

# gStreams: A new technique for fast recovery with capacity efficient protection in WDM mesh networks

Arunabha Sen and Sudheendra Murthy  
Dept. of Computer Science & Engineering  
Arizona State University  
Tempe, Arizona 85281  
Email: {asen, sudhi}@asu.edu

Subir Bandyopadhyay  
School of Computing Science  
University of Windsor  
Windsor, Ontario N9B 3P4, Canada  
Email: subir@uwindsor.ca

**Abstract**—In a recent paper, Kim and Lumetta [6] proposed a capacity efficient protection scheme that provides fast recovery in WDM mesh networks. They introduced the notion of a *stream* that is utilized for this purpose. In this paper, we introduce the concept of *gStream*, which is a more generalized form of stream and develop efficient algorithms for maximizing capacity utilization without sacrificing the benefit of fast recovery. We show that the problem of finding the set of gStreams that maximizes capacity utilization is NP-complete. We present (i) an optimal solution for formation of streams and gStreams and (ii) a heuristic solution. The results of our experimental evaluation show that our heuristic provides near optimal solution to almost all instances of the problem in a fraction of the time needed for finding the optimal solution.

## I. INTRODUCTION

Survivability of all-optical networks has become an important area of research in recent years as evidenced by the increased attention the topic has received among the researchers [3], [9], [10]. Two techniques, *protection* at the WDM layer and *restoration* at the IP layer have emerged to be the leading contenders for fault management in optical networks [4], [7]–[9]. Each of these two schemes has its own advantages and disadvantages. In general, restoration is slow but capacity efficient, whereas protection is fast but less efficient in terms of network capacity utilization. This is due to the fact that the protection schemes compute a *backup path* and reserve network capacity (wavelength) on the backup path whenever a primary path between a source-destination node pair is established. The protection scheme can be further divided into two groups - *path protection* and *link protection*. The path protection scheme can be further subdivided into i) (1+1) scheme and ii) (1:1) scheme. In (1+1) scheme, the traffic is sent from the source to the destination through the primary and the backup path *all the time*. As such, this scheme provides fast recovery with minimal loss of data due to the failure of a primary path. In the (1:1) scheme, network capacity is reserved on the backup path but is not used to transfer data, unless there is a failure on the primary path.

Thus, the resources on the optical fiber links can be shared among multiple backup paths as long as their primary paths are disjoint. Moreover, the reserved capacity can be utilized to carry low priority traffic that can be preempted in case of failure on a primary path. In *Dedicated Path Protection* (DPP) schemes, the backup paths do not share network resources. In *Shared Path Protection* (SPP) schemes, two backup paths are allowed to share network resources when the corresponding primary paths are edge disjoint. The photonic cross-connects (PXC) are not configured ahead of a failure in SPP schemes, as they are in DPP schemes. As a result, the SPP scheme is slower in comparison with the DPP scheme. However, in terms of capacity utilization, the SPP scheme is considerably more efficient than the DPP scheme as it allows for sharing of resources on the backup paths.

Clearly, there is a trade-off between time-to-recover and capacity utilization when we compare the SPP scheme with the DPP scheme. Recently in [6], the authors proposed a scheme where they introduced a notion of *stream*, which is a collection of *overlapping* backup paths. The stream scheme can be conceptualized as a member of a *virtually shared* DPP scheme. In this scheme, all the PXC are preconfigured (just like DPP schemes) and in case of a failure, the traffic is sent through the backup path that forms a part of a stream. This scheme results in fast recovery as the PXC are not required to spend any time for decision making at the time of a failure. The speed of recovery in this scheme is much faster than that of SPP schemes as no signalling or configuration of PXC in the intermediate nodes lying on the backup path is needed after a failure [6]. The resource utilization in this scheme is much better than that of DPP schemes as it allows some sharing of resources on the backup paths between multiple connections. Thus, the scheme retains some of the advantages of both the DPP and the SPP. In the current taxonomy of schemes for survivability in optical networks, one can view a stream as a scheme that lies somewhere between the SPP and the DPP.

In addition to introducing the concept of stream, Kim and Lumetta in [6] proposed algorithms for the formation of streams. Through extensive simulations they demonstrated

Supported in part by ARO grant W911NF-06-1-0354

the efficacy of their methods. In this paper, we propose an expanded definition of streams called *gStreams*. The new definition subsumes the definition of stream given in [6]. The contributions of this paper are as follows.

- We introduce a novel concept of a *gStream* that generalizes the notion of a stream for better resource utilization.
- We show that the problem of finding the optimal number of *gStreams* (or streams) that maximizes capacity utilization is NP-complete even when the primary and backup paths are fixed in the network.
- We give a mathematical programming formulation for finding the optimal streams and *gStreams*.
- We provide a heuristic solution that computes near optimal streams and *gStreams* to most of the problem instances.
- Through extensive simulation, we demonstrate the benefits of this new concept and the efficacy of our heuristics in finding near optimal solutions.

Since the algorithm proposed by Kim and Lumetta is for a dynamic scenario where the streams are being formed as and when the call requests arrive and our heuristic is for a static scenario, the results produced by these two methods are not directly comparable. However, our results can be used as an upper bound on the capacity savings that can be achieved for the generated set of backup paths. In some sense, Kim and Lumetta provide an *online* algorithm for the problem, whereas ours is an *offline* algorithm.

## II. STREAM PROTECTION SCHEME

Although a *stream* was not *formally* defined in [6], the authors implied it to be a directed path from node  $v_{i_1}$  to  $v_{i_k}$  ( $v_{i_1} \rightarrow v_{i_2} \rightarrow v_{i_3} \rightarrow \dots \rightarrow v_{i_{k-2}} \rightarrow v_{i_{k-1}} \rightarrow v_{i_k}$ ) formed by overlapping backup paths that connect multiple source-destination node pairs. For example, the stream from  $v_{i_1}$  to  $v_{i_k}$  may be formed by the concatenation of the overlapping backup paths (i)  $v_{i_1} \rightarrow v_{i_2} \rightarrow v_{i_3} \rightarrow \dots \rightarrow v_{i_{k-2}}$ , (ii)  $v_{i_2} \rightarrow v_{i_3} \rightarrow v_{i_4} \rightarrow \dots \rightarrow v_{i_{k-1}}$  and (iii)  $v_{i_3} \rightarrow v_{i_4} \rightarrow v_{i_5} \rightarrow \dots \rightarrow v_{i_k}$ . The notion of backup paths sharing network resources is not new and is the key feature of the SPP schemes.

However, arbitrary sharing of backup paths may lead to a situation, in which preconfiguration of the PXC's on the backup paths may not be possible. Consider the network shown in Figure 1.

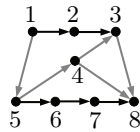


Fig. 1. Stream formation

Let the primary paths connecting the nodes (1,3) and (5,8) be  $1 \rightarrow 2 \rightarrow 3$  and  $5 \rightarrow 6 \rightarrow 7 \rightarrow 8$  respectively. Let the respective backup paths be  $1 \rightarrow 5 \rightarrow 4 \rightarrow 3$  and  $5 \rightarrow 4 \rightarrow 8$ . Suppose the two backup paths use the same wavelength, then the PXC at node 4 cannot be preconfigured before the occurrence of a fault on the primary paths, since node 4 cannot decide ahead of a fault whether to forward the traffic coming on the backup lightpath to node 3 or 8. However, instead of using  $5 \rightarrow 4 \rightarrow 8$  as the backup path from 5 to 8, if  $5 \rightarrow 4 \rightarrow 3 \rightarrow 8$  is used, the PXC at node 4 can be preconfigured as there

is no need to wait until the occurrence of a failure. Irrespective of the failure of the primary path from 1 to 3 or 5 to 8, the PXC at 4 has to forward the traffic to node 3. The key idea of the streams scheme is to form the backup paths in such a way that *divergence* of the paths are not necessary. The absence of divergence of overlapping backup paths eliminates the need to wait until the occurrence of a failure on the primary path to configure a PXC thereby ensuring faster recovery. In this way, the use of the concept of stream can provide performance comparable to that of a DPP scheme. In the example of figure 1 path  $1 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 8$  is a *stream* formed by the overlapping backup paths  $1 \rightarrow 5 \rightarrow 4 \rightarrow 3$  and  $5 \rightarrow 4 \rightarrow 3 \rightarrow 8$ . In order to get DPP performance for fast recovery, with the backup paths  $1 \rightarrow 5 \rightarrow 4 \rightarrow 3$  and  $5 \rightarrow 4 \rightarrow 8$  (without stream formation), five units of network resources (wavelengths) have to be reserved:  $\lambda_1$  on the links (1,5), (5,4) and (4,3) and  $\lambda_2$  on the links (5,4) and (4,8). However, if a stream,  $1 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 8$  is formed, it will require only four units of resources:  $\lambda_1$  on the links (1,5), (5,4), (4,3) and (3,8). Thus, the formation of streams can provide fast recovery (comparable to DPP) with better capacity utilization. We will refer to the difference between the resource usage in a non-stream solution (5 units) to the stream solution (4 units) as the *gain*. The objective will be to form the streams in such a way that will *maximize* the gain.

### A. Current approach to stream formation

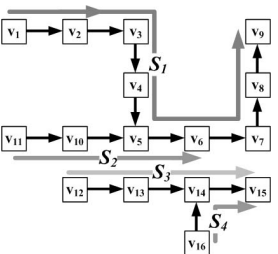
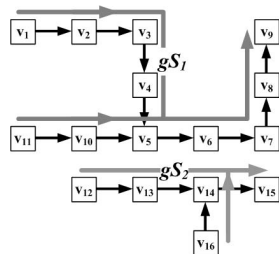
The stream formation algorithm given in [6] assumes a dynamic scenario where call requests arrive at random time intervals. A set of shortest paths,  $P(s, d)$  between each source-destination node pair is precomputed. In addition, corresponding to every  $p_i \in P(s, d)$ , a set of backup paths  $B(p_i, h)$  is also precomputed. The parameter  $h$  is used to restrict the length of potential backup paths to at most  $h$  hops more than the length of the shortest paths. As soon as a request for connection from a source node  $s$  to a destination node  $d$  arrives, the traffic manager chooses one primary and one backup path from the set of precomputed primary and backup paths from  $s$  to  $d$ . The primary and backup path-pair is chosen in such a way that the backup path is “compatible” with the existing streams and hence, will impose least increase in “cost” for the new connection request. A detailed description of the algorithm can be found in [6].

### B. Limitations of the current approach

The advantage of the stream formation scheme as given in [6] is its simplicity. It takes a simple *greedy* approach to the formation of the streams with the backup paths. However, the disadvantage of such a simple scheme is that the solution produced by this approach may be far from optimal. In the following, we provide two examples where the simple approach fails to realize the full benefit of stream formation.

**Case 1:** Suppose that at an instance of time  $t$ , there are only two non-overlapping streams in existence  $S_1: v_i \rightarrow v_{i+1} \rightarrow \dots \rightarrow v_j \rightarrow v_{j+1} \rightarrow \dots \rightarrow v_k$  and  $S_2: v_a \rightarrow v_{a+1} \rightarrow v_{a+2} \rightarrow \dots \rightarrow v_b \rightarrow v_{b+1} \dots \rightarrow v_c$ . If at this time a new backup path

TABLE I  
AN EXAMPLE DEMONSTRATING THE BENEFITS OF gSTREAMS OVER STREAMS

Backup Paths	Streams	gStreams
$B_1: v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$ $B_2: v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_8$ $B_3: v_7 \rightarrow v_8 \rightarrow v_9$ $B_4: v_{10} \rightarrow v_5 \rightarrow v_6$ $B_5: v_{11} \rightarrow v_{10}$ $B_6: v_{12} \rightarrow v_{13} \rightarrow v_{14}$ $B_7: v_{13} \rightarrow v_{14} \rightarrow v_{15}$ $B_8: v_{16} \rightarrow v_{14} \rightarrow v_{15}$		
(Primary paths $P_1, P_2, \dots, P_8$ are all mutually disjoint)	$S_1 (B_1+B_2+B_3): v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_8 \rightarrow v_9, \text{Gain} = 3$ $S_2 (B_4+B_5): v_{11} \rightarrow v_{10} \rightarrow v_5 \rightarrow v_6, \text{Gain} = 0$ $S_3 (B_6+B_7): v_{12} \rightarrow v_{13} \rightarrow v_{14} \rightarrow v_{15}, \text{Gain} = 1$ $S_4 (B_8): v_{16} \rightarrow v_{14} \rightarrow v_{15}, \text{Gain} = 0$ <b>Total Gain = 4</b>	$gS_1 := B_1+B_2+B_3+B_4+B_5, \text{Gain} = 4$ $gS_2 := B_6+B_7+B_8, \text{Gain} = 2$ <b>Total Gain = 6</b>

$v_j \rightarrow v_{j+1} \rightarrow \dots \rightarrow v_k \rightarrow v_{k+1} \rightarrow \dots \rightarrow v_a \rightarrow v_{a+1} \rightarrow \dots \rightarrow v_b$  is selected, the algorithm proposed in [6] combines this backup path either with the stream  $S_1$  or with  $S_2$ . It may be noted however that as the new backup path shares links with both  $S_1$  and  $S_2$ , the two streams can be combined into a single stream  $S_3: v_i \rightarrow v_{i+1} \rightarrow \dots \rightarrow v_j \rightarrow v_{j+1} \rightarrow \dots \rightarrow v_k \rightarrow v_{k+1} \rightarrow \dots \rightarrow v_a \rightarrow v_{a+1} \rightarrow \dots \rightarrow v_b \rightarrow v_{b+1} \rightarrow \dots \rightarrow v_c$  with the help of the newly arrived backup path, thereby increasing capacity utilization.

**Case 2:** Suppose that at an instance of time  $t$ , there is only one stream in existence  $S_1: v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5$  formed by only one backup path  $B_1$ . At this time, a second backup path  $B_2: v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6$  is selected. The stream formation technique proposed in [6] combines the newly arrived backup path with the stream  $S_1$  to form a new stream  $S_2: v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6$ . Suppose a third backup path  $B_3: v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_7$  is selected at this time.  $B_3$  cannot be combined with the stream  $S_2$  as this will lead to divergence at node  $v_5$ . The net gain of combining  $S_1$  with  $B_2$  will be 2 units as  $S_1$  and  $B_2$  have overlapping path segment  $v_3 \rightarrow v_4 \rightarrow v_5$ . However, if  $S_1$  had not been combined with  $B_1$ , but combined with  $B_3$  to form the stream  $S_3: v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_7$ , the gain would have been 3 units since  $S_1$  and  $B_3$  have overlapping path segment  $v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5$ .

### C. Definition of gStream

As noted earlier, although *stream* was not formally defined in [6], the authors implied it to be a directed path formed by the concatenation of overlapping backup paths. Two backup paths  $B_1$  and  $B_2$  can form a stream as long as there is no divergence between these two paths. For example, backup paths  $B_1: v_1 \rightarrow v_2 \rightarrow v_3$  and  $B_2: v_2 \rightarrow v_3 \rightarrow v_4$  can form a stream  $S_1: v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$ . However, paths  $B_1: v_1 \rightarrow v_2 \rightarrow v_3$  and  $B_2: v_4 \rightarrow v_2 \rightarrow v_5$  cannot be concatenated to form a stream as it leads to diverge at node  $v_2$ . We make a key observation regarding the formation of streams. We note that in order to avoid divergence, combination of

two overlapping backup paths  $B_1$  and  $B_2$  need not always form a path. Divergence can still be avoided if for example,  $B_1: v_1 \rightarrow v_2 \rightarrow v_3$  and  $B_2: v_4 \rightarrow v_2 \rightarrow v_3$ . Although the combination of  $B_1$  and  $B_2$  does not form a path, it still retains the important property that all nodes in this combined structure can be preconfigured. We refer to such a structure as generalized streams(gStreams). We give a formal definition of a gStream next.

- **Outdegree-1 Graph (OD1G):** A directed graph is called an Outdegree-1 Graph if the outdegree of each node is at most one.
- **gStream:** A connected OD1G is called a generalized stream.

The example in Table I shows the streams and gStreams formation with 8 backup paths. In the example, suppose that the primary paths are all mutually disjoint and there are at least 2 wavelengths that can be used on all the backup paths. The directed path definition of streams results in the formation of 4 streams (second column) resulting in a total gain of 4 units. However, the OD1G definition of gStreams results in 2 gStreams (third column) with a net total gain of 6 units. Since a OD1G has outdegree at most one, there never is any divergence. Therefore, we can combine backup paths to form gStream as long as the combined backup paths form a OD1G. It may be noted that a directed path, directed cycle, Reverse-Branching and Reverse-Arborescence [5] are examples of Outdegree-1 Graphs. Thus, the definition of stream as a path used in [6] is a special case of gStream.

### III. PROBLEM FORMULATION

In the most general form of the problem, we have a collection of source-destination node pairs  $(s_i, d_i), 1 \leq i \leq k$  and corresponding to each  $(s_i, d_i)$ , we would like to establish a primary path  $P_i$  and a backup path  $B_i$  (node disjoint from  $P_i$ ) such that the network resources used for the establishment of these paths is minimized. The channels on optical fiber links are the network resources that we plan to optimize. If

a primary path  $P_i$  is established over three optical fiber links using wavelength  $\lambda$ , we say that the path used three units of network resources. Under SPP scheme, two backup paths  $B_i$  and  $B_j$  can share a channel over a link if the corresponding primary paths are disjoint. We assume that no wavelength converters are available and as such, wavelength continuity constraint must be satisfied by all primary and backup paths. In this paper, we do not address the issues related to the selection of primary and backup paths. Rather, we address the issue of stream formation from the set of backup paths such that resource utilization is maximized. We are assuming that based on some considerations, the primary and the backup paths between the source-destination node pairs have already been selected and our job is to form the streams in a way that maximizes resource utilization. We prove that even when the set of wavelengths  $\Lambda_i$  that can be used to establish a backup path  $B_i$  is equal to  $\Lambda$ , the set of wavelengths available on a fiber link, the stream formation for maximum resource utilization problem is NP-complete.

#### A. Definitions

- **Compatible backup paths:** Two backup paths  $B_i$  and  $B_j$  are said to be *compatible* (denoted by  $B_i \Leftrightarrow B_j$ ) if (i) their primary paths  $P_i$  and  $P_j$  are node disjoint (ii) there is no divergence in the structure created by their concatenation and (iii)  $B_i, B_j$  can be realized using the same wavelength  $\lambda$  (wavelength continuity). It may be noted that compatibility is not transitive.
- **Gain of a pair of backup paths:** If two compatible backup paths  $B_i$  and  $B_j$  are combined to form a stream, then the *gain* associated with this pair of paths denoted by  $g(i, j)$  is equal to the number of common arcs between them. The gain of a pair of compatible paths is a non-negative number. If  $B_i \not\Leftrightarrow B_j$ , then  $g(i, j) = -\infty$ .
- **Path Intersection Graph (PIG):** The path intersection graph  $H(V_H, E_H, G_H, L_H)$  corresponding to a set of backup paths  $\{B_1, B_2, \dots, B_n\}$  is an edge-weighted, *completely* connected, undirected graph, in which each vertex  $v_i$  represents a backup path  $B_i$ .  $G_H$  is the set of gain values  $g(i, j)$  for every edge  $(v_i, v_j) \in E_H$ .  $L_H$  is the set of wavelength lists  $\Lambda_i$  for  $v_i \in V_H$  where  $\Lambda_i$  represents the set of wavelengths that can be used throughout on backup path  $B_i$ .
- **Gain of a vertex set:** If  $V \subseteq V_H$ , then the gain associated with  $V$  denoted by  $g(V)$  is given by  $\sum_{v_i, v_j \in V} g(v_i, v_j)$ .
- **Gain of a graph partition:** If  $\mathcal{V} = \{V_1, V_2, \dots, V_k\}$  represents a partition of the node set  $V_H$  of a graph, then the gain associated with this partition denoted by  $g(\mathcal{V})$  is given by  $g(\mathcal{V}) = \sum_{i=1}^k g(V_i)$ .

#### B. Graph Partitioning for Gain Maximization (GPGM) Problem

*Instance:* Given a Path Intersection Graph  $H(V_H, E_H, G_H, L_H)$  with  $V_H, E_H, G_H$  and  $L_H$  as defined before and a positive integer  $R$ .

*Question:* Is there a partition of  $V_H$  into  $\mathcal{V} = \{V_1, V_2, \dots, V_k\}$  with  $g(\mathcal{V}) \geq R$  such that

- 1) there is at least one wavelength that can be assigned to all vertices belonging to the same component  $V_i$ . That is,  $\forall l, 1 \leq l \leq k, \bigcap_{v_i \in V_l} \Lambda_i \neq \emptyset$ , and
- 2) two backup paths whose concatenation results in divergence cannot be assigned the same wavelength.

It may be noted that in the partition  $\mathcal{V}$  that has the maximum gain, a component  $V_i \in \mathcal{V}$  does not contain  $-\infty$  gain edges. Each component represents a connected OD1G of the backup paths in the original network and thus, a gStream.

#### IV. COMPUTATIONAL COMPLEXITY OF GPGM PROBLEM

Consider the restricted version of the GPGM problem, in which all wavelengths in the network are available for use at every backup path. That is,  $\forall v_i \in V_H, \Lambda_i = \Lambda$ . This assumption eliminates the wavelength constraints in the GPGM problem. We prove that this simplified version of the GPGM problem termed as RGPGM problem itself is NP-complete.

##### A. Restricted Graph Partitioning for Gain Maximization (RGPGM) Problem

*Instance:* Given a Path Intersection Graph  $H(V_H, E_H, G_H)$  with  $V_H, E_H$  and  $G_H$  as defined before, positive integer  $R$ .

*Question:* Is there a partition of  $V_H$  into  $\mathcal{V} = \{V_1, V_2, \dots, V_k\}$  such that  $g(\mathcal{V}) \geq R$  ?

To prove the NP-completeness of the RGPGM problem, we transform Exact Cover by 3-Sets (X3C) [2] into RGPGM problem.

**Theorem 1:** RGPGM problem is NP-complete.

*Proof:* Clearly, RGPGM problem is in NP since a non-deterministic algorithm need only guess a partition  $\mathcal{V}$  of vertex set  $V_H$  and verify in polynomial time whether  $g(\mathcal{V}) \geq R$ .

Suppose a finite set  $X$  with  $|X| = 3q$  and a collection  $C = \{c_1, \dots, c_k\}$  of 3-element subsets of  $X$  make up the instance of X3C. From this instance of X3C, we will construct an instance of RGPGM such that a partition  $\mathcal{V}$  of the node set  $V_H$  with  $g(\mathcal{V}) \geq R$  exists if and only if  $C$  contains an exact cover. We use local replacement technique for the NP-completeness proof. Corresponding to each subset  $c_i \in C$  ( $c_i = \{x_i, y_i, z_i\}$ ), we will construct a subgraph of  $G$  with 12 nodes and 18 edges, as shown in figure 2. Since this construct corresponds to the 3-element subset  $c_i$ , we will refer to the edges of this construct as  $E_i$ . Thus, the instance of the RGPGM will have  $V_H$  vertices where  $V_H = X \cup \bigcup_{i=1}^{|C|} \{a_i^j : 1 \leq j \leq 9\}$ . The instance of the RGPGM will have two types of edges,  $E_A$  and  $E_B$ , where  $E_A = \bigcup_{i=1}^{|C|} E_i$ . It may be recalled that the instance of the RGPGM problem is an edge-weighted, *completely* connected undirected graph. If two nodes  $v_i$  and  $v_j$  are not connected by an edge  $e \in E_A$ , then there must be an edge  $e' \in E_B$  connecting these two nodes. The weight on all edges  $e \in E_A$  is 1 and the weight on all edges  $e' \in E_B$  is  $-\infty$ . Finally, we set  $R = 3q + 9|C|$ . The number of vertices of the instance of RGPGM will be  $|X| + 9|C| = 3q + 9|C|$  i. e.,  $|V_H| = 3q + 9|C| = 3q'$ , where  $q' = q + 3|C|$ . It is easy to

see that this construction procedure can be done in polynomial time.

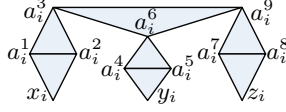


Fig. 2. Local replacement for  $c_i = \{x_i, y_i, z_i\} \in C$  for transforming X3C to RGPGM.

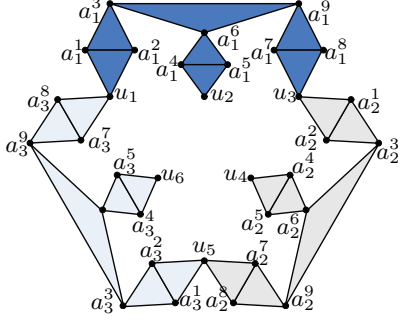


Fig. 3. Local replacement for 3 subsets  $c_1 = \{u_1, u_2, u_3\}$ ,  $c_2 = \{u_3, u_4, u_5\}$  and  $c_3 = \{u_1, u_5, u_6\}$

*Claim:* There is a partition  $\mathcal{V}$  of the instance of the RGPGM problem with  $g(\mathcal{V}) \geq R$ , if and only if the instance of the X3C has an exact cover.

Suppose that  $c_1, c_2, \dots, c_q$  are the 3-element subsets from  $C$  that make up an exact cover of  $X$ . In this case, we can partition the node set  $V_H$  into  $V_1 \cup V_2 \cup \dots \cup V_{q'}$  by taking  $\{a_i^1, a_i^2, x_i\}, \{a_i^4, a_i^5, y_i\}, \{a_i^7, a_i^8, z_i\}, \{a_i^3, a_i^6, a_i^9\}$  from the vertices of  $E_i$ , whenever  $c_i = \{x_i, y_i, z_i\}$  is in the exact cover, and by taking  $\{a_i^1, a_i^2, a_i^3\}, \{a_i^4, a_i^5, a_i^6\}, \{a_i^7, a_i^8, a_i^9\}$  from the vertices meeting  $E_i$ , whenever  $c_i$  is not in the exact cover. Thus if there is an exact cover in the instance of X3C, the instance of the RGPGM can be partitioned into  $q'$  subsets  $V_1, \dots, V_{q'}$ , where each  $V_i, 1 \leq i \leq q'$  is a completely connected subgraph (clique) of size 3. Since the edge weights associated with each edges that form the clique is 1, the gain associated with each clique  $V_i, 1 \leq i \leq q'$  is 3. Since  $V_H$  has been partitioned into  $q'$  cliques, the gain of this partition is  $3q' = 3q + 9|C|$ . Thus, if there is an exact cover, there is a partition with gain  $R$ .

Conversely, suppose that there exists a partition  $\mathcal{V}$  of the instance of RGPGM such that  $g(\mathcal{V}) = R = 3q' = 3q + 9|C|$ . First we show that if such a partition exists, it must be possible to partition the instance of RGPGM into cliques of size 3, such that all edges belonging to the cliques have weight 1. Then, we show that if such a partition into cliques of size 3 with all edge weights 1 exists, the instance of X3C must have an exact cover.

It may be noted that the way the instance of the RGPGM was created, the graph does not contain a clique of size 4 or higher with all edge weights equal to 1. Suppose that the graph  $H$  is partitioned into cliques of size 1, 2 and 3 and suppose that the partition  $\mathcal{V}$  is composed of  $K_1$  cliques of

size 1,  $K_2$  cliques of size 2 and  $K_3$  cliques of size 3. In this case,  $|V_H| = 3q + 9|C| = 1.K_1 + 2.K_2 + 3.K_3$  and  $g(\mathcal{V}) = 0.K_1 + 1.K_2 + 2.K_3 < |V_H|$ . Thus, the only way the instance of RGPGM can be partitioned with gain  $g(\mathcal{V}) = 3q + 9|C|$  is when it can be partitioned into cliques of size 3 with all edge weights equal to 1. If  $V = V_1 \cup V_2 \cup \dots \cup V_{q'}$  is a partition of  $H$  into cliques of size 3 with edge weight 1, then the corresponding exact cover can be computed by choosing those subsets  $c_i \in C$  such that  $\{a_i^3, a_i^6, a_i^9\} = V_j$  for some  $j, 1 \leq j \leq q'$ . It can be easily be verified that the subsets chosen this way will constitute an exact cover for the X3C problem. ■

## V. OPTIMAL SOLUTION THROUGH MATHEMATICAL PROGRAMMING

In this section, we provide an Integer Linear Program formulation [1] for finding the gain maximizing partition of the path intersection graph  $H(V_H, E_H, G_H, L_H)$ . Let the network graph be  $G$  and  $\Lambda$  be the set of all wavelengths in the network.  $\Lambda_i$  is the set of wavelengths associated with vertex  $v_i \in V_H$  as specified by  $L_H$ .  $g(i, j)$  is the gain of edge  $(v_i, v_j) \in E_H$  as specified by  $G_H$ . Let  $|V_H| = n$ . Suppose the optimal partition of  $V_H$  is  $\mathcal{V}$  with components  $\{V_1, V_2, \dots\}$ . There can be at most  $n$  components in the optimal partition. The indicator variables are defined as follows.  $\forall v \in V_H, 1 \leq i \leq n, x_v^i = 1$  if vertex  $v \in V_i$ , 0 otherwise.  $\forall u, v \in V_H, 1 \leq i \leq n, y_{u,v}^i = 1$  if vertices  $u, v \in V_i$ , 0 otherwise.  $\forall v \in V_H, 1 \leq c \leq |\Lambda|, z_{v,c} = 1$  if vertex  $v$  is assigned wavelength  $c$ , 0 otherwise. The *objective* function is to maximize the total gain of the partition. That is, *Maximize*  $\sum_{i=1}^n \sum_{u=1}^n \sum_{v=1}^n g(u, v) y_{u,v}^i$ . The set of *constraints* is given below.

- Each vertex must be in exactly one component of the partition. That is,  $\forall v \in V_H, \sum_{i=1}^n x_v^i = 1$ .
- Each vertex must be assigned exactly one wavelength from its list of available wavelengths. That is,  $\forall v \in V_H, \sum_{c \in \Lambda_v} z_{v,c} = 1$  and  $\sum_{c \notin \Lambda_v(v)} z_{v,c} = 0$ .
- An edge  $(u, v)$  is in component  $i$  if and only if both  $u$  and  $v$  are in  $i$ . Mathematically,  $y_{u,v}^i = x_u^i x_v^i$ . This quadratic constraint can be replaced by two linear constraints.  $\forall u, v \in V_H, 1 \leq i \leq n, x_u^i + x_v^i \geq 2y_{u,v}^i$  and  $x_u^i + x_v^i - 1 \leq y_{u,v}^i$ .
- All vertices belonging to a component must be assigned the same wavelength. Define for convenience, an indicator variable for each vertex pair  $u, v$  and wavelength  $c$  as follows.  $\forall u, v \in V_H, 1 \leq c \leq |\Lambda|, s_{u,v}^c = 1$  if vertices  $u, v$  are both assigned the same wavelength  $c$ , 0 otherwise. The quadratic constraint  $s_{u,v}^c = z_{u,c} z_{v,c}$  can be replaced by the two linear constraints:  $\forall u, v \in V_H, 1 \leq c \leq |\Lambda|, z_{u,c} + z_{v,c} \geq 2s_{u,v}^c$  and  $z_{u,c} + z_{v,c} - 1 \leq s_{u,v}^c$ . Now, the constraint that all vertices in a component must be assigned the same wavelength can be represented as:  $\forall u, v \in V_H, 1 \leq i \leq n, y_{u,v}^i \leq \sum_{c=1}^l s_{u,v}^c$ .
- Two backup paths whose combination causes divergence cannot be assigned the same wavelength. That is,  $\forall u, v \in V_H$  such that  $g(u, v) = -\infty$  &  $B_u, B_v$  are not disjoint in  $G, \sum_{c=1}^l s_{u,v}^c = 0$ .

## VI. HEURISTIC FOR COMPUTING GSTREAMS

**Algorithm 1** heuristic\_streams( $H(V_H, E_H, G_H, L_H)$ , set  $B_i$  of backup paths in  $G$ )

---

```

1:  $k \leftarrow 1$ 
2: while  $\exists$  edge  $(v_i, v_j) \in E_H$  such that  $g(i, j) \geq 0$ , do
3:    $(v_a, v_b) \leftarrow (v_i, v_j)$  such that  $\Lambda_i \cap \Lambda_j \neq \emptyset$  and  $g(i, j)$  is largest
     among all  $(v_i, v_j) \in E_H$ 
4:    $V_k \leftarrow \{v_a, v_b\}$ 
5:   Select vertex  $v_t \in V_H \setminus V_k$  such that  $\sum_{v_s \in V_k} g(s, t)$  is
     positive, maximum and  $\Lambda(V_k) \cap \Lambda_t \neq \emptyset$ . If no such vertex
     exists, go to step 8.
6:    $V_k \leftarrow V_k \cup \{v_t\}$ .
7:   Remove vertex  $v_t$  and all incident edges from  $H$ . Go to step
     5.
8:   Select wavelength  $\lambda$  from  $\Lambda(V_k)$  uniformly at random and
     assign it to all vertices of  $V_k$ .
9:    $\forall v_i \in V_H \setminus V_k$  and  $\exists v_j \in V_k$  such that  $B_i \cap B_j \neq \emptyset$ ,  $\Lambda_i \leftarrow$ 
      $\Lambda_i \setminus \lambda$ 
10:   $k \leftarrow k + 1$ 
11: end while
12: if  $V_H \neq \emptyset$  then
13:   while  $V_H \neq \emptyset$  do
14:    Select at random, a vertex  $v_i \in V_H$  into component  $V_k$ .
15:    Select wavelength  $\lambda$  from  $\Lambda_i$  uniformly at random and
     assign it to  $v$ .
16:     $\forall v_j \in V_H \setminus v_i$  such that  $B_i \cap B_j \neq \emptyset$ ,  $\Lambda_i \leftarrow \Lambda_i \setminus \lambda$ 
17:     $V_H \leftarrow V_H \setminus v_i$ ,  $k \leftarrow k + 1$ 
18:   end while
19: end if
20: return  $\{V_1, V_2, \dots, V_{k-1}\}$ 

```

---

In this section, we present the heuristic to compute gStreams from the Path Intersection Graph  $H = (V_H, E_H, G_H, L_H)$ . Recall that  $\Lambda_i \in L_H$  represents the set of wavelengths available for vertex  $v_i \in V_H$ . Let  $\Lambda(S)$  for  $S \subseteq V_H$  be defined as the set of wavelengths common to all vertices of  $S$ . That is,  $\Lambda(S) = \bigcap_{v_i \in S} \Lambda_i$ . The heuristic first selects two vertices  $v_a$  and  $v_b$  into a component such that edge  $(v_a, v_b)$  has the largest gain in the path intersection graph and  $v_a, v_b$  have at least one available wavelength in common. The heuristic proceeds by selecting vertices from the rest of the graph that have at least one wavelength in common with the vertices in the component and result in the largest component gain. When no such vertex can be found, this component is complete and the heuristic assigns a wavelength randomly from the list of wavelengths common to all vertices in the component. These vertices (and the incident edges) are then removed from the graph. The wavelength that was assigned (if present) is removed from the set of available wavelengths of those vertices in the graph whose backup path in the network graph shares edges with the backup paths corresponding to the vertices in the component. This process is repeated until there are no edges with positive gain left in the graph. When the graph has no positive gain edges, the heuristic partitions the remaining vertices in the graph into separate partitions and assigns wavelengths randomly from their list of available wavelengths. The heuristic runs in  $O(n^2l)$  time, where  $n$  is the number of backup paths and  $l$  is the number of wavelengths.

## VII. SIMULATION ENVIRONMENT, RESULTS AND DISCUSSION

We conducted simulations to evaluate the gain for stream and gStream partition. In both cases, we compared the optimal gain value obtained by solving the Integer Linear Program with that of the heuristic. The experiments were conducted on two network graphs namely, National network (24 nodes, 44 links) and NSFNET (14 nodes, 21 links). The experiments were conducted for different number of wavelengths ( $\lambda$ ). For each network and wavelength,  $n$  source-destination ( $s-d$ ) pairs were randomly selected. The shortest path between each ( $s-d$ ) pair was chosen to be the primary path between ( $s-d$ ). A randomly selected wavelength that can be used throughout on the path was assigned to all the nodes on the path. For the backup path between ( $s-d$ ), the first 5-shortest paths that are disjoint with the primary path were computed. One of these paths was selected uniformly at random to be the backup path between ( $s-d$ ). From the set of backup paths, compatibility and gain values between the paths for forming streams and gStreams were computed. The algorithm to compute the gain value that can be obtained by the concatenation two backup paths is provided in the appendix.

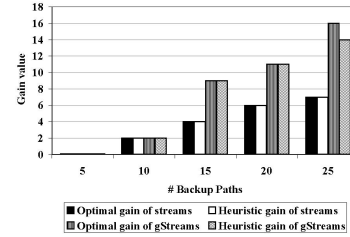


Fig. 4. Comparison of gain of streams and gStreams for National network with  $\lambda = 13$

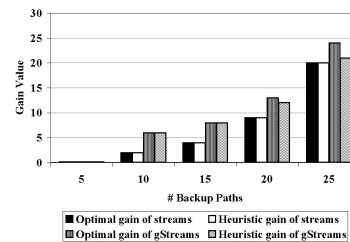


Fig. 5. Comparison of gain of streams and gStreams for NSFNET network with  $\lambda = 13$

All the simulations were conducted on a Pentium-4 3.2 GHz computer with 2 GB RAM. The ILPs were solved optimally using ILOG CPLEX 10.0. In the experiments of Figures 4 and 5, we set out to study the effect of increasing the number of backup paths on the gain value in case of streams and gStreams in National and NSFNET networks respectively. We varied the number of backup paths from 5 to 25, while the total number of wavelengths in the network was 13. In both the networks, CPLEX took less than 30 minutes to solve for 20 or

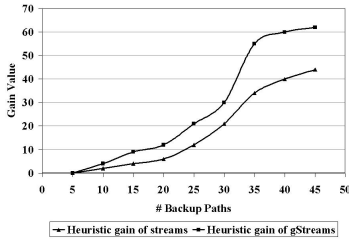


Fig. 6. Comparison of gain of streams and gStreams obtained from heuristic for large number of backup paths for National network with  $\lambda = 15$

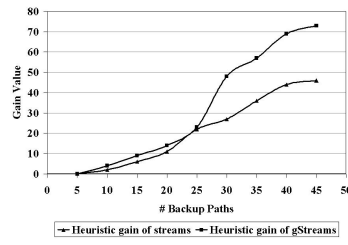


Fig. 7. Comparison of gain of streams and gStreams obtained from heuristic for large number of backup paths for NSFNET network with  $\lambda = 15$

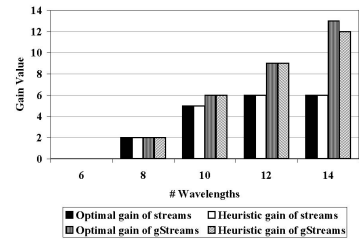


Fig. 8. Effect of increasing the number of wavelengths on the gain for streams and gStreams for National network with 20 backup paths

less backup paths, while for 25 backup paths, it typically took 17 hours to compute the optimal solution. The time taken by heuristic for all the cases was in the order of a few seconds. In 88% of the test cases, the gain value obtained by the heuristic differed with the optimal gain value by at most 1. This strongly suggests that our heuristic provides near optimal solution to most instances of the problem in a fraction of the time needed for finding the optimal solution. In the experiments of Figures 6 and 7, we increased the number of backup paths to 45. Because of the large execution times for solving the ILP in each instance, we resorted to the heuristic. We varied the number of backup paths from 5 to 50 and computed the heuristic gain values of streams and gStreams. It can be seen that the gain value of gStreams becomes significantly higher than that of streams as the number of backup paths increase. This is because, gStream with its OD1G structure can take better advantage of more number of backup paths than streams which is made up of only directed paths. In figure 8, we studied the effect of increasing the number of wavelengths in the network on the gain value of streams and gStreams. When more number of wavelengths are available, gStreams gain is significantly higher than streams gain since increase in number of wavelengths relaxes the wavelength continuity constraint and allows more gStream formations. From all these experiments, the superior benefits of gStreams are evident.

## VIII. CONCLUSION

In this paper, we presented a generalized notion of streams and developed efficient algorithms for maximizing capacity utilization. Through our extensive simulation, we demonstrated that the gStream scheme provides significantly better resource utilization compared to streams.

## REFERENCES

- [1] M. Bazaraa, J. Jarvis, and H. Sherali, "Linear Programming and Network Flows (3rd Edition)", John Wiley & Sons, 2005.
- [2] M. R. Garey and D. S. Johnson, "Computers and Intractability: An Introduction to the Theory of NP-Completeness", W. H. Freeman, 1979.
- [3] O. Gerstel and R. Ramaswami, "Optical layer survivability: A services perspective", *IEEE Communications Magazine*, vol. 38, no. 3, March 2000.
- [4] W. D. Grover and D. Stamatelakis, "Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration", *IEEE ICC*, Atlanta, 1998.
- [5] N.-F. Huang, and T.-H. Huang, "On the Complexity of Some Arboriscences Finding Problems on a Multihop Radio Network", *BIT*, vol. 29, no. 2, 1989.

- [6] S. Kim and S. S. Lumeta, "Capacity-efficient protection with fast recovery in optically transparent mesh networks", *Proceedings of Broadnets*, 2004.
- [7] M. Medard, S. Finn and R. A. Barry, "WDM Loop-back Recovery in Mesh Networks", *Infocom*, New York, March 1999.
- [8] G. Mohan, C. S. R. Murthy and A. K. Somani, "Efficient algorithms for routing dependable connections in WDM optical networks", *IEEE/ACM transactions on Networking*, vol. 9, no. 5, October 2001.
- [9] S. Ramamurthy and B. Mukherjee, "Survivable WDM Mesh Networks", *Parts I and II Proc. of IEEE INFOCOM*, March 1999, and *ICC*, Vancouver, CA, June 1999.
- [10] A. Sen, B. Hao, B. H. Shen and G. Lin, "Survivable Routing in WDM Networks: Logical Ring in arbitrary Physical Topology", *IEEE ICC*, New York, May 2002.

## APPENDIX: GAIN COMPUTATION

### Algorithm 2 compute\_gain( $B_i, B_j$ )

---

```

1: Initialize gain ← 0,  $E = \emptyset$ ,  $\delta_i^- \leftarrow 0$ ,  $\delta_i^+ \leftarrow 0$ ,  $1 \leq i \leq n_b$ 
2: for all edges  $(u, v) \in B_i$  do
3:    $E \leftarrow E \cup \{(u, v)\}$ 
4:    $\delta_u^+ \leftarrow 1$ ,  $\delta_v^- \leftarrow 1$ 
5: end for
6: for all edges  $(u, v) \in B_j$  do
7:   if  $(u, v) \in E$  then
8:     gain ← gain + 1
9:   else
10:     $E \leftarrow E \cup \{(u, v)\}$ 
11:   end if
12:    $\delta_u^+ \leftarrow \delta_u^+ + 1$ ,  $\delta_v^- \leftarrow \delta_v^- + 1$ 
13:   if  $\delta_u^+ > 1$  or  $\delta_v^- > 1$  then
14:     return  $-\infty$ 
15:   end if
16: end for
17: return gain

```

---

The function `compute_gain` takes as arguments two backup paths  $B_i$  and  $B_j$  and determines the gain value that can be obtained by including  $B_i$  and  $B_j$  in a stream. If  $B_i$  and  $B_j$  are incompatible, then this function returns  $-\infty$ . Let  $n_b$  represent the number of vertices in  $B_i \cup B_j$ . Let  $E$  be the set of unique edges in  $B_i \cup B_j$ . Let  $\delta_v^-$  and  $\delta_v^+$  represent the indegree and outdegree of vertex  $v$  respectively in  $B_i \cup B_j$ . Computing the gain value for gStreams requires only a small modification to the above procedure. Recall that in the case of gStreams, vertices with indegree  $> 1$  are possible. Hence for gStreams, we need not compute and check  $\delta_v^-$  in the above procedure.