M_A in Fig. 1a is disjoint.

Definition 3 (k-survivability) *A mapping M_{E^L} of the logical topology G^L (i.e., of all edges from G^L) on the physical topology G^ϕ is k–*survivable *if a simultaneous failure of k physical links in G^ϕ does not disconnect G^L.*

It is easy to check that in Fig. 1b the mapping M_{E^L} is 1-survivable. To lose this property it is enough to change the lightpath corresponding to the logical link a from $M_{E^L}(a) = \{c^\phi, d^\phi\}$ to $M_{E^L}(a) = \{a^\phi, b^\phi\}$; then the failure of a^ϕ disconnects the logical topology.

Given the the logical topology $G^L = (V, E^L)$ and the physical topology $G^\phi = (V, E^\phi)$, *our goal* is to find a 1-survivable mapping M_{E^L} of G^L on G^ϕ. (Later we extend it to more sophisticated failure scenarios.)

One of the main operations in the SMART algorithm is *contraction*, defined as follows:

Definition 4 (Contraction [17]) Contracting *an edge $e \in E$ in a graph $G = (V, E)$ is deleting that edge and merging its end–nodes into one. The result is called a* contracted graph G^C*, and denoted by $G^C = G \downarrow e$. We will also allow contracting the set of edges $A \subset E$ resulting in a contracted graph $G^C = G \downarrow A$ (see Fig. 2). Note that the order of the edges in A does not affect the result.*

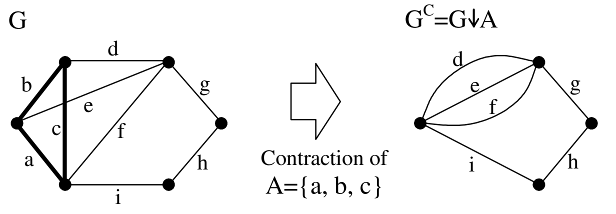

Figure 2. Contraction example. Contraction might result in creation of multi-edges (and also self–loops).

3. The SMART algorithm

The idea of the SMART algorithm is the following. First choose from the logical topology G^L a cycle (a 'ring') $C \subset E^L$ and map it *disjointly*. The disjoint mapping of a logical cycle ensures that this cycle will remain connected after any single fiber failure. In other words the cycle C is already mapped in a 1-survivable way. Now, we have to take care about the rest of the logical graph $G^L \backslash C$ and a survivable connection of $G^L \backslash C$ with C. Therefore we contract the cycle C in the logical topology G^L (we 'trim' it) and repeat the above procedure for the resulting graph $G^L \downarrow C$. We iterate this until the contracted logical topology converges to a single node, which guarantees survivability. The example run of SMART is ilustrated in Fig. 3.

3.1. Algorithm

The pseudo-code of the SMART algorithm is:

Initialization Contracted logical topology $G^C := G^L$. Mapping M_A and its domain A are empty: M_A=∅, A=∅

Step 1 Pick a cycle C in G^C. Prefer shorter cycles, but only those not considered since the last successful iteration. IF no cycle found, THEN RETURN unsurvivable mapping M_A. END.

Step 2 Map disjointly the cycle C on the physical topology using the *DisjointMap* function (see Sec. 3.2).
IF the disjoint mapping is not found, THEN the iteration is *unsuccessful*; GOTO Step 1
IF the disjoint mapping is found, THEN the iteration is *successful*. Denote this mapping by M_C.

Step 3 Extend the mapping M_A by M_C, i.e., merge those mappings and call the result a new M_A. This operation is well defined since the sets of logical links A and C are disjoint.

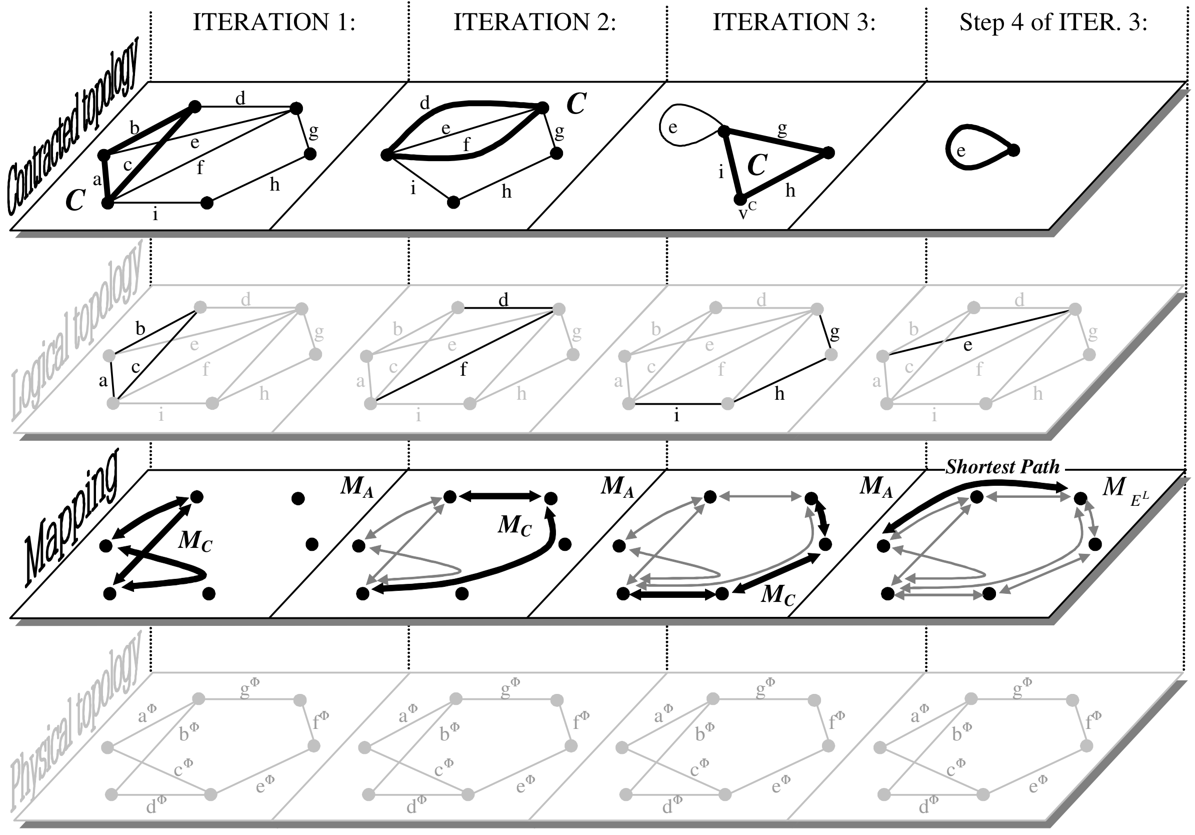

Figure 3. Illustration of the SMART algorithm. We have four layers, from bottom to top: physical topology G^{ϕ}, mapping M_A, logical topology G^L and contracted logical topology G^C. During a run of the SMART algorithm, only the contracted topology G^C and the mapping M_A change from one iteration to the next one. The logical and physical topologies are included only for the context; therefore they are set in grey. At each iteration a cycle C picked from the contracted topology layer is set in bold. For example at Iteration 2 we have $C = \{d, f\}$. Then a disjoint mapping M_C is found for this cycle; it is set in bold in the mapping. (At each iteration, the grey lightpaths in the mapping are those found at all preceding iterations.) Next the cycle C is contracted, resulting in a new contracted logical topology G^C used at the subsequent iteration. Once G^C has converged to a single node, the underlying mapping is 1-survivable. However, there might be some remaining (unmapped) logical links, forming self-loops in G^C. In our example e is such a link; we map it in any way (here it is the shortest path). Now we combine the mappings found in every iteration to obtain the mapping M_{E^L} (last column) of the entire logical topology. The mapping M_{E^L} is 1-survivable.

Step 4 Contract C in G^C (i.e., $G^C := G^C \downarrow C$).
IF G^C consists of *one* node, THEN map the remaining logical links (self-loops in G^C) in any way and RETURN survivable mapping M_{E^L}. END.

Step 5 GOTO Step 1.

Remark:
Note, that even if each iteration of this algorithm is unsuccessful, it will not stop before checking *all* cycles existing in graph G^C. Since the number of all cycles grows rapidly with the size of the graph, it seems reasonable to limit the number of consecutive unsuccessful iterations. For our implementation we allowed for at most 10 such iterations.

3.2. DisjointMap function

In Step 2 of the SMART algorithm, we applied a function searching for a disjoint mapping of a set C^L of logical links; we called this function $DisjointMap$. The problem is equivalent to the edge-disjoint paths problem [18], that is proved to be NP-complete. Therefore we applied a simple $DisjointMap$ heuristic, as follows. Let each physical edge have a weight (these weights will be used exclusively within $DisjointMap$) and let this weight be initially set to one. At each iteration, the $DisjointMap$ heuristic maps the logical links from C with shortest path. If no physical link is used more than once, the disjoint solution was found. Otherwise, the weight of each physical link used more than once is increased, and a new iteration starts. After several unsuccessful iterations the DisjointMap function fails.

3.3. Complexity of SMART is polynomial

The complexity of *one iteration* of the SMART algorithm is dominated by the $DisjointMap$ function, which in turn uses $O(1)$ times the Dijkstra shortest path algorithm. Therefore the complexity of one iteration of the SMART algorithm is equal to the complexity of Dijkstra algorithm, i.e., $O(Dijkstra)$.
Now we try to estimate the *number of iterations* needed for SMART to converge to a survivable solution. We need to map $O(N)$ edges. A successful iteration maps one short cycle, i.e., of the length of order $O(1)$. So we need $O(N)$ *successful* iterations. Between any two successful iterations we can have a number of unsuccessful iterations. As explained in Remark in Sec. 3.1, in real implementations we kept this number constant, i.e., of order $O(1)$. Consequently the total number of iterations is of order $O(N)$.
The complexity of SMART is equal to the complexity of one iteration multiplied by the number of iterations = $O(Dijkstra) \cdot O(N)$. Since in the worst case $O(Dijkstra) = O(N^2)$, the worst case complexity of SMART algorithm is $O(N^3)$. However, the average case is simpler. In simulations on large topologies (see Section 5) we observed the complexity equal to $O(N^{2.4})$.

4. Adaptation to various failure scenarios

In this section we give straightforward extensions of the standard version of SMART, which deal with failure scenarios more sophisticated than a single physical link failure. The main idea is always the same - at each iteration we find a survivable mapping of some subgraph of the contracted logical topology and then contract this subgraph into one node.

4.1. Span failures

A mapping is declared *span-survivable* if it preserves the connectivity of the logical graph after any single fiber failure or after a cut of one of the multi-link spans. The SMART approach requires only a minor upgrade of the DisjointMap function to be adapted to span-survivability. Assume for instance that in the physical topology in Fig. 3 the fibers d^ϕ and e^ϕ are laid partially in the same span. Then, in order to map a cycle in a survivable way, the DisjointMap function may use those fibers *only in one lightpath*. For example, at Iteration 3, the presented mapping of the cycle $C = \{g, h, i\}$ could not be accepted, because the cut of the span $d^\phi - e^\phi$ would separate the vertex v^C. But if we change the lightpath assigned to the logical edge i from $M_C(i) = \{d^\phi\}$ to $M_C(i) = \{c^\phi, a^\phi, b^\phi\}$ then the cycle C remains connected after a simultaneous cut of d^ϕ and e^ϕ. A modification of SMART taking into account span failures will be referred to as *SMART-Span*.

4.2. Node failures

Clearly, after a failure of a vertex $v \in V$, the logical topology is disconnected - at least the vertex v is always separated. The best we can do is to keep connected the remaining part of the logical topology. Therefore a mapping is declared *node-survivable* if, after any single failure of vertex $v \in V$, the logical topology $G^L \backslash \{v\}$ remains connected. Node-survivability is a significantly more difficult problem than 1-survivability. However, the SMART approach efficiently solves this problem as well. We need only the DisjointMap function to search for *node–disjoint* mappings instead of link–disjoint ones. Then no single node failure can disconnect the mapped ring. We will refer to this version of our algorithm as *SMART-Node*.

4.3. Double-link failures

To obtain a 2-survivable mapping we use the same basic idea as before - at each iteration we find a 2-survivable mapping of some subgraph of the contracted logical topology G^C and then contract this subgraph into one node. However, the subgraph we find will differ. Before, in the case of single-link failures, the simplest subgraph we could search for was a cycle; the disjoint mapping of this cycle ensured 1–survivability. In the case of double-link failures, the subgraphs are more complex as they must be at least 3–edge–connected. Fig. 4 presents the three structures we used. Note that only the first one is a simple graph, i.e., it has no multi-edges. In fact it is the smallest possible 3–edge–connected simple graph. Since the logical topology is also a simple graph, only the Structure 1 may be found at the first iteration of the algorithm. However, at subsequent iterations we work on the *contracted* logical topology, that may have multi-edges and self-loops. Then Structures 2 and 3 will become useful.
The extension of our algorithm searching for a 2-survivable mapping is called *SMART-DF* (*DF* - Double-link Failures). The pseudo code of SMART-DF is similar to that of SMART given in Section 3.1 with one exception: in Step 1 we search for structures depicted in Fig. 4 instead of cycles. The rest of the code remains the same. In particular, SMART-DF searches for a *disjoint* mapping of the structures found in Step 1. The same DisjointMap function is applied, as defined in Section 3.2. Note that this is a slightly too restrictive approach. For instance the mapping of Structure 1 does not necessarily have to be completely disjoint - the two edges set in bold in Fig. 4.1 might use the same

physical links, still preserving the 2-survivability of the entire Structure 1 (assuming that the remaining lightpaths are link–disjoint). Similar edge pairs can be found in Structure 3. As results obtained by the 'completely' disjoint approach turned out to be satisfactory, we do not take those peculiarities into account. For the same reason we do not extend the set of structures we were looking for in Step 1, taking only these depicted in Fig. 4, which were the most frequent 3-edge-connected topologies encountered at various iterations of the algorithm.

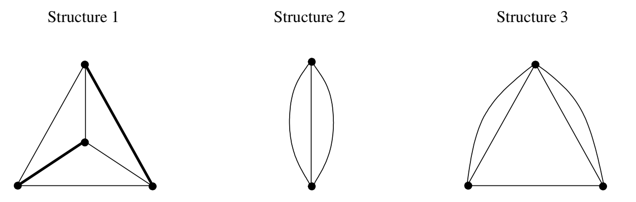

Figure 4. Subgraphs of the contracted logical topology attempted to be mapped to obtain 2-survivability.

5. Simulation results

We run the SMART algorithm on various physical and logical topologies. The *physical topologies* are presented in Fig. 5. Some modifications of standard graphs were needed to investigate the problem of span failures (b) and double–link failures (d). To simulate the large-scale networks, we generated random square lattices (e,f). The *logical topologies* were random graphs and constant degree random graphs; both of various average node degree. We also used the stack of 300 topologies identical with those used in [6].

This section is divided into several subsections according to the problems we address. First we consider the survivability against single fiber failures (1–survivability). We compare our results with the results of other approaches found in the literature. In particular, we demonstrate the efficiency and scalability of SMART, when mapping large topologies. Next, we investigate the performance of adaptations of SMART to more complex failure scenarios, i.e., span failures, node failures and double–link failures. To the best of our knowledge, this is the first time these problems are addressed by the IP restoration approach. Therefore we have no benchmark against which we can compare our algorithm; the only comparison is made with the standard version of SMART.

5.1. 1-survivability

Many solutions of the 1-survivable mapping problem can be found in the literature [6, 7, 5, 10, 12]. In contrast to our basic approach, many of them take some additional constraints into account, such as capacity, delays, number and localization of wavelength converters. For the comparison purposes we have chosen three proposals, that focus exclusively on 1-survivability issue. The first one is based on Integer Linear Programming (ILP). The other two, Tabu Search and Simple Layout Algorithm, are heuristics.

5.1.1. ILP approach

In [6] the authors specify the necessary and sufficient conditions for a mapping to be 1-survivable. These conditions are injected into the Integer Linear Programming (ILP) formulation, that is used to find a 1-survivable mapping. Then a simple relaxation (ILP-Relax) for the ILP is introduced, which substantially reduces the processing time.

We ran the SMART algorithm for exactly the same topologies as in [6], namely NSFNET as the physical topology and the same 300 random graphs of degree $\bar{d} = 3, 4$ and 5 as those in [6] for the logical topologies. A 1-survivable mapping was found in *all runs* when using ILP, ILP-Relax and SMART approaches. Therefore it is interesting to compare the *run–times* of the algorithms. The machines were not the same, yet comparable (Sun Sparc Ultra-10 vs. Pentium 500). However, we have to stress that SMART was implemented in pure C++ whereas ILP required a dedicated program (CPLEX) that could significantly affect the results. The run-times from [6] are reprinted in Table 1; the last column shows the results of the SMART algorithm. The SMART algorithm is several orders of magnitude faster then pure ILP, and about 3 orders of magnitude faster than the relaxed version of ILP. Note that the degree of the logical topology practically does not affect the run-time of SMART. This is because most of the processing time is consumed in the search of a 1-survivable mapping (convergence of the contracted logical topology to a single node); the remaining logical links (e.g., the self–loop e in Fig. 3) are mapped by the shortest path Dijkstra algorithm, which takes negligible time.

Average degree $\bar{d}$	ILP	ILP-Relax	SMART
3	8.3 sec	1.3 sec	0.0028 sec
4	2 min 53 sec	1.5 sec	0.0028 sec
5	19 min 17 sec	2.0 sec	0.0029 sec

Table 1. Run-times of ILP and SMART

5.1.2. Tabu Search and large topologies

One of the most efficient and widely used techniques to solve a 1-survivable mapping problem is *Tabu Search*. Our implementation of Tabu Search follows the one in [5]; we will refer to it as *Tabu97*. Since Tabu Search turned out to be substantially faster than the ILP approach (described in previous section), we carried out the simulations for relatively large graphs and studied Tabu Search and SMART scalability.

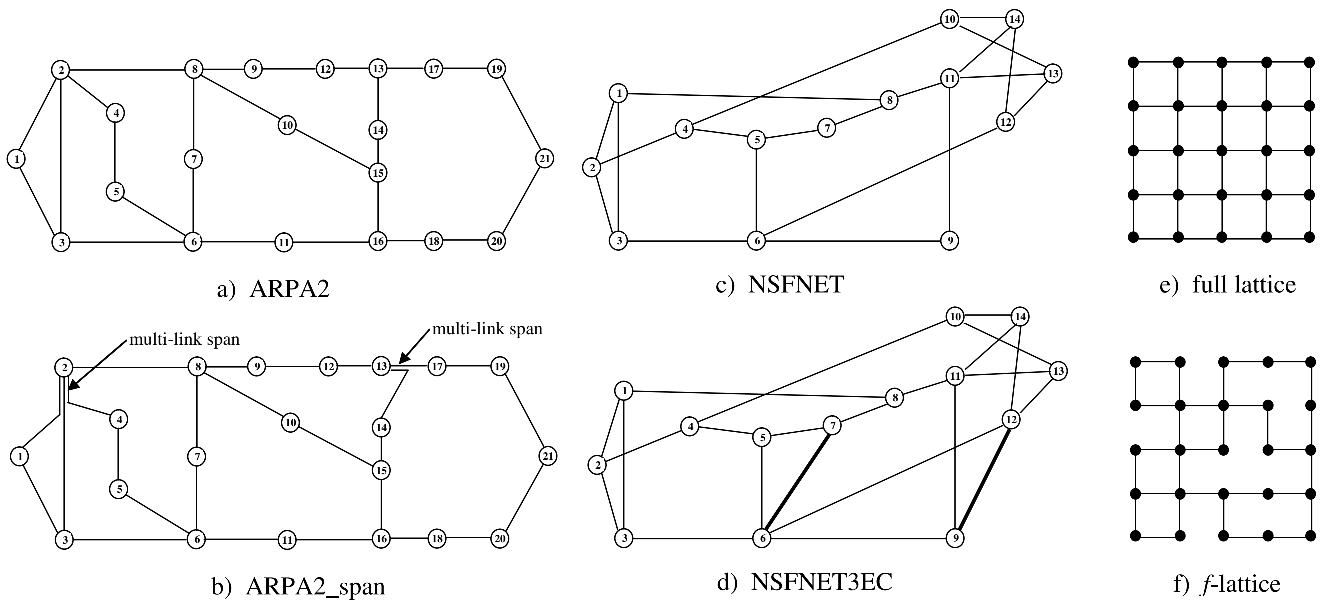

Figure 5. Physical topologies used for simulations: a) ARPA2; b) ARPA2 with two multi-link spans (spans with multiple physical links); c) NSFNET; d) NSFNET with two additional links to obtain 3–edge–connectedness (3EC); e) full square lattice; f) *f*–lattice constructed from full square lattice by deleting fraction f of links, while preserving 2–edge–connectivity.

The *physical topology* is an *f-lattice* (Fig. 5f) with the fraction of deleted edges f ranging from 0 to 0.35. The maximal value 0.35 was chosen in such a way that even the smallest topologies could be 2–edge–connected.[1] The *logical topology* was a 2–edge–connected random graph of the average vertex degree $\bar{d} = 4$. In Fig. 6 we present the results obtained in simulations on a Pentium 4 machine. We investigate the topologies with a number of vertices ranging from 16 to 900. Figs. 6a,b are related to the *efficiency* of algorithms, i.e., their ability to find a 1-survivable mapping. In Fig. 6a the fraction of successfully mapped topologies is drawn against the fraction of deleted edges f for a constant number of nodes $N = 49$. We observe that the fraction of mapped topologies is substantially higher for SMART that for Tabu97. Fig. 6a also confirms that, in general, it is more difficult to map the same logical topology on a sparser physical topology.

Fig. 6b depicts the dependence of the efficiency on the size of topologies. The advantage of SMART grows with N; for $N = 16$ it is negligible but for $N = 100$ it already reaches 30%. Again, this is because SMART divides the whole task into tiny subtasks. For a small N there is not much to divide and SMART cannot take advantage of this property.

1 A full lattice of 16 nodes has 24 edges. So for $f = 0.35$, the corresponding f–lattice will have $24(1-0.35) \simeq 16$ edges, which is the smallest value still enabling 2–edge–connectivity (a cycle topology).

It is interesting to investigate the run–times of the algorithms. Since Tabu97 was integrated with our implementation of SMART (both in C++, using the same structures and functions), it is reasonable to compare their real processing times. They are given in Fig. 6c in log-log scale. The curves are almost linear, hence the observed complexities of the algorithms are polynomial, with $O(N^{3.5})$ for Tabu97 and $O(N^{2.4})$ for SMART. Both values fit in the theoretical maximal bounds, which are $O(N^4)$ [5] and $O(N^3)$, respectively. Note that Tabu97 took about *11 hours* when solving 900 node problem, which is a lot more than the *25 seconds* measured for SMART. Fig. 6d is a direct comparison of run–times of the two algorithms. SMART converged orders of magnitude faster than Tabu97; again the difference strongly depends on N.

5.1.3. Simple Layout Algorithm

The Simple Layout Algorithm by Sasaki et al. [12], similarly to SMART, breaks down the survivable mapping problem into a set of small and easy to solve subproblems. Therefore we expected it to be as fast as SMART. We implemented the version of the Simple Layout Algorithm with multiple link computation, *DEGR* node ordering and *EL–M* link cost (the detailed description can be found in [12]). The authors reported this set of parameters to be the most efficient. The physical and logical topologies were the same as described in the previous subsection (Tabu Search). The run–times of the Simple Layout Algo-

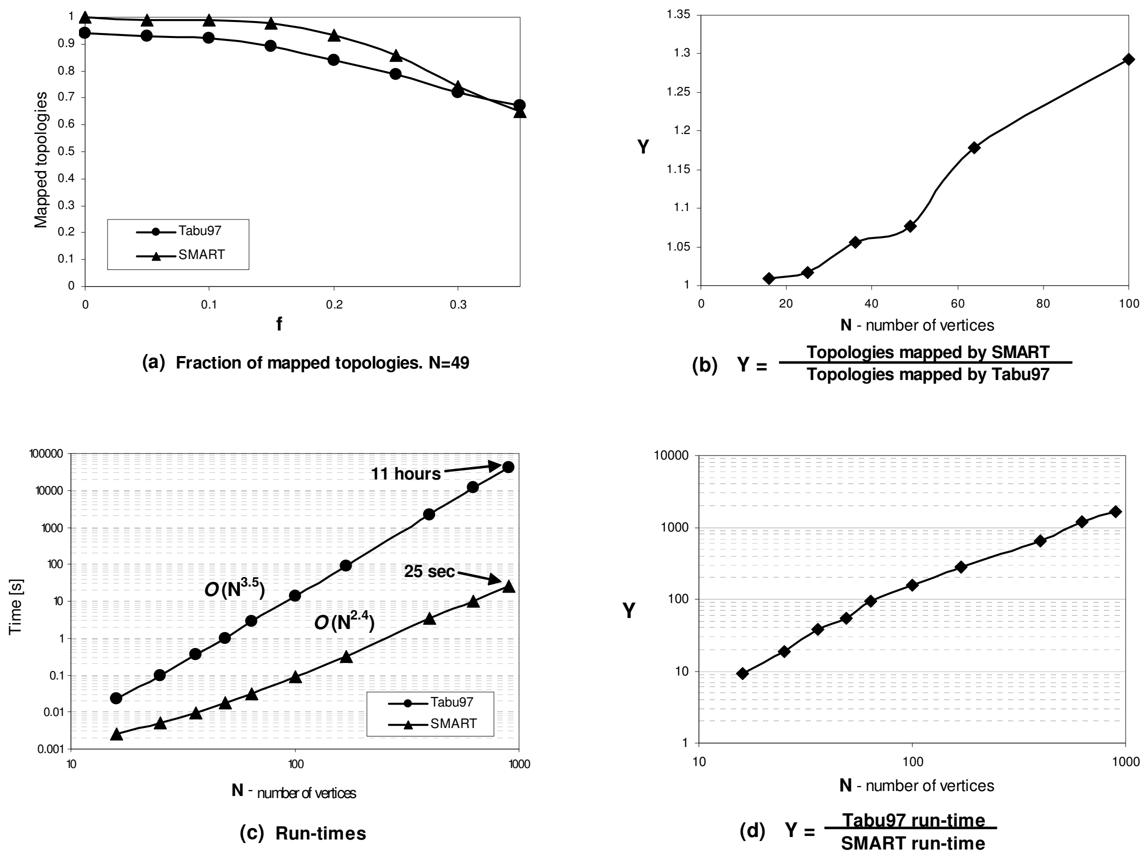

Figure 6. SMART vs. Tabu97. Random graph logical topology of average node degree $\bar{d} = 4$ mapped on f-lattice physical topology of N nodes, f=0...0.35, N=16...900. A pair of topologies is declared to be 'mapped,' if it is mapped in a 1-survivable way.

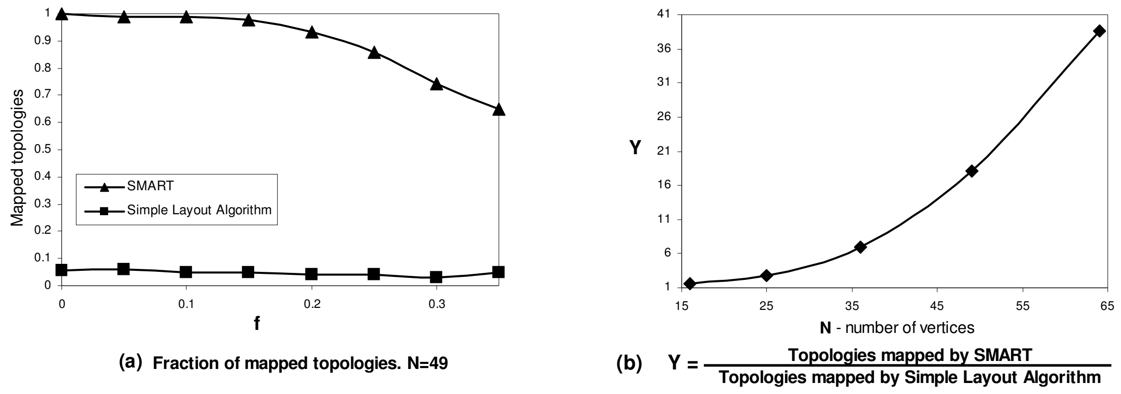

Figure 7. SMART vs. Simple Layout Algorithm. A pair of topologies is declared to be 'mapped,' if it is mapped in a 1-survivable way.

rithm were about three times shorter than those of SMART. However, as illustrated in Fig. 7, the fraction of topologies mapped (in a 1-survivable way) by Simple Layout Algorithm is dramatically smaller than the fraction mapped by SMART. Although for $N = 16$ the results are comparable, the $N = 64$ already yields a forty times difference. The reason for the poor efficiency of the Simple Layout Algorithm and its strong dependence on the size of topologies is the following. By construction, the Simple Layout Algorithm prevents only *single* nodes from being separated in the case of a fiber failure. Since in small graphs (authors used six–node topologies for simulations) a separation of a single node is the most common type of loss of connectivity, the Simple Layout Algorithm approach is efficient. But for large graphs there are substantially more possibilities for the separation of larger (than a single node) subgraphs. The Simple Layout Algorithm does not take them into account, which results in a dramatic fall in its efficiency. This is yet another evidence that the survivability is a complex problem and requires relatively sophisticated algorithms.

5.2. Span failure survivability

For the illustration of a SMART-Span version of our algorithm (see Section 4.1) we took the ARPA2 network and assumed two multi-link spans as depicted in Fig. 5b. The logical topologies were 2–edge–connected random graphs of average node degree $\bar{d}$ =3…6. We generated 1000 logical topologies for each $\bar{d}$. The results are presented in Table 2. The Shortest Path algorithm and SMART (with no span protection) are added for reference. The results show that even very few multi-link spans may result in a loss of survivability when the mapping does not take them into account. However, SMART-Span enables an efficient protection. The run–times of SMART and SMART-Span (not shown here) are comparable.

Average degree $\bar{d}$	Shortest Path	SMART	SMART-Span
3	999	681	183
4	994	373	64
5	965	177	8
6	867	129	1

Table 2. Number of topologies *failed* to be mapped in a span-survivable way (out of 1000).

5.3. Node failure survivability

SMART-Node (see 4.2) was tested in the same setting as SMART-Span in previous section, except that the physical topology was an ARPA2 network with no modifications, as in Fig. 5a. The results of the simulations are presented in Table 3. SMART-Node performs fairly well and its run–time (not shown here) is only slightly higher than that of ‘pure’ SMART.

Average degree d	Shortest Path	SMART	SMART-Node
3	999	903	428
4	995	717	204
5	963	410	23
6	859	228	3

Table 3. Number of topologies *failed* to be mapped in a node-survivable way (out of 1000).

5.4. 2–survivability

In order to obtain 2-survivability, we applied SMART-DF (see Section 4.3). Since the necessary condition for 2–survivability is the 3–edge–connectedness of the physical and logical topologies, we used NSFNET3EC (see Fig. 5d) as the physical topology and 3–edge–connected random graph of average vertex degree $\bar{d} = 5 \ldots 7$ as the logical topology. Note that the degree is larger than in previous examples, because 2-survivability naturally requires the topologies to be connected more strongly.
Table 4 presents the results. Clearly ‘pure’ SMART is completely inefficient when dealing with double–link failures, whereas, SMART-DF performs a lot better. The run–time of SMART-DF (not shown here) was three times longer than that of SMART.

Average degree $\bar{d}$	Shortest Path	SMART	SMART-DF
5	1000	998	422
6	992	971	36
7	950	924	3

Table 4. Number of topologies *failed* to be mapped in a 2–survivable way (out of 1000).

We also tested SMART-DF for larger topologies, with N=49. The physical topology was a 3–edge–connected f–lattice, where $f = 0.1$. The four corner nodes of this f–lattice were additionally connected to the closest nodes (on diagonal) to make the 3–edge–connectivity possible. The logical topology was a 3–edge–connected random graph of average node degree equal to 7. SMART-DF mapped 732 out of 1000 topologies. The average run–time was 0.08 sec, which is only 4 times longer than a run–time of pure SMART in the same scenario. These times were captured in runs *without* verification of 2–survivability. This is because the process of verification turned out to play a very important role for double-link failures. The average 2–survivability verification time was twice as long as the time of the entire mapping by SMART-DF itself; it lasted about 0.15 sec.

This is because the connectivity test of the logical topology must be run for each of $|E^\phi|(|E^\phi|-1)/2$ fiber pairs ($|E^\phi|$ - number of edges in the physical topology). Note that 1–survivability verification would require only $|E^\phi|$ tests, so 2–survivability verification is $(|E^\phi|-1)/2$ times more complex than 1-survivability verification. In our case $|E^\phi|=80$, yielding a multiplication by almost 40. Indeed, the average 1–survivability verification in this scenario lasted about 0.004sec$\simeq$ 0.15sec/40. Now recall that in standard heuristic approaches (e.g., Tabu Search or Simulated Annealing) the temporary solution is evaluated (verified) many times before converging to a final one. Therefore the overall run–time of those heuristics is dominated by the time of solution evaluation. Since the 2–survivability verification is $(|E^\phi|-1)/2$ times more complex than 1–survivability verification, the scalability of e.g. Tabu Search looking for 2–survivable mapping would be dramatically limited, contrary to the SMART-DF approach.

6. Conclusions and future work

The SMART algorithm seems to be a very promising technique for constructing a survivable mapping in WDM networks. Our simulations have shown that SMART works about 2-3 orders of magnitude faster than the previous proposals and is more scalable. Moreover the SMART approach proved suitable to deal with the failures of fiber spans, nodes and independent fiber pairs. This was probably the first time these scenarios were addressed with IP restoration.
We have left the formal analysis and the proof of correctness of the SMART algorithm for the future work. We will also work on more realistic applications including capacity constraints, wavelength continuity and traffic scenarios.

7. Acknowledgements

We would like to thank Eytan Modiano and Aradhana Narula-Tam for sending us the topologies they used in [6], and Frederick Ducatelle for interesting discussions and some benchmarking results.

References

[1] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee. Fault management in IP-Over-WDM Networks: WDM Protection vs. IP Restoration. *IEEE Journal on Selected Areas in Communications*, 20(1), January 2002.

[2] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier-1 backbone. *Sprint ATL Research Report Nr. RR03-ATL-030666*.

[3] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C-N. Chuah, and C. Diot. Characterization of Failures in an IP Backbone. *Proc. of IEEE INFOCOM'04*, 2004.

[4] G. Li, B. Doverspike, and Ch. Kalmanek. Fiber Span Failure Protection in Mesh Optical Networks. *Optical Networks Magazine*, 3(3):21–31, May/June 2002.

[5] J. Armitage, O. Crochat, and J. Y. Le Boudec. Design of a Survivable WDM Photonic Network. *Proceedings of IEEE INFOCOM 97*, April 1997.

[6] E. Modiano and A. Narula-Tam. Survivable lightpath routing: a new approach to the design of WDM-based networks. *IEEE Journal on Selected Areas in Communications*, 20(4):800–809, May 2002.

[7] F. Giroire, A. Nucci, N. Taft, and C. Diot. Increasing the Robustness of IP Backbones in the Absence of Optical Level Protection. *Proc. of IEEE INFOCOM 2003*, 2003.

[8] E. Leonardi, M. Mellia, and M. Ajmone Marsan. Algorithms for the Logical Topology Design in WDM All-Optical Networks. *Optical Networks Magazine*, January 2000.

[9] O. Crochat and J. Y. Le Boudec. Design Protection for WDM Optical Networks. *IEEE Journal of Selected Areas in Communication*, 16(7):1158–1165, September 1998.

[10] A. Nucci et al. Design of Fault-Tolerant Logical Topologies in Wavelength-Routed Optical IP Networks. *Proc. of IEEE Globecom 2001*, 2001.

[11] A. Fumagalli and L. Valcarenghi. IP Restoration vs. WDM Protection: Is There an Optimal Choice? *IEEE Network*, Nov/Dec 2000.

[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] S. il Kim and S. Lumetta. Addressing node failures in all-optical networks. *Journal of Optical Networking*, 1(4):154–163, April 2002.

[14] H. Choi, S. Subramaniam, and H.-A. Choi. On Double-Link Failure Recovery in WDM Optical Networks. *Proc. of IEEE INFOCOM'02*, 2002.

[15] W. He, M. Sridharan, and A. K. Somani. Capacity Optimization for Surviving Double-Link Failures in Mesh-Restorable Optical Networks. *Proc. of OptiComm'02*, 2002.

[16] M. Clouqueur and W. D. Grover. Mesh-restorable Networks with Complete Dual-failure Restorability and with Selectively Enhanced Dual-failure Restorability Properties,. *Proc. of OptiComm'02*, 2002.

[17] J. Gross and J. Yellen. *Graph Theory and its Applications*. CRC Press, 1999.

[18] A. Frank. *Packing paths, circuits and cuts - a survey (in Paths, Flows and VLSI-Layout)*. Springer, Berlin, 1990.