Dynamic Connection Establishment and Network Re-optimization in Flexible Optical Networks

P. Soumplis, K. Christodoulopoulos, E. Varvarigos
Department of Computer Engineering and Informatics, University of Patras, Greece and
Computer Technology Institute and Press – Diophantus, Patra, Greece
Emails: {soumplis,kchristodou,manos}@ceid.upatras.gr

***Abstract*—We propose a dynamic connection establishment algorithm for flexible optical networks. When the spectrum is fragmented, blocking a connection establishment, the algorithm re-optimizes the network by shifting ("pushing") in the spectrum domain and/or rerouting established connections. We devised appropriate algorithms that search among different combinations of shifting and rerouting alternatives and select the one with the minimum cost. Since shifting or rerouting a connection might trigger more shifting and rerouting actions, the proposed algorithms are recursive. Since the solution space can be very large, we use a threshold on the recursion depth to reduce the complexity, and also provide a tradeoff between performance and running time.**

***Index Terms*—Flexible/Elastic optical networks, dynamic Routing and Spectrum Allocation, spectrum defragmentation.**

I. INTRODUCTION

Flexible optical networks (the term elastic is also widely used) are regarded as the most promising architecture for next generation backbone and metro area networks, since their increased spectral efficiency and adaptability is considered suitable for future requirements. These networks are based on the flex-grid technology where the spectrum is divided into 12.5 GHz spectrum slots, a smaller granularity than in traditional WDM networks. Moreover, the slots can be combined to create channels that are as wide as needed. With the Bandwidth Variable Transponders (BVT) that can adapt their transmission parameters, flexible networks become more dynamic, adaptive and efficient than traditional solutions [1].

During the operation of an optical network, new connections are established and torn down dynamically with time. In contrast to traditional WDM networks, where bandwidth assignment is uniform, in flexible networks the spectrum eventually becomes fragmented, a problem that becomes more severe as time progresses. Thus, after a point the available spectrum is inefficiently utilized, the network serves fewer demands than one would expect at its actual load level, and connections are blocked even though there are enough resources on the links that could be used to serve them.

To address this problem, two types of defragmentation methods have appeared: proactive and reactive methods. Reactive defragmentation is triggered when a new demand cannot be served, while proactive defragmentation is performed in a periodic or in an event-driven manner without being triggered by connection blocking (e.g. at connection release). The latter method aims at maintaining the network in a good shape without a priori knowing if the changes made will be needed or not. The former method is triggered only when needed, meaning that spectrum fragmentation has reached a critical point, and thus it has to be fast and efficient. Rearranging as few connections as possible and achieving low running time and little disruption in the network are the objectives that matter most in this case.

In this paper we propose a Dynamic RSA (D-RSA) algorithm for establishing transparent (without regenerators) and translucent (with regenerators) connections in a flexible optical network and reactively defragment it when deemed appropriate. We assume an optical network that encompasses slotted flex-grid and tunable BVT transponders, and a generic traffic scenario where demands arrive dynamically, each requesting a specific rate between a specific source and destination. Serving this demand requires the establishment of one or several connections, depending on the requested rate, the distance between the end-points and the capabilities of the transponders. So the D-RSA algorithm has to decide on how to break the requested connection demand into connections (if useful), determine where to use regenerators (if needed and if provisioned in the network), and allocate path(s) and spectrum to the connection(s). If the required resources are free the demand is served and the connection(s) is (are) established; otherwise, we use two defragmentation techniques to reactively re-optimize the network and serve the demand: (i) the push-pull technique, where established connections continue using the same path but are shifted in the spectrum domain without tearing them down, and (ii) the rerouting technique, where established connections are torn down and are routed over the same or different paths in a make-before-break manner. The two techniques can also be used jointly to achieve even better performance.

The long term goal of the D-RSA algorithm is to serve the demands so as to minimize network blocking. If a demand cannot be served at the current network configuration state, the algorithm applies the two aforementioned techniques to defragment the spectrum. A secondary objective while doing so is to affect as little as possible the current state of the network, that is, to minimize the disruptions/changes in existing connections. We define the re-optimization cost in terms of the spectrum slots that are shifted by the push-pull technique, or in terms of the number of rerouted connections for the rerouting technique. We devised appropriate algorithms that search among different combinations of shifting and rerouting alternatives and select the one with the minimum cost. Since shifting or rerouting a connection might in turn trigger more shifting and rerouting actions, the proposed algorithms are recursive, and the solution space can be enormous. To reduce the running time we used a threshold on the recursion depth. Our simulation results show that the blocking probability can be substantially reduced using the proposed techniques and that the selection of the recursion threshold is a good way to obtain a trade-off between the performance and the running time.

II. RELATED WORK

Dynamic flexible optical networks have received increased recent attention, with much of the research effort focusing on algorithms to cope with connection establishment and spectrum fragmentation [2]. One way to reduce spectrum fragmentation, is to re-optimize the network by rerouting (tearing down and re-establishing) existing connections. A second and often better approach is to use the finer granularity and channel adaptability offered by the advanced transponders envisioned in order to shift connections in the spectrum domain without interruption. This is achieved in [3] by the so called *push-pull* technique.

Irrespectively of the method they use (rerouting or spectrum shifting), the corresponding defragmentation algorithms can be divided, as mentioned before, into reactive [8][9] and proactive [4] [5][6][7] algorithms, according to whether they are triggered or not by a blocking/critical event.

The authors in [8] defragment the spectrum by rerouting existing connections so as to pack them into the lower spectrum slots, while also minimizing connection interruptions. In [9] different techniques for spectrum sharing between neighbouring connections are introduced to serve time varying traffic. In [5] the authors propose an algorithm that is invoked when a connection or a group of connections are torn down, by rerouting associated remaining connections at lower spectrum bands. In [6] the authors examine defragmentation in practice by rearranging connections spectrally while also considering the advantages obtained by different channel spacing selections. In [7] two defragmentation methods are proposed, with the first focusing on the most congested links, and the second performing network-wide proactive defragmentation by rerouting connections so as to pack them in a most-used spectrum slot assignment manner. A reactive defragmentation method is proposed in [8], where blocking triggers the rerouting of existing connections to make space for the new connection.

The novelty of our proposed solutions compared to previous works is fourfold. First, we provide general algorithms that take generic parameters as input. In particular, the input comes in the form of feasible transmission configurations of the transponders used in the network, which incorporate the physical layer impairments. Second, previous work focused on transparent networks, without, to the best of our knowledge, considering regenerators. Our algorithms take into account regenerators that can be used to achieve higher spectrum efficiency, lower cost and reduce blocking probability. Third, previous works perform defragmentation using either push-pull or rerouting, while in our work we also consider combining these techniques to achieve even better performance. Finally, we explore a wider defragmentation space than former approaches, by examining all possible combinations to make the required space and use a threshold on the recursion depth to control the complexity and consequently the running time of our algorithms.

III. PROBLEM DESCRIPTION

We are given an optical network $G = (V,E)$, where V denotes the set of nodes and E denotes the set of single-fiber links. Each link $l \in E$ is characterized by its length D_l, The spectrum is divided in spectrum slots of F GHz, where one spectrum slot corresponds to the switching granularity of the flexible network elements (flex-grid switches and bandwidth variable transponders - BVTs). The network support F_t number of slots.

The spectrum utilization of a link l is represented by a three state vector U_l, called the link slot utilization vector, of length equal to F_t, and U_{li} represents the i-th slot. A spectrum slot can be in one of the following states: (i) free (denoted by state u_f), (ii) used for data transmission (denoted by u_d), or (iii) used as guardband (denoted by u_g). The rules are that data slots cannot be used by new connections, free slots can be used for data, while free and guardband slots can be used for guardband by new connections. The slot utilization vector U_p of a path p can be computed using an (associative) 3-ary operator $\oplus$ for combining ("adding") the spectrum slots of the links that comprise it. The combining operator is defined as follows:

$u_f \oplus u_d = u_d$, $u_f \oplus u_g = u_g$, $u_f \oplus u_f = u_f$, $u_g \oplus u_g = u_g$, $u_d \oplus u_d = u_d$, $u_d \oplus u_g = u_d$

Thus, $U_{pi} = \bigoplus_{l \in p} U_{li}$, for all $i=1,2,\ldots,F_t$.

The traffic is served by BVTs that control (a) the modulation format and (b) the spectrum (in the form of contiguous spectrum slots) they utilize. By adapting these features, a BVT of cost c can be tuned to transmit r Gbps using bandwidth of b spectrum slots and a guardband of g spectrum slots from the adjacent spectrum connections to reach l km distance with acceptable quality of transmission (QoT). More formally, a specific transponder of cost (type) c is characterized by its physical feasibility function f_c that gives the reach $l=f_c(r,b,g)$ at which it can transmit with acceptable QoT as a function of the parameters r (rate), b (spectrum), and g (guardband) that we can control. This function captures the physical layer impairments, assuming worst-case contribution for the interference-related impairments (four-wave-mixing, cross-phase modulation, cross-talk), and can be obtained either through experiments or using analytical models [8][11].

Using function f_c we define (*reach-rate-spectrum-guardband-cost*) transmission *tuples*, $t=(l_t,r_t,b_t,g_t,,c_t)$, which correspond to *feasible* transmission configurations. The term "feasible" is used to signify that the tuple definition incorporates the limitations posed by physical layer impairments. The transponders have certain limitations in their capabilities, which are of the following forms: the maximum symbols per second (baud rate), and/or the maximum modulation format, and/or the maximum spectrum used, and/or the maximum transmission rate. Given the transponders' limitations, and since the modulation format and the spectrum are selected from discrete sets, we obtain the set of feasible transmission configurations for the transponders.

We assume that demands arrive at random time instants and are immediately served by the Dynamic Routing and Spectrum Allocation (D-RSA) algorithm. An incoming demand is characterized by its source-destination pair (s,d), its requested capacity D, and a mode indicator M that states if the demand is allowed or not to use regenerators (translucent or transparent, respectively). This definition is quite generic and can capture traffic serving in both transparent and translucent networks, but also allows for extra functionality as will be discussed shortly. If the demand cannot be satisfied in the current state of the network, the network is re-optimized and thus the problem can be viewed as an extension of the RSA and thus is NP-hard. In this paper we propose a heuristic D-RSA algorithm to provide solutions for large problem instances, with an exact algorithm been also developed but not reported here due to space limitations.

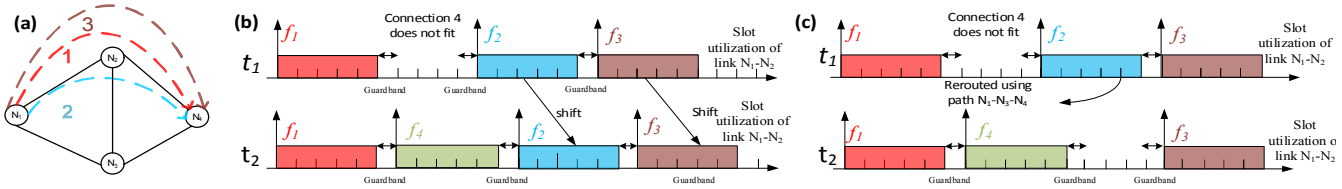

Fig. 1: (a) A flexgrid network where 3 connections exists at time t_1. (b) The spectrum slot allocation on link N_1-N_2 at two different time instants t_1 and t_2. Initially at time t_1, connections 1, 2 and 3 exist and a new connection 4 arrives. Not finding sufficient space to serve it, we shift (push-pull) connections 2 and 3 by two slots each, and the new connection 4 is established at time t_2. (c) Same scenario but using the rerouting technique.

Connection establishment in a network where the use of regenerators is not provisioned can be performed by requiring all demands to be served transparently. In a translucent network, if the mode indicator of a demand allows the use of regenerators, it is the up to the D-RSA algorithm to decide whether to use them or not. However, the extra functionality comes when a demand in a translucent network explicitly requires transparent service (e.g. for reduced cost) and this is performed by the proposed D-RSA by providing an appropriate non-regenerated solution.

To serve a connection requesting rate D, the D-RSA establishes one or more (in case rate D is not supported at the respective distance by a single transponder) transparent or translucent connections from s to d. Note that translucent connections consist of transparent sub-connections (sub-paths) regenerated at intermediate points, and thus the transparent case can be viewed as a special case of the translucent case. In the remainder of this paper we will avoid specifying the mode (transparent or translucent) of a connection unless this is not apparent from the context.

The D-RSA algorithm examines a number of candidate paths between the source and destination, and depending on their link lengths, it finds the transmission tuples that can be used over each of them, what we call a feasible path-tuple pair. For a given demand some path-tuples will never be used (said to be "dominated" by others), since they require more spectrum and transponders. These are removed, limiting the search space without losing good solutions.

For each non-dominated path-tuple pair, since the requested rate D can be higher than the maximum transmission rate at the corresponding distance, the algorithm calculates the number of connections to be established, the nodes where regenerators will be placed, if needed and allowed, and the amount of spectrum to be used by each connection. Then it searches to allocate contiguous spectrum slots to these connection(s). Note that each path-tuple corresponds to a specific number of connections and regenerators, and by examining all the path-tuple pairs we examine all feasible combinations of these parameters. For example, we examine if a single high-rate short-reach connection requiring many regenerators at intermediate nodes should be used, or if the other extreme that uses many low-rate long-reach connections should be used, or if some other option in-between should be used.

If the algorithm finds sufficient available spectrum slots over a specific path-tuple pair, the connection(s) defined by the path-tuple pair choice is (are) established, subject to the RSA limitations. If the number of free spectrum slots is insufficient for at least one connection, the (i) push-pull or (ii) rerouting, or a combination of these techniques is used to reactively re-optimize the network, free spectrum and serve that connection.

The push-pull technique offers the ability to shift an existing connection's spectrum band with no service disruption (hitless) and can be used to defragment the network [3]. Fig. 1 presents an example where the push-pull technique is used to establish a connection that would otherwise be blocked. The time to push-pull an existing connection is proportional to the number of slots by which the connection is shifted, and experiments show that this can be done quite fast. Shifting more than one connection that are adjacent towards the same direction can be done in parallel, so that the overall reconfiguration time is the time of the longest shifting.

The push-pull algorithm we implemented selects a spectrum void and creates the required space to establish the blocked connection by shifting the void's neighbouring connections. The cost of using this particular void is defined as the spectrum shifting (in slots) of the connection that is shifted the longest, since this determines the time for performing the push-pull(s). Other important metrics that can be optimized are the number of shifted connections, or the total number of slots of all connections that are shifted, which indicate the network state changes and the complexity of the re-optimization.

When the establishment of a connection causes the shifting of other connections, the shifting can be done towards the upper and bottom direction by certain slots, until we create the required spectrum space. Different selections of the slots that are freed in each direction result in different solutions and costs. To achieve the lowest possible cost, the algorithm considers all possible combinations of slots freed in the two directions.

The rerouting algorithm we implemented is based on Make-before-Break (MbB) technique and utilizes additional transponders (and regenerators, when used) to re-establish an existing connection before tearing it down. In this technique, before rerouting an existing connection we re-establish it over the same or a different path. Since we can have different lengths, the use of a different modulation format with higher transmission reach may be required to serve the demand. In some cases we may have to utilize more than one transponders to make the transmission over the new path feasible. The old and the new connection(s) will both be active for a small interval of time, while the traffic is switched from the old to the new one. Finally, the old connection is torn-down releasing the spectrum that was reserved for it. Similarly to the push-pull technique, when more than one connections are rerouted we assume that this is done in a pipelined manner, minimizing the number of spare transponders/regenerators required. The MbB technique guarantees small traffic disruption (this depends on the difference in the lengths of the two paths used and would be seamless if we reroute the connection over the same path). Compared to the push-pull, when rerouting a connection we may end up installing more than one connections and use more regenerators than before. The downside of the push-pull

technique is the inability to change the path of an already established connection, which may be necessary in some cases, and is achieved using the rerouting technique.

The rerouting algorithm we devised reroutes upper and lower neighbouring connections from the selected void so as to free the required spectrum space. It examines all possible combinations of reroutings, similarly to the push-pull algorithm, and selects the combination that minimizes the cost, considered to be the number of rerouted connections.

The algorithm that combines the two aforementioned techniques examines cases where some connections are rerouted and others are push-pulled to create the appropriate space, selecting the combination that achieves the lowest cost. Although our algorithm is parametric to the cost of these two techniques, we assume that reroutings have higher cost due to the need for spare transponders and the disruption of service that might be experienced in certain cases.

IV. D-RSA Algorithm description

In this section we present the Dynamic Routing and Spectrum Allocation (D-RSA) algorithm for operating flexible networks. The algorithm takes as input the feasible transmission options described by the (reach-rate-spectrum-guardband-cost) tuples, the number of candidate paths k to be checked for each demand and a threshold H on the depth of neighbouring connections that can be pushed or rerouted. It gets invoked every time a new demand described by (s,d,D,M) arrives, where (s,d) are the source and destination, D is the demanded capacity in Gbps, and M is the mode indicator that specifies whether the demand requests to be served transparently or not. We will describe the algorithm considering translucent connections and the use of regenerators, since it is more generic and captures the only-transparent setting, indicating the differences between the two cases when deemed necessary.

In order to reduce the time required to serve a demand, for every (s,d) pair we pre-calculate the set P_{sd} of k paths, using a variation of the k-shortest path algorithm. These paths are the candidate paths that will be used by each demand. The main advantage of this technique is the small running time of the algorithm, which is critical for online and reactive algorithms.

For each pre-calculated path we identify the configurations (tuples) that can be used by the transponders over that path, based on the lengths of its links. In particular, we examine if a feasible transponder configuration tuple $t=(l_t,r_t,b_t,g_t,c_t)$, has transmission reach l_t higher than the length of the path p for the transparent case, or higher than the maximum link length of path p for the translucent case. A translucent (sub)connection is terminated at the regeneration node and a new (sub)connection is initiated, to create an end-to-end translucent connection. For a translucent network, for each acceptable path-transmission tuple pair (p,t), the path p is swept from left to right and a regenerator is placed whenever required, that is, at the last node before the transmission distance l_t of the tuple is reached. Thus, for path-transmission tuple pair (p,t) we find the set of nodes where regenerators have to be placed to make the transmission feasible. So for each (s,d) we have pre-calculated a set Q_{sd} of path-transmission tuple pairs (p,t) that are candidate solutions to serve demands between s and d.

When a new demand (s,d,D,M) is processed, we examine each path-transmission tuple pair $(p,t)\in Q_{sd}$ as a candidate option to serve it. For a specific path-tuple pair this translates to the establishment of one (or more, if the required capacity D is higher than the transmission rate of the specific tuple) connections. If a demand is broken into more than one connections, they all follow the same path when established.

To serve the demand with the specific path-tuple pair, the algorithm computes the path utilization vector based on the links that comprise the path. For each connection required to serve the demand with the specific path-tuple pair the algorithm checks if there are voids of spectrum able to serve that connection, taking also into account the guardband needed. If there are more than one voids, the algorithm selects the smallest one so as to leave bigger voids for future connections with higher spectrum needs. If there is no void to accommodate at least one connection of the path-tuple pair under consideration, we move to examine the next path-tuple pair. We stop the first time we are successful with a path-tuple pair, that is, we find appropriate voids to serve all connections required for that path-tuple pair selection. If we examine all candidate path-tuple pairs and none was successful we proceed with our network re-optimization techniques: push-pull and reroutings. Serving the demands with the available voids is considered zero cost and is preferred over the case where we have to re-optimize the network.

The above phase of the algorithm is described in Fig. 2(a). When we have ruled out the option of serving the demand without re-optimizing the network, we again start searching all candidate path-tuple pairs. For each path-tuple pair, for each of its connections that there is not enough spectrum the algorithm selects the biggest void and tries to create the extra slot space in one of the following ways.

Assume that we use the push-pull technique and start with a void that we need to expand by n slots to establish the connection that has guardband needs of gb_v spectrum slots. We can make this spectrum space by shifting connections that are upper or bottom adjacent to the void under examination. Let F_s and F_e be the first free slot and the last free slot of the void, respectively. For an upper connection i, we let F_i be its starting frequency and gb_i be its corresponding guardband needs.
Then we have to shift this connection by

$$n - Fi + F_e - \max(gb_v, gb_i)$$

For a bottom connection j, we again let F_j be its ending frequency and gb_j be its corresponding guardband needs. Then we have to shift this connection by

$$n - F_j + F_s - \max(gb_v, gb_j)$$

Shifting one connection may trigger the shifting of its adjacent connections. This is treated recursively by the same algorithm taking the shifted connection as the void. To make the required space there are n+1 combinations: shift the upper connections and make n slots space, or shift the upper to make n-1 slots and the bottom to make 1 slots space, ..., or shift only the bottom to make n slots space. We examine all different upper-bottom pushing combinations and calculate their cost; this is done quite fast as we only examine the two extreme cases and the costs of the other combinations can be calculated from them. The cost is defined as the number of spectrum slots of the connection that is shifted the longest, since this is proportional to the time required, but other interesting metrics could also be used, including the number of shifted connections or the number of shifted slots of all connections, etc. The algorithm stops execution when there are no other connections that need to be recursively pushed or when we reach the recursion

```
INPUTS: Network topology G=(V,E)
        Tuple lookup table
        W:Objective weight
        H:Recursion depth threshold
        Slot utilization vector of links
        New demand (s, d, D,M)

Calculate k paths

FOR each path-tuple pair
   Calculate number of
        connections,spectrum slots needed
   Calculate path slot utilization vector
   Find spetrum voids
   FOR each connection
      IF(size(void)>required_spectrum)
              Select best fit void
              Establish the connection
      ELSE
              Call defragmentation
      ENDIF
   ENDFOR
ENDFOR
```

```
Push pull defragmentation

FOR each blocked connection
  Select the biggest void
  Z←Number of slots to push the adjacent connections
  FOR ALL combinations of pushing upper by Z_u
        and lower by Z_l  Z=Z_u+Z_l
     Find up and down connections
     Push each connection
     WHILE (pushing feasible)
         WHILE (more connections to push)
            IF (Recursion-step≤H)
              Call push for the adjacent connection
              WHILE (More connections to push)
                 IF (no other combinations exist)
                    Select minimum cost combination
                 ENDIF
              ENDWHILE
            ENDIF
         ENDWHILE
     ENDWHILE
     Select combination with minimum cost
  ENDFOR
ENDFOR
```

```
Joint defragmentation

For each combination of reroutings
    Reroute the connection(s)
    Calculate free space
    IF (enough space)
       Calculate cost
    ELSE
       Call Push-pull defrag
    ENDIF
    Keep minimum cost combination
ENDFOR
```

Fig 2: (a). Pseudocode of the algorithm that served the demand by examining all combinations of path-tuple pairs establishing the connections in the existing spectrum voids. (b) Pseudocode of the push-pull algorithm. (c). Pseudocode of the joint (unified) algorithm.

threshold H in which case we consider that the connection is blocked for the specific upper-bottom pushing combination. If more than one combinations are feasible, we select the one that yields the smallest cost. If for the given path-tuple pair the void that we examine cannot be expanded, we move to examine the remaining path-tuple pairs that require less spectrum slots. If we examine all path-tuple pair and none is successful, the demand is blocked.

Since shifting an adjacent connection may trigger the shift of its own adjacent connections and so on, this can go deep and we can end up by examining (and making) a huge number of changes in the network, the whole network configuration in the worst case scenario.

To avoid the high running time of the algorithm, we use the threshold H to control the recursion depth. The depth of the push-pull algorithm is defined as follows. The initial void has depth zero, and every connection that is shifted inherits the depth level from the connection that shifts it and adds one. Although this limitation increases the blocking probability, it has the advantage of lower running times and can be used in cases when this is a critical parameter. The push-pull algorithm is described in Fig. 2(b).

The second way to create the required spectrum space needed to establish the new connection is by rerouting one or more connections. The key difference is that in this case we re-establish neighbouring connections, and we take them to remote spectrum blocks or over different paths, instead of shifting them. We again start with a void that we need to expand by n slots to establish the connection. We calculate the set of neighbouring connections from each side of the void, denoted by AC' and BC', as follows: the set AC' contains connections that utilize at least one spectrum slot in the interval $[F_e, F_e+n]$, and the set BC' contains connections that utilize at least one spectrum slot in the interval $[F_s-n, F_s]$. We then try the different combinations of rerouting the connections in these sets. This is done as follows. We first reroute all connections from one side, e.g the upper side, so as to make spectrum space of n slots. Then we reroute the connections from the upper side so as to create space n-1 slots and reroute connections from bottom side to create space equal to 1 slot, and so on. When a connection cannot be rerouted (we cannot find spectrum to reroute it), the algorithm has to free the remaining slots from the other direction. If connections cannot be rerouted form both directions and the freed spectrum is less than n then connection establishment is blocked. We search all the different upper-bottom rerouting combinations, and calculate for each one its cost defined as the number of reroutings performed multiplied by the number of connections each rerouting consists of (higher than one in case of rerouting a translucent connection). Finally, the algorithm selects the combination with the minimum cost.

Note that when rerouting over a different path it is not granted that the transmission configuration used originally in the rerouted connection will be feasible (e.g. when the new path is longer). To address this we treat the rerouted connection in a manner similar to that of the new connection, which is established only if there is a void with the appropriate size. In extreme cases where candidate paths have significant difference in their lengths, more than one connection may be required to be established in order to replace the initial connection. Also in order to be consistent and have end-to-end control rerouting a translucent connection involves the rerouting of the whole connection instead of its transparent sub-connection that causes the blocking.

The aforementioned techniques can be combined and used together in a unified algorithm to achieve even better performance. The algorithm that combines these two techniques works as follows. After performing a rerouting, following the same direction of the rerouting, the push-pull algorithm is applied to the remaining connections. Then more connections are rerouted, and so on. The cost of the solution is the combined cost of the push-pull and rerouting operations. The joint algorithm is shown in Fig. 2(c).

The intuition behind the unified algorithm is that when one or more connections block the expansion of the void, because they cannot be shifted, rerouting may be feasible and vice versa, making feasible the establishment of connections where the two algorithms separately would fail.

V. Numerical Results

We implemented the proposed D-RSA algorithm in Matlab and used it to evaluate its performance and that of the proposed re-optimization techniques. All simulations were carried out on a desktop with Intel Core i3 processor-2.3GHz, 4GB RAM, and 64-bit operating system. We used the 30-node Telefonica network topology in our experiments [12]. Spectrum slots were taken to occupy F=12.5 GHz, while the network supports F_t=320 slots. We assumed the use of a single type of flexible OFDM transponder that transmits up to 400 Gbps. The (reach-rate-spectrum-guardband) tuples used as input to these experiments were obtained from studies on physical layer

impairments for optical OFDM networks [6]. Demands at each node are generated according to a Poisson process with arrival rate λ and an exponentially distributed duration with mean $1/\mu$=1 time unit and destination uniformly chosen among all nodes. The demanded rate is drawn from a uniform distribution on the close interval [0,400] Gbps, rounded with a 10 Gbps step. This traffic generation wasselected to cover scenarios where the network is quite dynamic and adaptable to edge traffic changes.

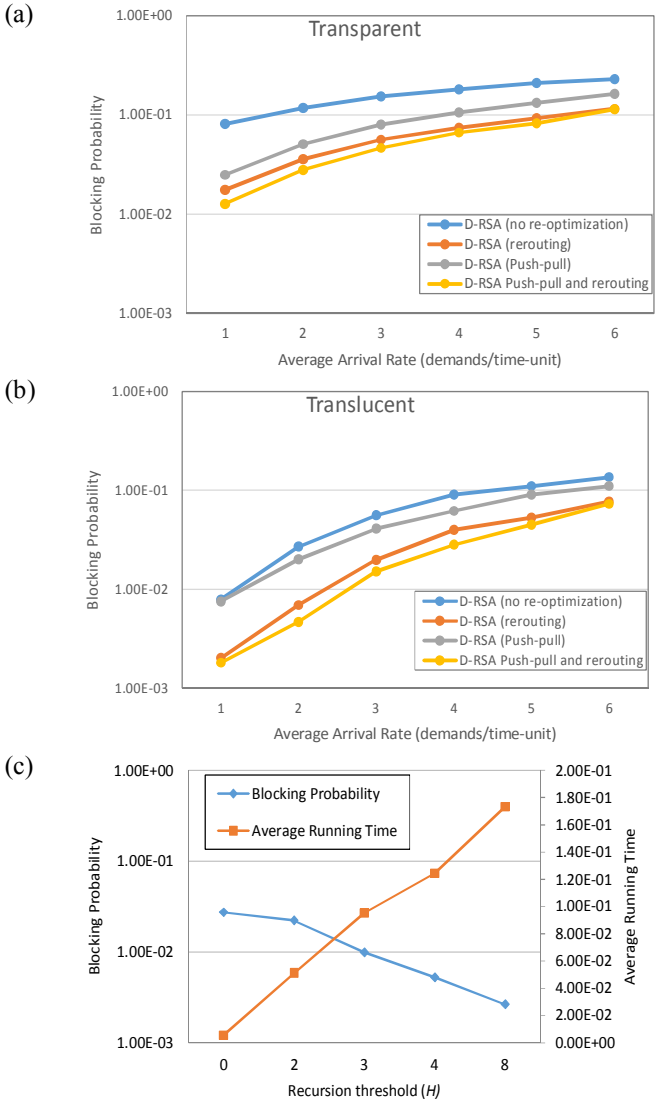

Fig 3: Blocking probability for the (a) transparent and (b) translucent case, (c) Blocking probability and average running time of the D-RSA/push-pull algorithm for λ= 4 demands/time unit as a function of the recursion threshold H.

Fig. 3(a) and (b) show the blocking probabilities of the D-RSA transparent and translucent algorithm, respectively, (i) without re-optimization, (ii) using the push-pull technique, (iii) using the rerouting technique and (iv) using both push-pull and rerouting (joint). For both network settings the D-RSA without re-optimization has the worst performance and is used as reference for the other solutions. Push-pull and rerouting techniques improve the performance, rerouting being better, since it exploits the solution space of different paths. Using both techniques in the joint algorithm improves the performance slightly more than the rerouting algorithm.

By comparing the results for transparent and translucent mode of operation we notice that in all cases the translucent mode achieves lower blocking probability. Since transparent connections are established according to the spectrum continuity constraint, the longer the connections the more difficult it is to find the required space (and the more spectrum it is required). In the case of translucent mode, the transparent sub-connections that are established are shorter and relax the spectrum continuity constraint at regeneration points. As a matter of fact, it is easier for the D-RSA algorithm to find continuous free slots to establish translucent connections, and re-optimize the translucent network.

Fig 3(c) show the effect of the recursion threshold H on the performance of the push-pull algorithm (note that up to now H was set to infinite). We see that lowering H increases the blocking probability but reduces the running time, trading-off these two metrics. Thus, H can be chosen so as to meet the response time requirements for serving demands.

VI. Conclusions

We proposed an algorithm for setting up connections and re-optimizing a flexible optical network. When a connection is blocked due to spectrum fragmentation the push-pull or/and rerouting techniques is used to make appropriate spectrum space with no service interruption. We devised appropriate algorithms that search among different combinations of shifting and rerouting alternatives. Our results show that we can substantially reduce blocking using the proposed algorithms and we can trade-off performance for running time by appropriate parameter selection.

Acknowledgements

This work has been partially funded by IDEALIST project.

References

[1] O. Gerstel, et al., “Elastic Optical Networking: A New Dawn for the Optical Layer?”, IEEE Communications Magazine, 50 (2), Feb, 2012.
[2] S. Gringeri, et al., “Flexible architectures for optical transport nodes and networks”, IEEE Communications Magazine, July 2010.
[3] Cugini, F.; et al.,P., "Push-Pull Defragmentation Without Traffic Disruption in Flexible Grid Optical Networks," Lightwave Technology, Journal of , vol.31, Jan.1, 2013.
[4] A.N. Patel,et al, "Defragmentation of transparent Flexible optical WDM (FWDM) networks",Optical Fiber Communication Conference, 2011.
[5] Wang, Xi; , et al "A hitless defragmentation method for self-optimizing flexible grid optical networks," (ECOC), 2012.
[6] Eira, A, et al., "Defragmentation of fixed/flexible grid optical networks," Future Network and Mobile Summit, 2013.
[7] Jie Luo, et al., "Partial defragmentation in flexible grid optical networks," Communications and Photonics Conference (ACP), 2012.
[8] Takagi, T, et al, "Disruption minimized spectrum defragmentation in elastic optical path networks that adopt distance adaptive modulation," Optical Communication (ECOC), 2011
[9] Velasco, L.; et al,"Elastic spectrum allocation for variable traffic in flexible-grid optical networks," Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012
[10] A. Klekamp,, et al “Limits of Spectral Efficiency and Transmission Reach of Optical-OFDM Superchannels for Adaptive Networks”, IEEE Photonics Technology Letters, 23 (20), 2011.
[11] R Borkowski, et al., “Experimental Study on OSNR Requirements for Spectrum-Flexible Optical Networks”, Journal on Optical Communications and Networking, 4 (11), 2012.
[12] Idealist deliverable: D1.1 - Elastic Optical Network Architecture: reference scenario, cost and planning.