Anycast Routing for Survivable Optical Grids: Scalable Solution Methods and the Impact of Relocation

Ali Shaikh*, Jens Buysse†, Brigitte Jaumard*, and Chris Develder†
* CSE Department, Concordia University, Montreal (QC) Canada
† Dept. of Information Technology – IBCN, Ghent University – IBBT, Ghent, Belgium

***Abstract*—In this paper, we address the issue of resiliency against single link network failures in optical grids and show how the anycast routing principle, which is typical of grids, can be exploited in providing efficient shared path protection.**

We investigate two different integer linear program (ILP) models for the full anycast routing problem, deciding on the primary and backup server locations as well as on the lightpaths toward them. The second model is a large scale optimization model which can be efficiently solved using column generation techniques. We also design two new heuristics: the first one is an improvement of a previously proposed one which, although providing near optimal solutions, lacks scalability, while the second one is highly scalable, at the expense of a reduced accuracy.

Numerical results are presented for three mesh networks with varying node degrees. They allow an illustration of the scalability of the newly proposed approaches. Apart from highlighting the difference in performance (i.e., scalability and optimality) among the algorithms, our case studies demonstrate the bandwidth savings that can be achieved by exploiting relocation rather than using a backup path to the original (failure-free) destination site. Numerical results for varying network topologies, as well as different number of server sites show that relocation allows bandwidth savings in the range of 7–21%.

I. Introduction

Challenging e-Science applications in different domains including high-level computing, parallel programming, fluid-dynamics, astrophysics and climate modeling have given rise to the idea of interconnecting geographically dispersed (high performance) computing sites in so-called grids. A grid is typically defined as a software and hardware infrastructure that provides access to high-end computational resources in a decentralized way, using general-purpose protocols and interfaces while providing a scalable quality of service (QoS) aware architecture [1].

Optical networks, which offer high bandwidth and low latency, are obviously prime candidates for interconnecting the various grid sites. This has given rise to the concept of so-called optical grids [2]. Given the typically high volume of data being processed, it is crucial for grids to be able to survive grid resource failures by providing resiliency mechanisms [2]. This holds for both optical networks and servers (storage and/or computational resources).

In this paper, we consider an optical circuit-switched network (such as an Automatically Switched Optical Network - ASON, see, e.g., [3]), based on Wavelength Division Multiplexing (WDM). We investigate on providing resiliency against network failures with the adoption of *shared path (SP) protection* under the anycast routing principle [4], for the grid to survive from any possible single link failure. We consider two variants of the SP protection schemes for which we provide a generic large scale optimization model, that we compare with the two proposed heuristics and a previously proposed classical integer linear program (ILP), which we adapted to the studied protection schemes.

In Classical Shared Path protection (CSP), a primary path connecting a given pair of source and destination nodes is protected by a link-disjoint backup path. The sharing offers the opportunity to limit the spare network resources, by allowing backup paths to reuse the same physical resources in case the corresponding primary paths are link disjoint. Note that such a scheme can be easily extended to node protection, by requiring node-disjoint paths, instead of link-disjoint paths.

In the context of the usual traffic in a WDM optical network, the node destination is given at the outset together with the description of the traffic requests to be provisioned. However, under the anycast principle [4], which is typical of grids, the destination is not necessarily given a priori. Hence, we will only assume the knowledge of the origin of the grid jobs and let the routing problem decide on an optimized choice of their destination (server location) site. The anycast routing principle even allows identifying a backup-site different from the one under failure-free conditions. This means that, instead of reserving a back-up path to the original destination determined by the grid scheduler, it could be of interest to relocate the job to another server in case of a link failure, assuming either tools for seamless transfer of running jobs [5], or that we deal with so-called Bag-of-Tasks (BoT) applications [6]. Under the assumption of BoT applications, the overall job requests can be broken up into smaller pieces, or at least into not overly time-critical ones. Consequently, we can assume it is acceptable to resubmit the jobs to the new location, and therefore do not need to account for migration of jobs running at a failed location to a new one. As demonstrated further in this paper, exploiting job relocations allows an overall reduction of (backup) network capacity and can be achieved by a Shared Path Protection with Relocation (SPR-A) scheme under Anycast routing principle. Hence, we will compare two cases, the first one when the backup location is identical to the primary server location (CSP-A scheme), and the second case where freedom is given to select a backup location which may be different from the primary one (SPR-A scheme).

In previous work, Buysse *et al.* [7], [8] demonstrated that such a relocation strategy can significantly decrease the number of network resources (number of bandwidth units) compared to its traditional counterpart, under the assumption of a fully specified traffic matrix, i.e. both job source and destination server nodes were known; relocation was then allowed to change the destination under link failures. Subsequently, in [9], the authors proposed both an ILP model and a heuristic for solving the relocation problem in the anycast case, while optimizing the selection of the destination server site for both working and backup paths.

The current paper significantly extends the aforementioned work with the following contributions: (i) a highly scalable column generation (CG) ILP model and solution; (ii) two new heuristics; (iii) an extensive comparison of the CG-ILP algorithm and of the heuristics, in terms of running times and optimality gaps; (iv) an investigation of the impact of the number of resources (server sites) where to execute the jobs; and (v) an assessment of the bandwidth relocation gains (compared to classical shared path protection) for varying topologies, in terms of average node degree (dense vs. sparse networks).

Note that, in the current work, we do not consider task flows of interdependent and possibly concurrent processes, which may thus run in parallel on potentially different locations (as, e.g., considered in [10]). Note that a computing site in our work is actually a complete data center (or cluster). In our work we focus on the wide area network interconnecting various data centers, not intra-data-center communication. Hence, we assume that grid jobs entirely run at a single such data center location. Extension of our work to multi-site traffic patterns will be considered in future work.

The remainder of this paper is structured as follows. In Section II, we give an overview of related work. In Section III, we detail the ILP models (previous and new column generation ones), and their solutions. Heuristics are discussed in Section IV. The comparative performances of the exact vs. heuristic solutions is dealt with in Section V. In the same section, throughout a case study, we also present the advantage of using relocation, as compared to classical shared path protection. Conclusions are drawn in Section VI.

II. Related Work

The problem addressed in this paper is a generalization of the classical Routing and Wavelength Assignment (RWA) problem in WDM networks. The vast research literature devoted to RWA focuses on finding a suitable routing path and wavelength, assuming both source and destination of connection requests are given (i.e., the *unicast routing* case). The most studied objectives are the minimum number of wavelengths (min-RWA) and the maximum grade of service, i.e., number of granted requests (max-RWA). For an extensive overview of such classical RWA literature, we refer to [11], [12] and more specifically to the Integer Linear Programming (ILP) approaches reviewed recently in [13]–[15].

As highlighted before, in this paper, we address the *anycast routing* case, where the problem is complicated by the fact that the destination is not known a priori, but can be freely chosen (among a given set of possible destinations, i.e., server sites). We consider the objective of minimizing the number of wavelengths summed over all network links, i.e., the number of bandwidth units. We consider here an off-line network design problem, aiming to decide on the network and server resource dimensions. Note that we will assume a given set of server sites as destinations; to select them, the approach discussed in [16] can be used. The related problem of accepting arriving connection requests in an on-line fashion (on a given, capacitated network instance), such as considered in [17], is out of the scope of this paper.

ILP formulations have been widely exploited in previous works in order to solve the RWA problem, as they provide a convenient way to flexibly and unambiguously define the problem and its instance-specific parameters: cost functions, wavelength conversion, protection scheme, etc. These ILP formulations typically fall into one of the following two categories: link or path based formulations, see, e.g., [13], [15] for a comparison of them. While some of these ILP formulations are more efficient than others, they all lack scalability when it comes to solve large instances, whether it consists of larger networks or larger traffic data sets. In order to overcome the scalability issues, large scale optimization models need to be devised such as the column generation model of [14]. Therein, the RWA problem is decomposed according to a set of configurations, where a configuration is added only if it contributes to the improvement of the current value of the objective.

While the works described above only have relevance for network primary provisioning, their formulations typically only require a few modifications to cover network protection cases. The works of Stidsen *et al.* [18] and Koster *et al.* [19] provide joint optimization of working and protection paths with the classical path protection scheme.

III. Proposed Solution Approaches

We aim to investigate two protection schemes, Classical Shared Path protection with Anycast (CSP-A) and Shared Path protection with Relocation and Anycast (SPR-A) from a network dimensioning perspective, i.e., we extend one step further the Classical Shared Path (CSP) and Shared Path with Relocation (SPR) models which were studied in [20].

We start from a demand vector expressing for every source of an optical grid network, the number of desired connections (i.e., job requests). It is up to the optimization model to choose which primary and backup server sites to use. For the CSP-A protection model, we impose the primary and the backup servers to be the same, while they can differ in the SPR-A model. Furthermore, we assume that every optical cross-connect (OXC) in the network is able to perform full wavelength conversion, which is sometimes referred to as the Virtual Wavelength Path (VWP) network [21]. Our network is modeled as follows:

G	$= (V, L)$, directed graph representing an optical grid, where V is the node set and L is the set of (directed) links, where we assume that every link has an unlimited transport capacity.
V	Node set, indexed by $v \in V$, representing the OXCs and possibly collocated server sites (computational and/or storage servers).
V_d	$\subset V$. Server node set, indexed by v or v_d, comprising the server sites (capable of processing grid jobs), i.e., potential candidate destinations.
L	Directional link set, indexed by ℓ. Each pair of connected nodes is usually connected by two links, one in each direction.

A. Standard ILP Model

For evaluation purposes, we briefly recall a first standard ILP model which was previously proposed in [8], [9]. We have simplified the notations of its first formulation and adapted it to the protection schemes studied in this paper, i.e., to the CSP-A and SPR-A protection schemes. Note that the first ILP was proposed to study the CSP scheme in which destination server nodes were given at the outset. Traffic instances are described by a set of requests, $k \in K$, where each request k originates at source node $v_s(k)$.

Variables of the first standard ILP model are as follows.

$p^{\text{W}}_{k\ell}$	$\in \{0,1\}$. $p^{\text{W}}_{k\ell}$ is equal to 1 if request k is routed (working path) through ℓ, 0 otherwise.
$p^{\text{B}}_{k\ell}$	$\in \{0,1\}$. $p^{\text{B}}_{k\ell}$ is equal to 1 if request k is routed (backup path) through ℓ, 0 otherwise.
d^{W}_{kv}	$\in \{0,1\}$. d^{W}_{kv} is equal to 1 if server site v is used as the primary server site for connection k. (Note that $d^{\text{W}}_{kv} = 0$ for $v \in V \setminus V_d$).
d^{B}_{kv}	$\in \{0,1\}$. d^{B}_{kv} is equal to 1 if server site v is used as the backup server site for connection k. (Note that $d^{\text{B}}_{kv} = 0$ for $v \in V \setminus V_d$).
b^{B}_{ℓ}	$\in \mathbb{Z}^+$. b^{B}_{ℓ} is equal to the number of shared backup bandwidth units on link ℓ.
$\delta_{k\ell\ell'}$	$\in \{0,1\}$. $\delta_{k\ell\ell'}$ is equal to 1 if and only if link ℓ' is used to protect link ℓ on the primary path of connection k.

The objective function aims at minimizing the overall network capacity, in terms of required working and backup bandwidth units on all links:

$$\min \quad \text{COST}_{\text{ILP}}(p^{\text{W}}, b^{\text{B}})$$

where

$$\text{COST}_{\text{ILP}}(p^{\text{W}}, b^{\text{B}}) = \sum_{\ell \in L} \left(b^{\text{B}}_{\ell} + \sum_{k \in K} p^{\text{W}}_{k\ell} \right). \tag{1}$$

We next describe the set of constraints. The first set of constraints defines the demand constraints and the flow conservation constraints for the primary paths (where $\omega^+(v)$ is the set of v's outgoing links, and $\omega^-(v)$ that of its incoming links):

$$\sum_{\ell \in \omega^+(v)} p^{\text{W}}_{k\ell} - \sum_{\ell \in \omega^-(v)} p^{\text{W}}_{k\ell} = \begin{cases} 1 & \text{if } v = v_k \\ -\,d^{\text{W}}_{kv} & \text{if } v \in V_d \\ 0 & \text{otherwise} \end{cases} \qquad v \in V, k \in K. \tag{2}$$

The next set of constraints expresses the demand constraints and flow conservation constraints for the backup paths:

$$\sum_{\ell \in \omega^+(v)} p^{\text{B}}_{k\ell} - \sum_{\ell \in \omega^-(v)} p^{\text{B}}_{k\ell} = \begin{cases} 1 & \text{if } v = v_k \\ -\,d^{\text{B}}_{kv} & \text{if } v \in V_d \\ 0 & \text{otherwise} \end{cases} \qquad v \in V, k \in K. \tag{3}$$

Then, we must ensure that working and backup paths do not overlap and do not share any link that could fail simultaneously. For that purpose, we introduce the following constraints:

$$p^{\text{W}}_{k\ell} + p^{\text{B}}_{k\ell} \leq 1 \qquad \ell \in L, k \in K \tag{4}$$

$$p^{\text{W}}_{k\ell} + p^{\text{B}}_{k\ell'} \leq 1 \qquad \ell, \ell' \in L : \ell \text{ and } \ell' \text{ are opposite to each other}, k \in K. \tag{5}$$

Next, we calculate the shared path protection capacities on each link ℓ:

$$b^{\text{B}}_{\ell'} \geq \sum_{k \in K} \delta_{k\ell\ell'} \qquad \ell, \ell' \in L : \ell \neq \ell' \tag{6}$$

$$\delta_{k\ell\ell'} \geq p^{\text{W}}_{k\ell} + p^{\text{B}}_{k\ell'} - 1 \qquad k \in K;\ \ell, \ell' \in L : \ell \neq \ell'. \tag{7}$$

In order to ensure that every demand (i.e., job request) is assigned to a single server, both in working provisioning and for backup purposes, we enforce the following constraints:

$$\sum_{v_d \in V_d} d^{\text{W}}_{kv} = 1 \qquad k \in K \tag{8}$$

$$\sum_{v_d \in V_d} d^{\text{B}}_{kv} = 1 \qquad k \in K. \tag{9}$$

Constraints (2)-(9) define the formulation for the SPR-A protection scheme. In order to get the formulation for the CSP-A scheme, we need to add the following constraints stating that the primary and backup servers have to be the same:

$$d^{\text{B}}_{kv} = d^{\text{W}}_{kv} \qquad v \in V_d, k \in K. \tag{10}$$

B. Column Generation ILP Model

While column generation techniques allow the solution of very large, even huge, ILP models, they often require to rethink the modeling in order to exhibit a decomposition of the set of constraints, and consequently to allow an implicit enumeration of the variables, its key feature for overcoming non scalability.

Here, in order to get a column generation formulation, we introduce the concept of a configuration $c \in C$, where C denotes the overall set of configurations. A configuration c is defined for a given source node $v_s \in V$, and describes a potential provisioning of the working and backup paths of a set of job requests originating at v_s. In the CSP-A protection scheme, destinations of a pair made of a working and a backup path must be the same server nodes, while in the SPR-A scheme, there is no such requirement. Of course, several such configurations exist and we denote by C_s the set of potential configurations associated with job requests originating at v_s.

The provisioning model of all job requests is then decomposed into: *(i)* a so-called Master Problem (MP) which will select the most promising / best configurations, a sufficient large number so as to satisfy the set of job requests for each source node, and *(ii)* so-called Pricing Problems (PP). Each pricing problem is associated with a given source node and generates potential configurations related to that source node.

The second change we introduce in order to get an efficient column generation is to define the traffic in a slightly different, but equivalent way. Let

K_s	Set of job requests originating at source node $v_s \in V \setminus V_d$.
D_s	$= \lvert K_s \rvert$, i.e., number of job requests in K_s.
S	$\subseteq V$, set of demand source nodes such that:

$$\forall v_s \in S : D_s > 0.$$

To complete the characterization of the configurations, we need the following parameters:

$p^{\text{W}}_{c\ell}$	$= 1$ if link ℓ is used by the working path of configuration c, 0 otherwise.
$p^{\text{B}}_{c\ell}$	$= 1$ if link ℓ is used by the backup path of c, 0 otherwise.

The master problem of the column generation ILP model uses two sets of variables: variables $z_c \in \mathbb{Z}^+$, $c \in C$ and $b_\ell \in \mathbb{Z}^+$. The value of each variable z_c is equal to the number of selected copies of configuration c. Variable b^{B}_{ℓ} is defined as in the ILP model of Section III-A.

1) Master Problem: The objective function which minimizes the total network capacity, can be written as follows:

$$\min \quad \text{COST}_{\text{CG\_ILP}}(z, b^{\text{B}})$$

where

$$\text{COST}_{\text{CG\_ILP}}(z, b^{\text{B}}) = \sum_{\ell \in L} \left(b^{\text{B}}_{\ell} + \sum_{c \in C} p^{\text{W}}_{c\ell}\, z_c \right). \tag{11}$$

The set of constraints are as follows. Firstly, we have the demand (job requests) constraints:

$$\sum_{c \in C_s} z_c \geq D_s \qquad v_s \in S. \tag{12}$$

Note that the demand of requests originating at v_s is not necessarily satisfied by a single configuration.

The next set of constraints expresses the capacity requirement for link ℓ' in a backup path. Indeed, if ℓ' protects link ℓ, with ℓ belonging to several working paths (modeled here throughout the various configurations associated with working paths containing ℓ), we must ensure that ℓ' has a large enough transport capacity:

$$\sum_{c \in C} p^{\text{W}}_{c\ell}\, p^{\text{B}}_{c\ell'}\, z_c \leq b^{\text{B}}_{\ell'} \qquad \ell, \ell' \in L : \ell \neq \ell'. \tag{13}$$

Note that, in practice, one works with the so-called *restricted master problem*, i.e., with a master problem restricted to a very small set of configuration variables. See Section III-B4 for a description of the algorithm for solving the CG-ILP model.

2) Pricing Problem: Each pricing problem corresponds to the design of a potential configuration, i.e., a potential working and backup provisioning for the job requests originating from a given source node $v_s \in V$. Per definition of the pricing problem, the objective function corresponds to the reduced cost of the configuration variable of the master problem, i.e., of variable z_c for $c \in C_s$, assuming we search for configurations in C_s. Readers not familiar with linear programming concepts, are referred to [22], [23].

In addition, the interest of the pricing problem lies in the identification of improving configurations, i.e., configurations c such that, if their corresponding variable z_c is added in the master problem, it will contribute to improve (here, to minimize further) the current value of the objective of the master problem. Such configurations are the ones with a negative reduced cost. In other words, assuming we minimize the reduced cost of the current pricing problem associated with source node v_s, either the minimum reduced cost is negative, and then we have obtained an improving configuration that we add to the current master problem, or the minimum reduced cost is positive. In the latter case, we conclude that, at this stage, no more improving configuration associated with v_s can be found, unless the values of the dual variables change following the addition of another configuration associated with another source node.

Let us express the objective function of the pricing problem associated with source node v_s, or PP(v_s) for short, i.e., the reduced cost of variable $z_c, c \in C_s$. For doing so, we need the dual values of the constraints involving variable z_c:

- u^1 ≥ 0, value of the dual vector associated with constraint (12-v_s) (we omit the s index to alleviate the notation),
- $u^2_{\ell\ell'}$ ≤ 0, values of the dual vector associated with constraints (13).

The reduced cost, $\overline{\text{COST}}_{\text{CG\_ILP}}$, of PP($v_s$), to be minimized, can then be written:

$$\overline{\text{COST}}_{\text{CG\_ILP}} = \sum_{\ell \in L} p^{\text{W}}_{\ell} - u^1 - \sum_{\ell \in L} \sum_{\ell' \in L : \ell \neq \ell'} u^2_{\ell\ell'} p^{\text{W}}_{\ell} p^{\text{B}}_{\ell'}. \tag{14}$$

Constraints are related to the working and backup provisioning of the job requests originating from v_s. The next two sets of constraints take care of the working and backup path definitions.

$$\sum_{\ell \in \omega^+(v)} p^{\text{W}}_{\ell} - \sum_{\ell \in \omega^-(v)} p^{\text{W}}_{\ell} = \begin{cases} 1 & \text{if } v = v_s \\ d^{\text{W}}_v & \text{if } v \in V_d \\ 0 & \text{otherwise} \end{cases} \quad v \in V, \tag{15}$$

$$\sum_{\ell \in \omega^+(v)} p^{\text{B}}_{\ell} - \sum_{\ell \in \omega^-(v)} p^{\text{B}}_{\ell} = \begin{cases} 1 & \text{if } v = v_s \\ d^{\text{B}}_v & \text{if } v \in V_d \\ 0 & \text{otherwise} \end{cases} \quad v \in V. \tag{16}$$

The next two sets of constraints deal with the overlap and the sharing of links pertaining to the working and backup paths, and are similar to constraints (4) and (5):

$$p^{\text{W}}_{\ell} + p^{\text{B}}_{\ell} \leq 1 \qquad \ell \in L \tag{17}$$

$$p^{\text{W}}_{\ell} + p^{\text{B}}_{\ell'} \leq 1 \qquad \ell, \ell' \in L : \ell \text{ and } \ell' \text{ are opposite to each other.} \tag{18}$$

Again, we need to impose a single node server for each path, i.e., working and backup:

$$\sum_{v \in V_d} d^{\text{W}}_v = 1, \tag{19}$$

$$\sum_{v \in V_d} d^{\text{B}}_v = 1. \tag{20}$$

This concludes the description of the set of constraints for the SPR-A scheme. For the CSP-A scheme, we have to enforce the constraints stating that the primary and backup servers need to be the same:

$$d^{\text{B}}_v = d^{\text{W}}_v \qquad v \in V_d. \tag{21}$$

3) Linearization: As can be observed, the expression of the reduced cost (14) is nonlinear. In order to linearize it, we introduce the variables $p^{\text{WB}}_{\ell\ell'}$ such that:

$$p^{\text{WB}}_{\ell\ell'} = p^{\text{W}}_{\ell} p^{\text{B}}_{\ell'} \qquad p^{\text{W}}_{\ell}, p^{\text{B}}_{\ell'} \in \{0,1\}; \ell, \ell' \in L : \ell \neq \ell' \tag{22}$$

together with the following set of constraints:

$$p^{\text{WB}}_{\ell\ell'} \geq p^{\text{W}}_{\ell} + p^{\text{B}}_{\ell'} - 1 \qquad \ell, \ell' \in L : \ell \neq \ell' \tag{23}$$

$$p^{\text{W}}_{\ell} \leq p^{\text{WB}}_{\ell\ell'} \qquad \ell, \ell' \in L : \ell \neq \ell' \tag{24}$$

$$p^{\text{B}}_{\ell'} \leq p^{\text{WB}}_{\ell\ell'} \qquad \ell, \ell' \in L : \ell \neq \ell'. \tag{25}$$

Not that inequalities (24) and (25) are not necessary, taking into account that variables $p^{\text{WB}}_{\ell\ell'}$ appear in the objective of the pricing problem with a negative coefficient, as $u^2_{\ell\ell'} \leq 0$, (see below) and, hence are minimized.

The expression of the objective (i.e., reduced cost) of the pricing problem PP(v_s) becomes:

$$\overline{\text{COST}}_{\text{CG\_ILP}} = \sum_{\ell \in L} p^{\text{W}}_{\ell} - u^1 - \sum_{\ell \in L} \sum_{\ell' \in L : \ell \neq \ell'} u^2_{\ell\ell'}\, p^{\text{WB}}_{\ell\ell'}. \tag{26}$$

4) Solution of the CG-ILP formulation: Column Generation (CG) techniques offer highly efficient solution methods for linear programs with a very large number of variables, where the constraints can be expressed implicitly. In order to speed-up the convergence of a column generation model, it is very often useful to use a "warm" start, i.e., to generate few as promising as possible configurations at the outset. This was achieved by solving PP(v_s) for $v_s \in S$, after modifying its objective as follows:

$$\min \quad \sum_{\ell \in L} \left(p^{\text{W}}_{\ell} + p^{\text{B}}_{\ell} \right). \tag{27}$$

The set of constraints is made of constraints (15)-(21).

On the other hand, one needs to devise a way to derive an integer solution once the linear relaxation of an ILP model has been solved using a column generation algorithm. Here, rather than developing a costly branch-and-cut algorithm, we solve the ILP model made of the columns generated in order to obtain the optimal linear programming solution. It is well known that it usually does not provide the optimal ILP solution, but, as will be seen in the numerical results section, in practice, that was enough in order to obtain near optimal solutions.

The detail of the column generation and ILP solution process is described in Algorithm III.1.

Algorithm III.1 Solution of the CG-ILP model

```
Step 1. Initialization
Build a set of initial configurations in order to set an initial
Restricted Master Problem (RMP).

Step 2. Solution of the linear relaxation of the master problem
Solve the LP relaxation of the current RMP
OPT ← .FALSE.
while OPT = FALSE do
  OPT ← .TRUE.
  for each source node $v_s$ do
    Solve PP($v_s$)
    if $\overline{\text{COST}}_{\text{CG\_ILP}}(\text{PP}(v_s)) < 0$ then
      OPT ← .FALSE.
      Add the improving configuration associated with PP($v_s$) to the current RMP
      Re-optimize the LP relaxation of the enlarged RMP
    end if
  end for
end while

Step 3. Deriving an optimal or a near optimal integer solution
Solve the ILP model made of the current set of columns (variables) of the RMP,
using either a branch-and-bound technique or a rounding off technique.
```

IV. Heuristics

While the classical ILP formulation presented in III-A allows to find an optimal solution, it does not scale at all for large data instances. Hence, in order to evaluate the relocation strategy on a larger scale, we proposed, in Section III-B, a new CG-ILP model based on a column generation formulation. This last model allows the solution of large size instances, while providing an optimal or a near optimal solution. We next propose two heuristic algorithms, in an attempt to find faster solution algorithms, without compromising too much on the quality of the solutions. The first heuristic, denoted by H1, improves the running time for medium size instances over CG-ILP, while finding solutions with a small optimality gap. The second one, denoted by H2, is faster than H1, and much faster than the CG-ILP algorithm, but outputs solutions with a larger optimality gap, especially for the CSP-A case. We next describe those two heuristics.

A. Heuristic H1

We adapted a heuristic described in [24] which tries to minimize the total resource usage by minimizing the resources for the primary connections as well as by maximizing the sharing among the backup network resources. We extended this heuristic to the grid case under the anycast principle, with the selection of the server nodes. We first describe the heuristic for the SPR-A scheme and further show how we can adapt it to achieve the CSP-A scheme.

1) Overview of heuristic H1: Heuristic H1, which is described in Algorithm IV.1, proceeds in three steps. We next comment those steps.

Algorithm IV.1 Heuristic H1 - SPR-A Protection Scheme

```
1: Step 0. Initialization
2: for $k \in K$ do
3:   $p^{\text{W}}_k \leftarrow \emptyset$ ; $p^{\text{B}}_k \leftarrow \emptyset$
4: end for
5:
6: Step 1. Create virtual resource $v_{n+1}$
7: for $v \in V_d$ do
8:   create two parallel links between $v$ and $v_{n+1}$, where
     node $v_{n+1}$ plays the role of a sink node.
9: end for
10:
11: Step 2. Find a candidate link disjoint pair of paths
12: for $k \in K$ (where $K$ is an ordered set) do
13:   $(p_1, p_2) \leftarrow$ Suurballe's algorithm$(s, v_{n+1})$
14:   $p^{\text{W}}_k = \arg\min(\text{LENGTH}(p_1), \text{LENGTH}(p_2))$
15:   $p^{\text{B}}_k = \arg\max(\text{LENGTH}(p_1), \text{LENGTH}(p_2))$
16: end for
17: Compute COST_H1 associated with those primary and
    backup paths
18:
19: Step 3. Optimization phase
20: # Changes ← 0 ; $index \leftarrow -1$
21: while # Changes $\leq |K|$ do
22:   $index = (index + 1) \mod |K|$
23:   $k \leftarrow K[index]$
24:   $p^{\text{W}}_k \leftarrow$ Dijkstra's algorithm$(v_s(k), v_{n+1})$
25:   $p^{\text{B}}_k \leftarrow$ FindBackupPath$(v_s(k), p^{\text{W}}_k, v_{n+1})$
26:   Compute NEW_COST_H1
27:   if NEW_COST_H1 < COST_H1 then
28:     # Changes ← 0
29:     COST_H1 ← NEW_COST_H1
30:   else
31:     # Changes ← # Changes + 1
32:   end if
33: end while
```

Algorithm IV.2 Algorithm FindBackupPath$(v_d, p_k^{\mathrm{W}}, v_{n+1})$

```
1: Remove the links of $p_k^{\mathrm{W}}$ in graph $G$
2: *For each backup path with a corresponding primary that is disjoint with $p_k^{W}$, set the link weights to zero*
3: **for** $k' \in K \setminus \{k\}$ **do**
4:   **if** $(p_k^{\mathrm{W}} \cap p_{k'}^{\mathrm{W}}) = \emptyset$ **then**
5:     **for** $\ell \in p_{k'}^{\mathrm{B}}$ **do**
6:       assign weight 0 to $\ell$
7:     **end for**
8:   **end if**
9: **end for**
10: **return** Dijkstra's algorithm$(v_s(k), v_{n+1})$
```

Step 1: We insert a *virtual resource* (i.e., a sink node) (lines 7-9), which is biconnected with a virtual edge (i.e., two links opposite to each other) of weight 0 to every other resource. Such a virtual resource makes it easy to find a pair of link disjoint paths to different potential resources. If we find two link disjoint paths to this virtual resource, the real resource is the second-last node (next-to-last hop) on each path.
Step 2: For every connection request $k \in K$ (line 12), find a pair of link disjoint paths from the fixed source to the virtual resource (line 13), using Suurballe's algorithm [25] (see also [26]), a reference algorithm for finding two link disjoint paths of minimum total weight. Assign the shortest path to the primary path, p_k^{W} (line 14), and the other path to the backup path, p_k^{B}, (line 15). We choose the longest as backup, since wavelengths along this path will (hopefully) be shared with others. The wavelengths on primary path links on the other hand need to be exclusively reserved for this particular request.
Step 3: For every connection (line 21), try to find a new primary resource (line 24, using Dijkstra's algorithm [27]). The search of the new backup path (line 25), using the procedure FindBackupPath, described in algorithm IV.2. Therein, we first delete the primary path, after which we consider every connection $k' \neq k$. If primary paths $p_{k'}^{\mathrm{W}}$ and p_k^{W} are link disjoint, we assign weight 0 to the links $\ell \in p_{k'}^{\mathrm{B}}$. Applying Dijkstra's algorithm on the modified network from the source to the virtual resource leads to a new backup path with a cost hopefully not greater than the cost of the previous backup path and even smaller because of possible additional sharing. This last step differs from [24] as we combine the separate rerouting steps into one step. Such a combination accommodates for the extra degree of freedom (vs. [24]) since we start from a source demand vector, rather than from an origin/destination demand matrix.

In order to accommodate all backup paths, the total number of bandwidth units on each link ℓ is calculated as follows:

$$b_\ell^{\mathrm{B}} = \max_{\ell' \in L} \sum_{k \in K} p_{k\ell}^{\mathrm{B}} \cdot p_{k\ell'}^{\mathrm{W}}. \tag{28}$$

2) Extending H1 heuristic for the solution of CSP-A: The introduction of the virtual resource is a handy trick in order not to exhaustively optimize over all possible resources and then choosing the best one. The trick cannot be used for CSP-A because the end points of the pair of link disjoint paths need to be the same. Hence, there is no other possibility than exhaustively iterate over every possible resource in both the initial configuration phase and the optimization phase. Note, however, that this exhaustive search for all resources is feasible, since we assume a reasonably small set V_d of

Algorithm IV.3 Heuristic H2 - SPR-A Protection Scheme

```
1: **Step 1:** *Initialization*
2: For all $\ell \in L$: $b_\ell^{\mathrm{B}} \leftarrow 0$ ; $\mathrm{WEIGHT}_\ell^{\mathrm{W}} \leftarrow 1$,
3:
4: **Step 2:** *Primary and backup paths*
5: **for all** $v_s \in V \setminus V_d$ **do**
6:   Concatenate all the requests originating at $v_s$ into a single aggregated request, denoted by $k(v_s)$, with a bandwidth requirement such that: $b_{k(v_s)} = \sum_{k \in K_s} b_k$.
7:   **Step 2a:** *Selection of the grid server location*
8:   **for all** $\ell \in L$ **do**
9:     $\mathrm{WEIGHT}_\ell^{\mathrm{B}} \leftarrow \left(\max_{\ell \in L} b_\ell^{\mathrm{B}}\right) - b_\ell^{\mathrm{B}} + 1$
10:  **end for**
11:  **for all** $v_d \in V_d$ **do**
12:    Compute the shortest path $p_{v_s v_d}$ from $v_s$ to $v_d$ with weights $\mathrm{WEIGHT}^{\mathrm{W}}$
13:  **end for**
14:  $p_s^{\mathrm{W}} \leftarrow \arg\min_{v_d \in V_d} \{\mathrm{LENGTH}(p_{v_s v_d})\}$ where $\mathrm{LENGTH}(p_{v_s v_d})$ is computed according to $\mathrm{WEIGHT}^{\mathrm{W}}$
15:
16:  **Step 2b:** *Tentative selection of the primary path*
17:  Temporarily remove from $G$ the links of $p_s^{\mathrm{W}}$
18:
19:  **Step 2c:** *Selection of the backup path and confirmation/new computation of the primary path*
20:  **if** there exists a path from $v$ to a server site **then**
21:    For all $v_d \in V_d$: Compute the shortest path $p_{v_s v_d}$ from $v_s$ to $v_d$ with weights $\mathrm{WEIGHT}^{\mathrm{B}}$
22:    $p_s^{\mathrm{B}} \leftarrow \arg\min_{v_d \in V_d} \{\mathrm{LENGTH}(p_{v_s v_d})\}$ where $\mathrm{LENGTH}(p_{v_s v_d})$ is computed according to $\mathrm{WEIGHT}^{\mathrm{B}}$
23:    Restore graph $G$ (put back all links)
24:  **else**
25:    Restore initial graph $G$ (put back all links)
26:    Compute the shortest pair of link disjoint paths between $v_s$ and $v_d$ with weights $\mathrm{WEIGHT}^{\mathrm{W}}$ and $\mathrm{WEIGHT}^{\mathrm{B}}$, for all $v_d \in V_d$.
27:    Let $p'$ and $p''$ be the two resulting routes. Let
         $p_s^{\mathrm{W}} = \arg\min \{ \mathrm{LENGTH}(p'), \mathrm{LENGTH}(p'') \}$;
         $p_s^{\mathrm{B}} = \arg\max \{ \mathrm{LENGTH}(p'), \mathrm{LENGTH}(p'') \}$.
28:  **end if**
29:  Update the bandwidth requirements ($b_\ell^{\mathrm{W}}$ and $b_\ell^{\mathrm{B}}$) on the links of the primary and backup paths. For $b_\ell^{\mathrm{B}}$, the updating formula is as follows:
         $b_\ell^{\mathrm{B}} = \max_{\ell' \in L} \left( \sum_{k \in K : \ell' \in p_k^{\mathrm{W}}, \ell \in p_k^{\mathrm{B}}} b_k \right)$,
     where $p_k^{\mathrm{W}}$ (resp. $p_k^{\mathrm{B}}$) is the aggregated working (resp. backup) path of request $k$.
30: **end for**
```

resource sites. This choice is motivated by [16] which shows that a small number of resource sites suffices and allows the minimization of the overall network load.

In order to get a solution for the CSP-A protection scheme, heuristic H1 should be modified as follows:

- Remove Step 1 (lines 7 to 9),
- For each server site, calculate a new primary path and an optimized backup path, following the approach of lines 14–15, and choose the combination that leads to the lowest bandwidth requirements (i.e. that minimizes COST_H1).

B. Heuristic H2

In this section, we describe another heuristic algorithm, H2, in an attempt to design a more scalable heuristic algorithm than heuristic H1. As we will see in Section V, we were quite successful in that attempt for the scalability aspect, less for the accuracy part. A key difference between H1 heuristic and H2 heuristic is that in H2, we combine all the requests originating from the same source node, as in the master problem of CG-ILP, while in H1, requests are considered on an individual basis, which increases the complexity of H1.

The H2 heuristic is based on an iterative approach which is detailed in Algorithm IV.3.

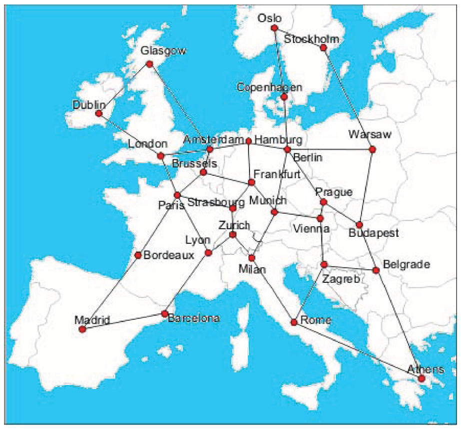
Shortest paths are computed using different weights for primary and backup path calculation. Backup weights account for sharing of wavelengths, while working weights account for the length of the path only:

WEIGHT$_\ell^{\mathrm{W}}$: Primary weights are all taken equal to one, meaning that when computing shortest paths with those weights, we indeed consider the length of the working paths in terms of the number of links they contain.

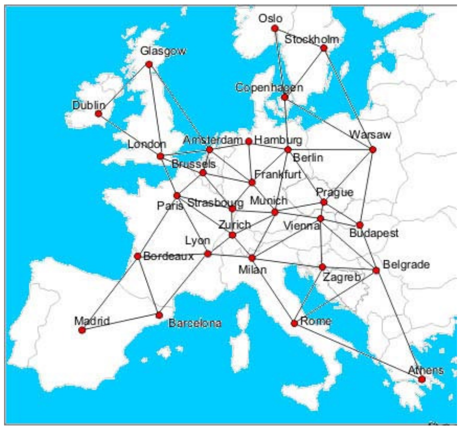
WEIGHT$_\ell^{\mathrm{B}}$: Backup weights are initialized to one, and will contain the complement of the protection bandwidth requirements with respect to the maximum link bandwidth requirement, see line 9. The reason is as follows. When computing shortest paths, we can either minimize or maximize their overall bandwidth requirements. When maximizing, instead of changing the shortest path algorithm in a longest path algorithm, one can also complement the protection weights with respect to the largest weight in order to go on using a shortest path algorithm (this is what is done on line 9 of the heuristic).

The underlying idea of the definition of the weights for the search of the backup path is that there are more opportunities for sharing with the links already contributing to bandwidth protection, or, in other words, the more protection bandwidth a link has, the more protection bandwidth sharing the link offers. For a given source node, there might be several requests. It is the choice of the network manager to route them all on the same primary paths or not. Indeed, it is not mandatory to assign each of the requests originating at the same node with the same server, and to assign them the same backup path. However, this is the choice which has been made in the H2 heuristic for scalability purposes.

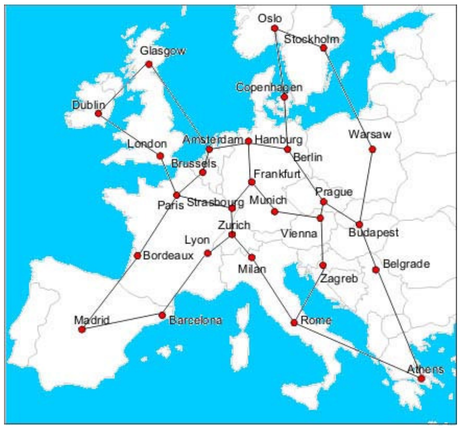
1) Extending H2 heuristic for the solution of CSP-A: As for heuristic H1, heuristic H2 can be easily adapted to the CSP-A scheme: the search for paths simplifies as they are restricted to pairs of working/backup paths with the same destinations.



(a) EU-base (the base topology from [28])



(b) EU-dense (the triangular topology from [28])



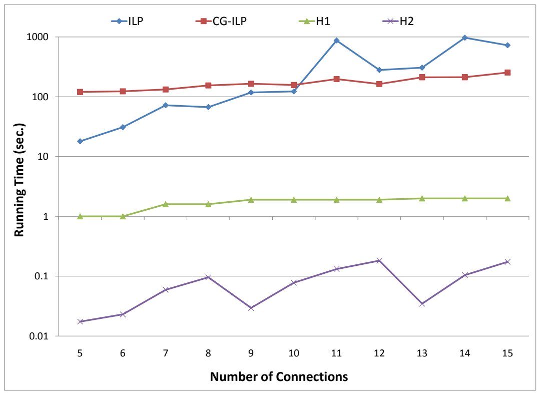
(c) EU-sparse (the ring topology from [28])

Fig. 1. The original pan-European network topology and two variants of it.

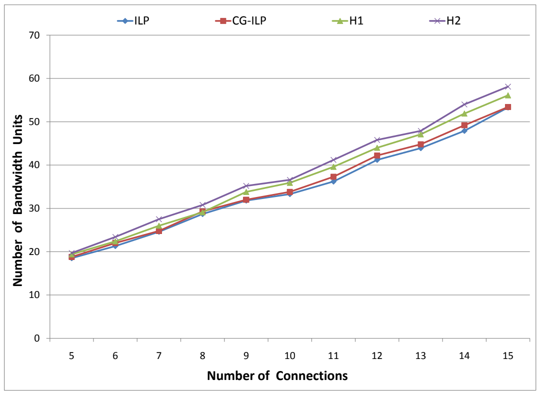
V. Performance Evaluation

A. Experiment Set-up

We will compare the performances for the Classical Shared Path Protection (CSP-A) and the Shared Path Protection with Relocation (SPR-A) schemes, both under the Anycast routing principle. In order to evaluate the influence of the topology on the achievable savings, we will compare three different topologies [28] as depicted in Fig. 1: (a) *EU-base*: a meshed network topology comprising 28 sites and 41 bidirectional links, corresponding to the pan-European network

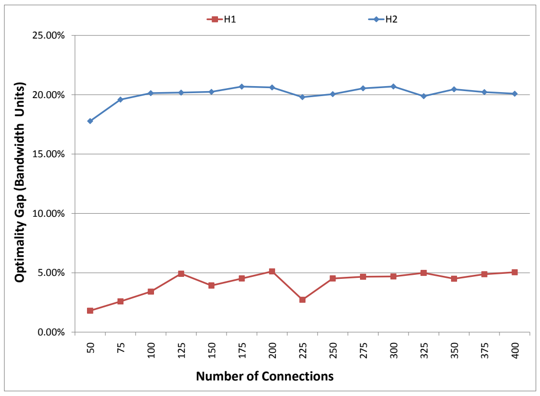


(a) Computing times (logarithmic scale)

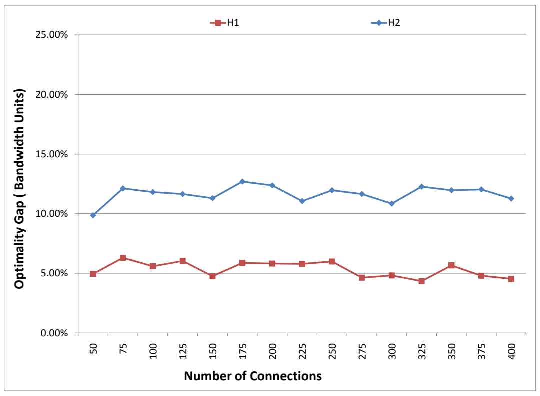


(b) Number of bandwidth units

Fig. 2. Compared Performances of ILP, CG-ILP, H1 and H2 on Small Data Sets (SPR-A Protection Scheme).



(a) CSP-A



(b) SPR-A

Fig. 3. Performances of H1 and H2 compared to CG-ILP.

of the LION and COST ACTION 266 projects; (b) *EU-dense*: a denser variant, with the same number of nodes, but 59 bidirectional links; and (c) *EU-sparse*: a sparser variant, again with the same node set, but with only 35 bidirectional links.

Traffic instances were generated as follows: for a given number, say $|K|$, of job requests, we randomly select $|K|$ source nodes $v_s \in V \setminus V_d$. The number of times a source node is selected gives the number of job requests originating from that node. Nodes which are hosting server nodes are excluded.

We compare the solutions of the two ILP models, as well as the solutions of the two heuristics described in the previous sections. We consider different sets of fixed server nodes:

$$\begin{aligned} V_d^3 &= \{\text{London, Vienna, Berlin}\} \\ V_d^5 &= V_d^3 \cup \{\text{Lyon, Zurich}\} \\ V_d^7 &= V_d^5 \cup \{\text{Munich, Zagreb}\} \end{aligned}$$

We use the IBM ILOG CPlex solver (release 11) to solve the ILP models under a C++/java implementation. All programs have been run on a cluster server node with 1 CPU of 2.2 GHz AMD Opteron 64-bit processor, 8Gb ram. In the forthcoming figures, each data point corresponds to average results over 10 random traffic instances.

B. Quality of the Solutions

1) Accuracy of the solutions: Before comparing the performances of the CSP-A and SPR-A protection schemes, it is necessary to have a look at the quality (i.e., accuracy) of the solutions output by the CG-ILP algorithm and the two heuristics. We measure the accuracy by comparing the heuristic values with the ILP values, using the ILP model for the small instances, and the CG-ILP model for the large instances. In order to do so, we conducted experiments on the base pan European network topology of Fig. 1(a), with 5 server nodes (set V_d^5).

In our previous work [20], we already compared the quality of the solutions provided by CG-ILP and an earlier version of H1, noted as H1$'$, for both the classical shared path protection (CSP) and the shared path protection with relocation (SPR) schemes, assuming the location of the servers was given at the outset, in the description of each job request. Therein, we observed that both CG-ILP and H1$'$ found very close solutions (less than 1% optimality gap) to the optimal ILP solution, on small instances, i.e., with a number of

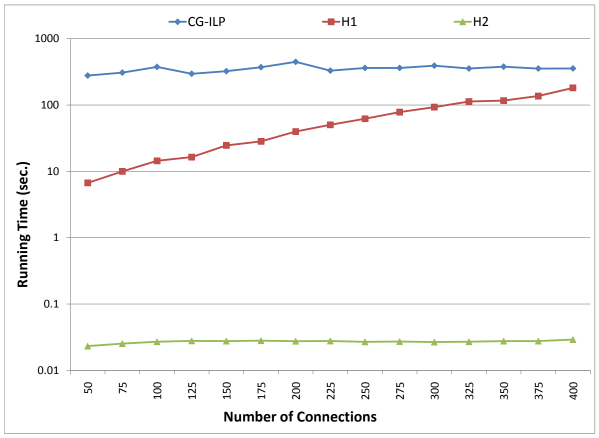


Fig. 4. Running times for SPA-R protection scheme.

requests less than 20. On larger instances, the ILP model is not scalable anymore, and we observed that CG-ILP and H1′ solutions were very close, with the CG-ILP algorithm being faster than H1, the more so as the number of requests was increasing. In addition, the optimality gap of CG-ILP was equal to 1% on average, while it was equal to $\approx 5\%$ for heuristic H1′.

If we now look at the CSP-A and SPR-A protection schemes, where the server location is not given at the outset (in comparison with the CSP and SPR schemes in [20]), we observe similar results for small data sets. We only provide the results for the SPR-A protection scheme, since the qualitative results for the schemes, CSP-A and SPR-A, are very similar. Indeed, for small data sets, where the classical ILP model remains solvable, see Figure 2, we observe that the ILP and the CG-ILP solutions are very close, meaning that the CG-ILP model leads to near optimal solutions which are within less than 2% accuracy, while the H1 heuristic finds solutions close to the optimal one (< 2% accuracy), see Figure 2(b). The comparison also includes heuristic H2, which is a faster heuristic than H1, at the expense of a larger optimality gap of 9%. With respect to the computing times (see Figure 2(a)), the heuristics are much faster than the two ILP algorithms. Observe that the ILP model's lack of scalability is visible from the clear increase in running time for larger demands (note the logarithmic scale), whereas the running time for CG-ILP seems more stable for increasing demands.

For larger data sets, the results are described in Figure 3. We have noted that CG-ILP has an optimality gap $< 0.5\%$ which means we get optimal solutions from a practical point of view. In both figures, we provide the relative performances of the two heuristics, H1 and H2, with respect to CG-ILP. The relative optimality gaps are computed as follows:

$$\frac{\text{COST}^\star_{\text{H1}} - \text{COST}^\star_{\text{CG-ILP}}}{\text{COST}^\star_{\text{H1}}} \quad \text{and} \quad \frac{\text{COST}^\star_{\text{H2}} - \text{COST}^\star_{\text{CG-ILP}}}{\text{COST}^\star_{\text{H2}}},$$

where $\text{COST}_\square$ denotes the cost value found by the $\square$ model/algorithm. Comparisons are made in Figure 3(a) for the CSP-A protection scheme, and in Figure 3(b) for the SPR-A protection scheme. The key observations are that the H1 heuristic provides better solutions than the H2 heuristic, but at the expense of longer computing times, as discussed below. Indeed, for both protection schemes, the H1 heuristic provides solutions with an average of 5% accuracy, compared to the CG-ILP solutions, while the relative accuracy varies between 10% and 20% for the H2 heuristic.

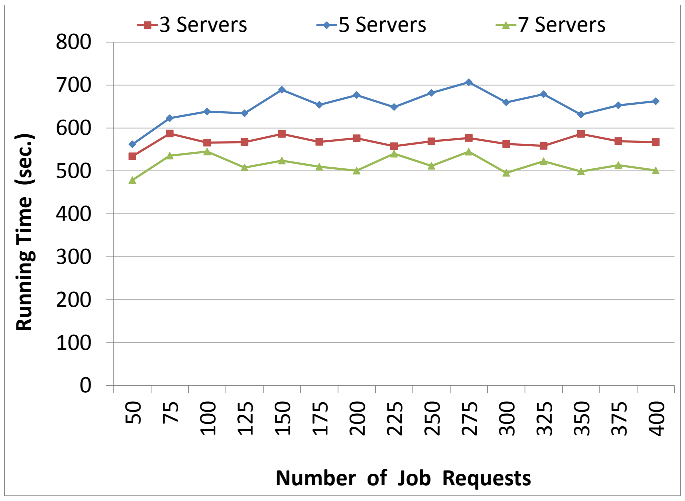
2) Computing times vs. solution accuracy: Note that the problem we consider in the paper is an offline dimensioning problem, i.e., how to determine the network dimensions and routing for a given (average) traffic pattern. Thus, we decide how many wavelengths that need to be activated on each link, as well as paths to set-up from traffic sources to server sites (i.e. data centers). The actual grid scheduler should then make use of these paths to send jobs and actually run them. Thus, the timescale at which to run our algorithms is an order of magnitude larger than that of, e.g., job inter-arrivals.

The results discussed here have again been obtained for the EU-base topology with 5 server sites. The optimal solution of the ILP model can only be compared with the other solutions on small data sets. There, we observe that the CG-ILP model very quickly provides near optimal solutions with a very good accuracy (less than 2%). In addition, on average, CG-ILP has smaller computing times than ILP as soon as we have more than 10 connection requests. The H1 solution is less accurate, with a consistent gap around 5%, for both the CSP-A and SPR-A schemes, but its computing times are much smaller than those of the solutions for the ILP models. Similar observations can be made for H2, which is even faster than H1, but with a reduced accuracy (around 9%).

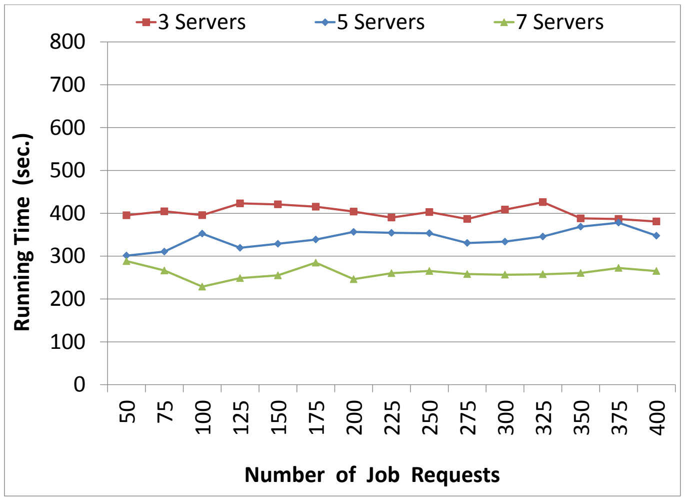
On larger data sets, only the solutions of the CG-ILP, H1 and H2 algorithms can be compared, see Figure 4. We observe that both CG-ILP and H2 algorithms are not sensitive to the number of requests, with H2 being much faster than CG-ILP. On the other hand, the computing times for H1 are increasing with the number of requests, and when the number of requests exceeds 500, H1 has higher computing times than CG-ILP. As shown by the results depicted in Figure 4, H1 provides better solutions than H2. However, when accuracy is not a major concern, but routes need to be found very fast, H2 is an interesting alternate choice and scales to very large demand sets.

C. Influence of the Number of Server Sites and the Topology

1) Number of servers: We compare here the performances of the CG-ILP algorithm with different numbers of resources (server nodes): 3, 5, and 7. Results are shown in Figure 5(a) (resp. 5(b)) for the CSP-A (resp. SPR-A) protection scheme. We observe, that for the CSP-A scheme, computing times are higher for 5 server locations than for 3, while computing times for 3 are higher than those for 7 server locations. For the SPR-A scheme, the running times with 3 server nodes are higher than with 5, and running times with 5 server nodes are higher than those with 7 server locations. We made experiments with a different data set, where the Berlin server was relocated in Copenhagen. Again, the results (not shown here) gave similar running times for 3 and 5 server locations, and lower ones for 7 server locations than for 3 or 5 server locations. Therefore, from the two case studies, no clear trend can be observed in runtime dependency on the number of server sites.

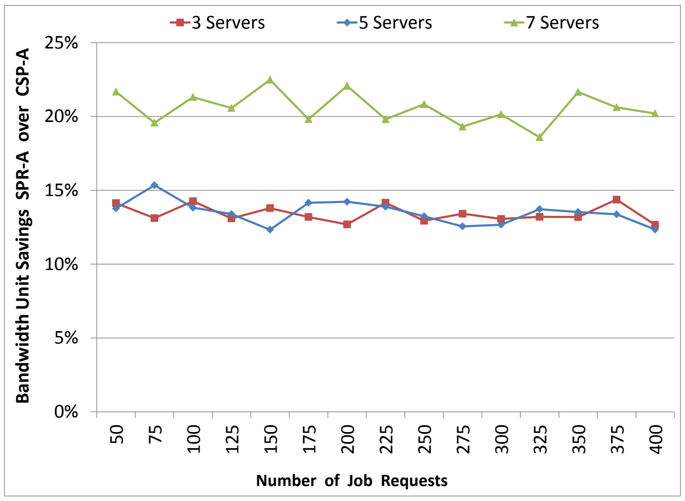


(a) CSP-A

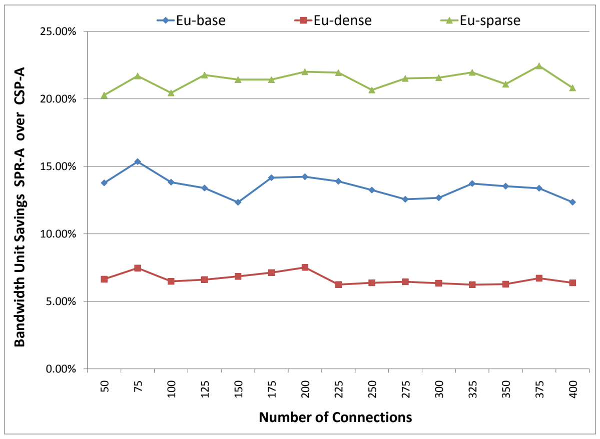


(b) SPR-A

Fig. 5. Comparison of the running times for different numbers of server nodes on the EU-base topology (CG-ILP algorithm).



(a) Depending on the # of server nodes on the EU-base topology



(b) Depending on the density of the network topology with 5 server sites

Fig. 6. SPR-A vs. CSP-A protection schemes with respect to the number of bandwidth units - (CSP-A - SPR-A) / CSP-A.

2) Impact of the topology connectivity: We next analyze the effect of the topology. For doing so, we considered the EU networks comprising the same number of nodes, but with different number of links (i.e., connectivity). We again considered the case for 5 server sites. Consequently, we investigate the performance of algorithm CG-ILP on the 3 topologies of the pan-European network (see Figure 1) described at the beginning of Section V: EU-base, EU-dense, EU-sparse with an average node degree of 2.93, 4.21, and 2.50 respectively.

Contrarily to the number of server sites, the topology seems a lot more influential, where a highly meshed network severely penalizes the execution time for CG-ILP, as observed in Figure 7. This was to be expected, since the number of possible paths increases.

3) Bandwidth savings by exploiting relocation: Lastly, we compared the bandwidth requirements of CSP-A and SPR-A, depending on the number of server nodes and the network topology. In Figure 6, we plotted the bandwidth savings that result from using the SPR-A scheme rather than the CSP-A scheme, using the ratio (bandwidth (CSP-A) – bandwidth (SPR-A)) / bandwidth (CSP-A). In all cases, there are meaningful bandwidth savings, which is rather stable with the number of job requests (experiments have been conducted for 50 up to 400 requests). On average, it is around 13% for 3 and 5 servers, and increases to around 21% for 7 servers. Indeed, the more servers, the more flexibility for an anycast scheme. With respect to the impact of the topology, the trend is as expecting, more bandwidth savings as the density is decreasing, i.e., bandwidth savings go from an average of 7% on a sparse topology, to an average of 13% for the base topology, and then to above 21% for the dense topology. It can be explained since, in such a ring-like network, a backup path to the same destination as the primary is likely to be quite long, and quite a bit longer (on average) than a path towards another server site.

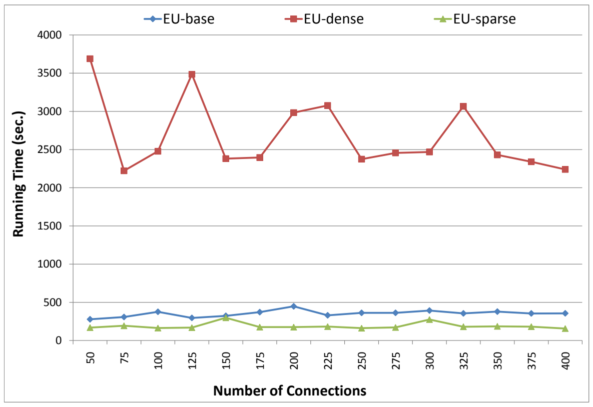


Fig. 7. Impact of the topology connectivity (CG-ILP algorithm): Running times for the SPR-A protection scheme.

VI. Conclusion and Future Work

In this paper, we formulated an ILP for network dimensioning purposes in an optical grid scenario with shared path protection against network single link failures. The fundamental difference with traditional RWA problems stems from the anycast routing principle: we also need to decide on the destination of the grid traffic (i.e. which grid server processes the submitted jobs originating from a particular source). Extensive case studies showed that solving the flow formulation ILP is not scalable, hence, we proposed heuristics able to solve large problem instances (with case studies ranging to networks of 28 nodes and 59 bidirectional links, and up to 400 connections). In addition, we also proposed a scalable exact method (CG-ILP) relying on column generation techniques, which offers a small to very small optimality gap (0.15 % and 1.8% on average for CG-ILP on large and small instances respectively).

With respect to the shared path protection scheme, we extended our earlier limited case studies [9] on assessing the amount of network bandwidth savings achievable by exploiting relocation. We investigated the influence of network topology, and in particular node degrees, on potential savings. We found that for lower node degrees, hence sparser networks, the potential savings are much higher; 7% for a European network with 28 nodes and average node degree of 2.5 (Fig. 1(b)) , versus 21% for node degree 4.21 (Figure 1(c)).

The network savings of our relocation strategy come at the price of increased load on the relocation servers. However, in reality this seemingly additional cost is one that would need to be made anyhow to provide resilience against server failures. Our future work will investigate this claim in more detail, by studying relocation-based protection mechanisms that offer survivability in case of both single node (including server node) and single link failures.

Acknowledgment

B. Jaumard was supported by NSERC (Natural Sciences and Engineering Research Council of Canada) and by a Concordia University Research Chair (Tier I) on the Optimization of Communication Networks. The work of C. Develder and J. Buysse was supported by the European Commission, through the GEYSERS (FP7–ICT–248657) project funded through the 7th ICT Framework Program, as well as by the Flemish government: C. Develder has been partly supported as a post-doctoral fellow of the Research Foundation - Flanders (FWO), J. Buysse was supported by the agency for Innovation by Science and Technology (IWT) with a Ph.D. grant.

References

[1] I. Foster and C. Kesselman, Eds., *The grid: blueprint for a new computing infrastructure*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 2004.

[2] M. De Leenheer, C. Develder, T. Stevens, B. Dhoedt, M. Pickavet, and P. Demeester, “Design and control of optical grid networks,” in *International Conference on Broadband Networks*, September 2007, pp. 107–115.

[3] A. Jaiszczyk, “Automatically switched optical networks: Benefits and requirements,” *Communications Magazine*, vol. 43, no. 2, pp. S10–S15, February 2005.

[4] T. Stevens, M. D. Leenheer, C. Develder, F. D. Turck, B. Dhoedt, and P. Demeester, “Anycast routing algorithms for effective job scheduling in optical grids,” in *European Conference on Optical Communications (ECOC)*, Cannes, France, September 2006, pp. 1–2.

[5] P. Anedda, S. Leo, S. Manca, M. Gaggero, and G. Zanetti. Suspending, migrating and resuming hpc virtual clusters. *Future Generation Computer Systems*, 26:1063–1072, 2010.

[6] Y.C. Lee and A.Y. Zomaya. Rescheduling for reliable job completion with the support of clouds. *Future Generation Computer Systems*, 26:1192–1199, October 2010.

[7] J. Buysse, M. D. Leenheer, C. Develder, and B. Dhoedt, “Exploiting relocation to reduce network dimensions of resilient optical grids,” in *Proceedings of IEEE/VDE Workshop on Design of Reliable Communication Networks - DRCN*, October 2009.

[8] J. Buysse, M. D. Leenheer, B. Dhoedt, and C. Develder, “Providing resiliency for optical grids by exploiting relocation: A dimensioning study based on ILP,” *Computer Communications*, p. In Press, 2011.

[9] ——, “On the impact of relocation on network dimensions in resilient optical grids,” in *Optical Network Design and Modeling - ONDM*, February 2010.

[10] X. Liu, C. Qiao, D. Yu, and T. Jiang. Application-specific resource provisioning for wide-area distributed computing. *IEEE Network*, 24(4):25–34, July/August 2010.

[11] R. Dutta and G. Rouskas, “A survey of virtual topology design algorithms for wavelength routed optical networks,” *Optical Networks Magazine*, vol. 1, no. 1, pp. 73–89, January 2000.

[12] H. Zang, J. P. Jue, and B. Mukherjee, “A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks,” *Optical Networks Magazine*, pp. 47–60, January 2000.

[13] B. Jaumard, C. Meyer, and B. Thiongane, “Comparison of ILP formulations for the RWA problem,” *Optical Switching and Networking*, vol. 4, no. 3-4, pp. 157–172, November 2007.

[14] ——, “On column generation formulations for the RWA problem,” *Discrete Applied Mathematics*, vol. 157, pp. 1291–1308, March 2009.

[15] ——, “ILP formulations for the RWA problem for symmetrical systems,” in *Handbook for Optimization in Telecommunications*, P. Pardalos and M. Resende, Eds. Kluwer, 2006, ch. 23, pp. 637–678.

[16] C. Develder, B. Mukherjee, B. Dhoedt, and P. Demeester, “On dimensioning optical grids and the impact of scheduling,” *Photonic Network Communications*, vol. 17, no. 3, pp. 255–265, June 2009.

[17] X. Liu, C. Qiao, W. Wei, X. Yu, T. Wang, W. Hu, W. Guo, and M.-Y. Wu, “Task scheduling and lightpath establishment in optical grids,” *Journal of Lightwave Technology*, vol. 27, no. 12, pp. 1796–1805, June 2009.

[18] T. Stidsen, B. Petersen, S. Spoorendonk, M. Zachariasen, and K. Rasmussen, “Optimal routing with failure-independent path protection,” *Networks*, vol. 2, no. 55, pp. 125–137, March 2010.

[19] A. Koster, A. Zymolka, M. Jäger, and R. Hulsermann, "Demand-wise shared protection for meshed optical networks," *Journal of Network and Systems Management*, vol. 13, no. 1, pp. 35–55, 2005.
[20] B. Jaumard, J. Buysse, A. Shaikh, M. Leenheer, and C. Develder, "Column generation for dimensioning resilient optical grid networks exploiting relocation," in *IEEE Global Telecommunications Conference - GLOBECOM*, 2010.
[21] K.-I. Sato, *Advances in Transport Network Technologies: Photonic Networks, ATM and SDH*. Artech House Publishers, 1996.
[22] V. Chvatal, *Linear Programming*. Freeman, 1983.
[23] L. Lasdon, *Optimization Theory for Large Systems*. New York: MacMillan, 1970.
[24] H. Zang, C. Ou, and B. Mukherjee, "Path-protection routing and wavelength assignment RWA in WDM mesh networks under duct-layer constraints," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 248–258, Apr. 2003.
[25] J. Suurballe, "Disjoint paths in a network," *Networks*, vol. 14, pp. 125–145, 1974.
[26] J. Suurballe and R. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1984.
[27] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269 – 271, 1959.
[28] S. De Maesschalck, D. Colle, I. Lievens, M. Pickavet, P. Demeester, C. Mauz, M. Jaeger, R. Inkret, B. Mikac, and J. Derkacz, "Pan-european optical transport networks: An availability-based comparison," *Photonic Netw. Commun.*, vol. 5, no. 3, pp. 203–225, May 2003.

Ali Shaikh Ali Shaikh is a Ph.D candidate of Computer Science in the Concordia University, Montreal. He has also received the M.Sc. Computer Science from the same university in 2008. His area of research is optimization in optical networks, specially in optical grid networks and passive optical networks (PONs). He has more than 10 years of industrial experience in the development of software engineering projects.

Jens Buysse J. Buysse a Phd. researcher at Ghent University, sponsored by the government agency for Innovation by Science and Technology. He received his Licentiate degree in Computer science in 2007 (now called Master degree) at Ghent University. His main interests lie in the field of photonic networks, distributed computing, virtualization techniques in optical network and Energy Efficient design principles. He participated in one National Project GEISHA where he was responsible for creating an application which made it possible to code videos in a distributed manner. He also participated in the European project PHOSPORUS where he helped developing a simulator, used for validating several algorithms developed during the project. He is currently involved in GEYSERS, where the goal is to qualify optical infrastructure providers and network operators with a new architecture, to enhance their traditional business operations.

Brigitte Jaumard Brigitte Jaumard holds a Concordia University Research Chair, Tier 1, on the Optimization of Communication Networks in the CIISE - Concordia Institute for Information Systems and Engineering - Institute at Concordia University. She was previously awarded a Canada Research Chair - Tier 1 - in the Department of Computer Science and Operations Research at Université de Montréal. She is an active researcher in combinatorial optimization and mathematical programming, with a focus on applications in telecommunications and artificial intelligence. Recent contributions include the development of efficient methods for solving large-scale mathematical programs, and their applications to the design and the management of optical and wireless, access and core networks. In Artificial Intelligence, contributions include the development of efficient optimization algorithms for probabilistic logic (reasoning under uncertainty) and for automated mechanical design. B. Jaumard has published over 150 papers in international journals in Operations Research and in Telecommunications.

Chris Develder C. Develder received the M.Sc. degree in computer science engineering and a Ph.D. in electrical engineering from Ghent University (Ghent, Belgium), in July 1999 and December 2003 respectively. From Oct. 1999 to Dec. 2003, he has been working in the Dept. of Information Technology (INTEC), at the same university, as a Researcher for the Research Foundation – Flanders (FWO), in the field of network design and planning. From Jan. 2004 to Aug. 2005, he worked for OPNET Technologies, on transport network design and planning. In Sep. 2005, he re-joined INTEC as a post-doctoral researcher, and as a post-doctoral fellow of the FWO since Oct. 2006. In Oct. 2007 he obtained a part-time, and since Feb. 2010 a fulltime associate professorship at Ghent University. He was and is involved in national and European research projects (IST David, IST Phosphorus, IST E-Photon One, BONE, IST Alpha, IST Geysers, etc.). His research interests include dimensioning, modeling and optimizing optical (Grid) networks and their control and management, smart grids, as well as multimedia and home network software and technologies.