NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF APPLIED SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

MASTER'S PROGRAM

MASTER THESIS

Path Computation for Resilient Networking

Evangelia Tsiontsiou

Supervisor: **Athanasia Allonistioti,** Lecturer
Nguengang Gerard, Thales Communications and Security

ATHENS

FEBRUARY 2013

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλγόριθμοι Δρομολόγησης για Εύρωστα Δίκτυα

Ευαγγελία Τσιόντσιου

Επιβλέποντες: **Αθανασία Αλωνιστιώτη,** Λέκτορας

ΑΘΗΝΑ

ΦΕΒΡΟΥΑΡΙΟΣ 2013

MASTER THESIS

Path Computation for Resilient Networks

Evangelia D. Tsiontsiou
R.N.: M 1189

Supervisor: **Athanasia Allonistioti,** Lecturer

EXAMINING COMMITTEE MEMBERS: **Athanasia Allonistioti,** Lecturer

February 2013

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αλγόριθμοι Δρομολόγησης για Εύρωστα Δίκτυα

Ευαγγελία Δ. Τσιόντσιου
Α.Μ.: Μ 1189

ΕΠΙΒΛΕΠΟΝΤΕΣ: Αθανασία Αλωνιστιώτη, Λέκτορας

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: Αθανασία Αλωνιστιώτη, Λέκτορας

Φεβρουάριος 2013

ABSTRACT

Current information systems require a very good quality of service (QoS) architecture and high tolerance to failures and disasters. Indeed, states, governmental and military organisations, companies and also the whole society rely more and more on the network for their daily activities. Therefore, the availability of those networks is crucial and any impairment can be dramatic. Building robust and resilient networks is mandatory and path protection and fast restoration appear to be a very good approach.

In order to increase the availability and resilience of their information systems, governmental and military organizations may rely on different networks to interconnect their sites such as private infrastructure networks, public ISP networks, SATCOM, tactical data links. This approach, called multi-homing is used to protect deployed services from impairments caused by potential network failures. Indeed, when a failure occurs on one of the networks, the traffic is automatically redirected on the other networks that serve as backup. However, although this protection scheme appears to be a good solution to increase the resilience of networks, its effectiveness and performance depends on the setup of full mesh interconnectivity between all the sites on each transit network. This approach is extremely costly and not always useful since all the services do not have the same operational criticality.

The existing work presents a set of path computation algorithms that were studied and extended to include the end user requirements (criticality, security, protection) in the establishment of both primary and backup paths. The robustness to faults and the cost by the setup of the protection scheme are evaluated in a military scenario where the sites are interconnected through three access networks.

SUBJECT AREA: Network Engineering

KEYWORDS: resilience, robustness, path computation, MCOP

ΠΕΡΙΛΗΨΗ

Τα τρέχοντα συστήματα τηλεπικοινωνιών απαιτούνε αρχιτεκτονική υψηλής ποιότητας υπηρεσιών (QoS – Quality of Service) και υψηλή ανοχή σε τυχόν προβλήματα και καταστροφές στο δίκτυο. Αυτό συμβαίνει καθώς η πολιτεία, οι κυβερνητικοί και στρατιωτικοί οργανισμοί, οι εταιρείες και ολόκληρη η κοινωνία στηρίζεται όλο και περισσότερο στα δίκτυα και τις τηλεπικοινωνίες για τις καθημερινές δραστηριότητες τους. Επομένως, η διαθεσιμότητα τους είναι εξαιρετικά σημαντική και οποιαδήποτε βλάβη σε αυτά μπορεί να δημιουργήσει τεράστια προβλήματα. Για αυτό το λόγο, η δημιουργία εύρωστων και στιβαρών δικτύων χρήζει τεράστιας σημασίας και μια καλή προσέγγιση για την πραγματοποίηση τους είναι η προστασία των μονοπατιών και η γρήγορη αποκατάσταση του δικτύου.

Οι κυβερνητικοί και στρατιωτικοί οργανισμοί, σκοπεύοντας να αυξήσουν την διαθεσιμότητα και την ελαστικότητα των πληροφοριακών συστημάτων τους, στηρίζονται σε διαφορετικά δίκτυα για την διασύνδεση των σταθμότοπων τους, όπως για παράδειγμα σε υποδομές ιδιωτικών δικτύων, δημόσιους παρόχους Internet, δορυφορικά δίκτυα και τακτικές ζεύξεις δεδομένων. Αυτή η προσέγγιση ονομάζεται πολυεστίαση (multihoming) και χρησιμοποιείται για να προστατέψει τις αναπτυσσόμενες υπηρεσίες από πιθανή κακή λειτουργία που προκαλείται από βλάβες του δικτύου. Πράγματι, όταν μια βλάβη πραγματοποιηθεί σε ένα από τα δίκτυα, η δρομολόγηση κίνησης γίνεται μέσω άλλου δικτύου αυτόματα λειτουργώντας σαν εφεδρική λύση. Παρόλα αυτά, αν και ο συγκεκριμένος τρόπος προστασίας των διαδρομών φαίνεται να είναι μια καλή λύση για την αύξηση της ελαστικότητας των δικτύων, η αποτελεσματικότητα και η απόδοση του εξαρτάται από την εγκαθίδρυση μια πλήρους διαπλεγμένης διασυνδεσιμότητας (full mesh interconnectivity) μεταξύ όλων των σταθμότοπων. Αυτή η προσέγγιση έχει τεράστιο κόστος και δεν είναι πάντα χρήσιμη εφόσον όλες οι υπηρεσίες δεν έχουν το ίδιο επίπεδο κρισιμότητας.

Η παρούσα εργασία παρουσιάζει ένα σύνολο αλγορίθμων υπολογισμού μονοπατιών (path computation algorithms) που μελετήθηκαν και επεκτάθηκαν έτσι ώστε να λαμβάνουν υπόψη τις απαιτήσεις των τελικών χρηστών (κρισιμότητα, ασφάλεια, προστασία) στην εύρεση πρωτεύοντος αλλά και δευτερεύοντος μονοπατιού. Η ευρωστία σε βλάβες και το κόστος της τεχνικής αποκατάστασης του δικτύου αξιολογούνται σε ένα στρατιωτικό σενάριο, όπου οι σταθμότοποι είναι διασυνδεδεμένοι διαμέσου τριών δικτύων πρόσβασης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Network Engineering

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: στιβαρότητα, ελαστικότητα, MPLS, δρομολόγηση, MCOP

CONTENTS

LIST OF DIAGRAMS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Problem Definition

The democratization of the Internet has drastically changed the way people communicate and live. Society, organizations, companies and even states rely on the network to sustain their everyday activities. Moreover, critical information systems such as SCADA also use the network to collect information from different sources and command critical infrastructures.

Therefore, ensuring a one hundred percent availability of the network is of paramount importance since any impairment can cause huge injury to all those activities which rely on the networks.

Since failures on large scale networks are unavoidable, organizations and companies rely on sites multi-homing which consists of connecting each corporate site to multiple ISP networks. Multi-homing aims to achieve some improvement in the reliability of the deployed services. In case of degradation or failure on one of the connections (due to a trouble in the core network), the traffic will be shifted to the second possible connection on q different network. However, multi-homing seems to be a satisfactory answer to the network resiliency issue if and only if all sites (e.g. of a given corporation) are interconnected in a full mesh way on each of the ISPs. This is not always feasible especially in the military context where ISPs PoP (Point of Presence) are not available everywhere. Furthermore, this approach is not cost effective since strong resilience will require the reservation of the same amount of resources on each of the involved ISPs networks.

At the meantime, applications that are deployed on top of the networks require different Quality-of-Service (QoS) levels. Some of them are more critical than others and have high availability requirements even in case of severe network impairments. QoS oriented constraints such as the level of trust in a network and its degree of reliability can also be part of the requirements of some of the critical applications.

In this thesis, different path computation algorithms were studied and implemented in a way that consider the criticality of the applications, -QoS- security and reliability requirements in the establishment of cost effective paths including the adequate restoration scheme.

1.2 Thesis Organization

The rest of the thesis is organized as follows. In chapter 2, the problem statement is explained. Then a literature survey on routing algorithms and resilience techniques is presented in chapter 3. In chapter 4, our approach is explained and analyzed and the evaluation results are presented and discussed in chapter 5.

2. PROBLEM STATEMENT

Resilience and reliability has been a traditional goal within telecommunication networks design as companies and organizations need to trust them for their everyday activities.

Multi-homing approach is a very good solution to the problem as it increases the reliability and it can be characterized as a first level of resilience. Using multi-homing, the sites are connected to many networks so that in case of failure a different network can be used to reach the destination. Companies and organizations, connect their sites in many Internet Service Providers, in order to avoid the lack of connectivity due to any disaster. Fig. 1 depicts our approach. We have considered three networks for multi-homing and partial connectivity between the sites.

Even if the above approach seems to be able to solve the problem, it is not 100% satisfactory as it requires full mesh connectivity, something that has a significant cost. As we can see in Fig. 1, when the enclaves are not interconnected in a full mesh way in each ISP, in case of failure there is not a guarantee of connection between those sites.

Three different user classes have been taken into account, for the case that different users require different QoS according to the criticality of their applications. This can be achieved by negotiations of SLAs (Service Level Agreement) for building a logical end-to-end service delivery infrastructure in top of the existing data transport networks. Companies and organizations negotiate these SLAs with the ISPs and set up the tunnels on the overlay networks. The SLA reserves an amount of bandwidth and provides a set of guarantee to the end user in terms of reliability, security, trust etc.

For that reason, we consider an overlay network which is based in a multi-homed environment. Using this kind of environment we are trying to engineer the available resources and use an effective protection and restoration scheme. We consider only the links, as we can see in Figure 2, and we try to set-up the paths in that overlay topology optimizing the robustness of the network.

Building robust networks requires to keep secondary paths which can serve the traffic whenever any impairment caused in the network. The reservation of one secondary path for every primary path is quite costly so in our work we do not focus only on the allocation of primary and backup paths but also in finding a cost effective technique to protect the paths. Moreover, primary and backup paths need to be diverse. This means that they must share as less links and nodes as possible avoiding correlated failures.

Last but not least, our goal is to find a minimum (cost wise) path between the sites as well as to assure that the capacity of the edges is not overloaded. The delay is also a parameter that must be considered for the delay-sensitive applications.

Trying to optimize many objectives and satisfy many constraints, our problem is a ***multi-constrained optimal path problem***. The mathematical formulation of the problem is given below:

We consider a network that is represented by a directed graph $G=(V,E)$*, where* V *is the set of nodes and* E *is the set of links. Each link* $(i,j) \in E$ *is associated with* A *additive QoS parameters* $w_a(i,j)$*, where* $a=1,2,...,A$ *and* B *boolean QoS parameters* $w_b(i,j)$*, where* $b=1,2,...,B$*. All parameters are not negative. Given* A *additive constraints* c_a*,* $a=1,2,...,A$*, and* B *boolean constraints* c_b*,* $b=1,2,...,B$ *we can define different the path computation problems:*

Definition 1:Multi-Constrained Optimal Path (MCOP) Problem: find a path p from a source node s to a destination node t such that:

1. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j) \le c_a, \forall\ a \in (1,2,...,A)$
2. $w_b(i,j) \le c_b, \forall\ (i,j) \in p\ \ \forall\ b \in (1,2,...,B)$
3. $w(p) = \sum_{(i,j)\in p} a * w_1(p) + b * w_2(p) + ... + M * w_A(p)$ is minimized over all feasible paths satisfying (1) and (2), $a = 1,2,...,A$

Definition 2: Restricted Shortest Path (RSP): find a path p from a source node s to a destination node t such that:

1. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j) \le c_a, \forall\ a \in (1,2,...,A)$
2. $w_b(i,j) \le c_b, \forall\ (i,j) \in p\ \ \forall\ b \in (1,2,...,B)$
3. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j)$ is minimized over all feasible paths satisfying (1) and (2), $a \in (1,2,...,A)$.

Definition 3: Multi Constraint Path (MCP): find a path p from a source node s to a destination node t such that:

1. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j) \le c_a, \forall\ a \in (1,2,...,A)$
2. $w_b(i,j) \le c_b, \forall\ (i,j) \in p\ \ \forall\ b \in (1,2,...,B)$

Objective Functions

For any set of pairs source – destination $(s_i, d_i) \in V, i = 1,...,N, N \ge 1$ the objective functions are:

1. Minimize the total cost of the paths (primary and backup) between source and destination

$$minimize \sum_{i=1}^{N} \text{cost(primary path } (s_i, d_i) + \sum_{i=1}^{N} \text{cost(backup path}(s_i, d_i)$$

2. Minimize the number of common edges between primary and backup paths (to increase the network survivability)

$$minimize \sum_{i=1}^{N} \text{no of common edges (primary path } (s_i, d_i), \text{backup path}(s_i, d_i))$$

Constraints

1. Capacity : resource availability insurance
2. Hop Limit : delay level insurance

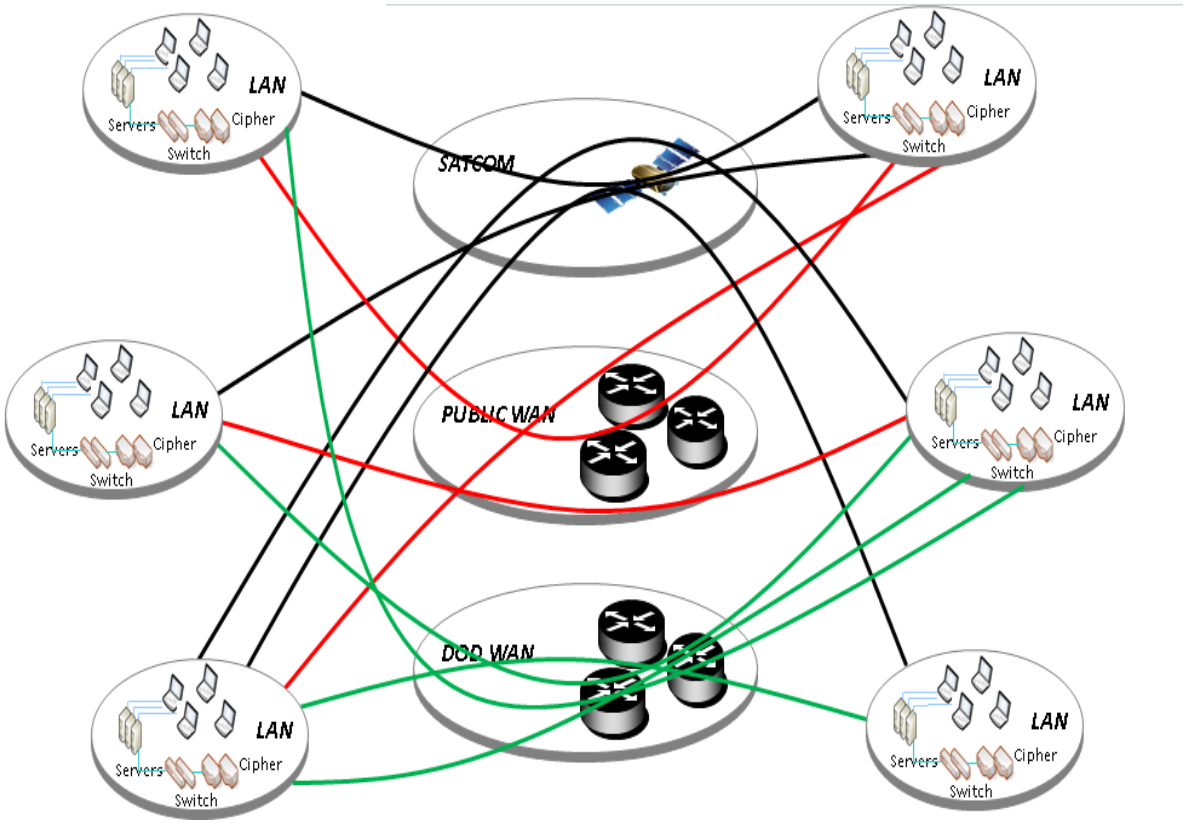

Figure 1: Multi-homing approach

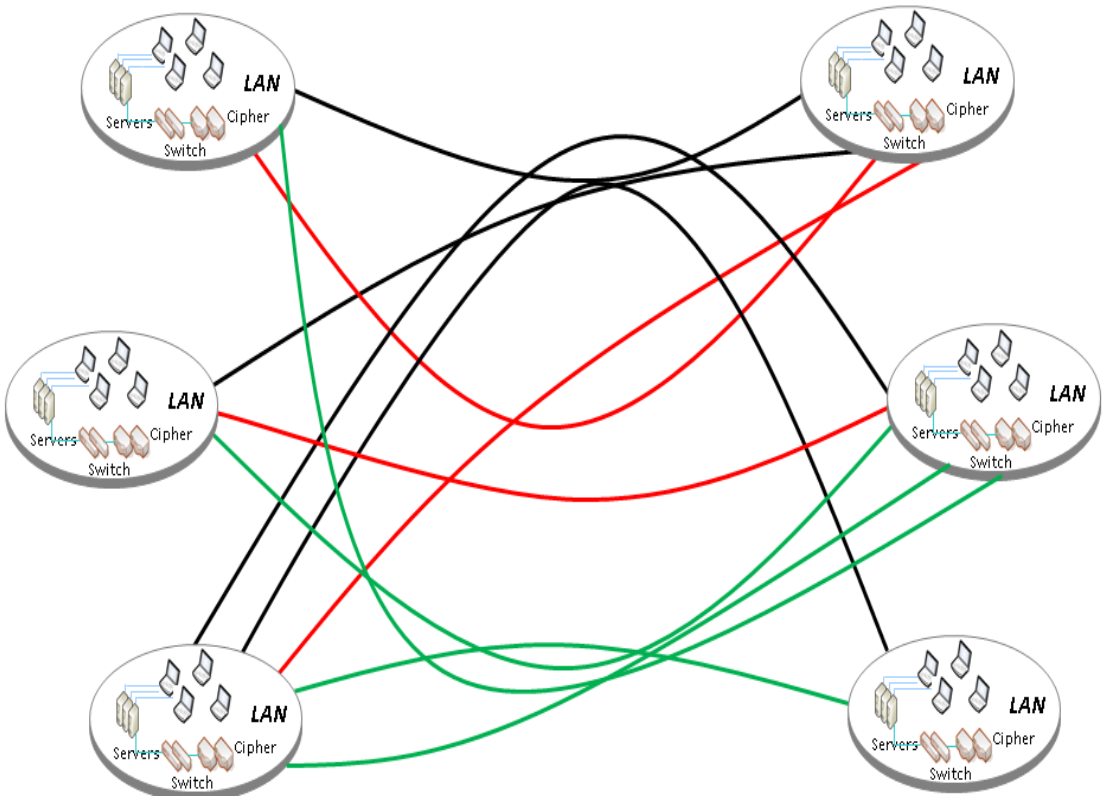

Figure 2: Overlay Network

3. LITERATURE SURVEY

In this chapter a small literature survey is presented. An important part of our work is the path computation algorithms. Since they are studied over 30 years and plenty of them can be found in the literature, a presentation of the most important algorithms is included in this work. Some different approaches of solving partially our problem can be found in 2.4. Finally, the different kinds of QoS metrics are explained.

3.1 QoS Metrics

The QoS metrics used in route computation can be distinguished in three categories:

- Additive,
- Concave
- Multiplicative metrics [1].

The additive and multiplicative metrics of a path is the sum and multiplication of the metric respectively for all the links constituting the path. The concave metric of a path is the maximum or the minimum of the metric over all the links in the path. This metric is usually dealt with a pre-processing step called topology filtering, where all the links that do not satisfy the constraint are pruned and not considered further in the path selection process. In case of optimizing these metrics, the path with the minimum metric is chosen.

Respectively, additive, multiplicative and boolean constraints exist. For the first two ones there is a value limit, so the metric of the path must satisfy this limit. The boolean constraints have been encountered by the same way.

Mathematically, the metrics can be represented as follows:

- Additive metrics (i.e. delay, jitter)

$$m(p) = \sum_{i=1}^{LK} lk_i$$

, where $m(p)$ is the total of metric m of path p, lk_i is the link in the path p and LK is the number of links in path p. Delay and jitter can be thought as examples of this class.

- Concave metrics (i.e. bandwidth)

$$m(p) = \min/\max(m(lk_i))$$

- Multiplicative metrics (i.e. reliability (error free transmission probability))

$$m(p) = \prod_{i-1}^{LK} m(lk_i)$$

3.2 Open Shortest Path First (OSPF)

The current internet is a best-effort network based on the routing protocol Open Shortest Path First (OSPF) using the shortest path approach. In the heart of OSPF is the Shortest Path First (SPF) algorithm, which is based on Dijkstra. This algorithm calculates the shortest path from a source to a destination. All the traffic is routed to the

shortest path and even if some alternate paths exist, they are not used as long as they are not the shortest ones. One drawback of this scheme is the congestion that can be occurs in some links, while some other links are not fully used.

3.2.1 Dijkstra's Algorithm

In this section, a brief overview of Dijkstra's algorithm is provided. If we assume a graph $G(V,E)$, where V is the set on nodes and E is the set of edges, all edge weights must be nonnegative, i.e., for all $(u,v) \in E$, we have $m(u,v) \geq 0$. In such a graph, Dijkstra's algorithm provides the optimal path(s) that solve(s) the single-source shortest-path problem. Internally, the algorithm assigns into a set S all the nodes whose final shortest path weights from the source s have already been calculated. The pseudo code of Dijkstra's algorithm is following:

1: procedure Dijkstra (G, m, s)

2: for all $v \in V$ do

3: $l[v] \leftarrow \infty$

4. $\pi[v] \leftarrow NIL$

5: end for

6: $Q \leftarrow V$

7: $l[s] \leftarrow 0$

8: while $Q \neq \varnothing$ do

9: $u \leftarrow Extract - Min(Q)$

10. $Q \leftarrow Q \setminus u$

11: for all node $v \in N(u)$ do

12: if $l[v] \succ l[u] \oplus m(u,v)$ then

13: $l[v] \leftarrow l[u] \oplus m(u,v)$

14: $\pi[v] \leftarrow u$

15: end if

16: end for

17: end while

18: end procedure

Mathematically, for all nodes $v \in S$, we have $l[v] = \pi^*(s,v)$. At each iteration, the node $u \in V - S \equiv Q$ that has the minimum shortest-path estimate $l[u]$ is inserted into S (and removed from Q). At the same time, all edges leaving u are relaxed. In the following, $N(v)$ denotes all the neighbors of V in G.

First, the algorithm performs the initialization of $l[v]$ and $\pi[v]$ values. Next, the set S is initialized to the empty set, and conversely the set Q is initialized to contain all the

nodes in G. In each iteration of the while loop, a node $u \in Q$ with the minimum shortest path estimate is extracted from $Q = V - S$ and moved into S. Finally, each edge (u,v) leaving u is relaxed. If the shortest path to v can be improved by going through u, the values of $l[v]$ and $\pi[v]$ are updated. The nodes only move from Q to S, not in the other direction. Therefore, since Q originally contains V nodes, the while loop is guaranteed to iterate exactly V times.

Running Dijkstra's algorithm on a graph $G = (V, E)$ with nonnegative weight function m and source s produces the shortest path weights $l[u] = \pi^*(s,u)$ for all nodes $u \in V$. The complexity of Dijkstra's algorithm is $O(V^2 + E) = O(V^2)$.

3.3 Constrained-Based Routing

As referred above, today's Internet can only provide "best-effort" service with no guarantees regarding loss rate, bandwidth, delay, delay jitter, etc. While this kind of service works fine for some traditional applications (such as FTP and email), there are many of them which require high bandwidth, low delay, and low jitter. The constrained-based routing is able to provide QoS.

The problems are coming up against multiple metrics and multiple constraints, with a sort of increasing order of complexity are [2]:

- **Multi Constrained Path (MCP)** [3] [4]: refers to the problem of finding a path through a network that satisfies all the constraints without considering any optimization.
- **Restricted Shortest Path (RSP)** [5]: refers to the problem of finding a path that satisfies all the constraints and is optimal to one objective only. RSP is also referred in the literature as Constrained Shortest Path First (CSPF).
- **Pareto** [1]: find one or more efficient solutions, possibly subject to M constraints. Pareto Optimal solutions are those that improvement in one objective can only occur with the worsening of at least one other objective. There usually exist several Pareto optimal solutions, which constitute the Pareto set.
- **Multi Objective Optimal Path** (MOOP) [2]: refers to the problem of finding the optimal path subject to M objectives. Even if the Pareto set gives many feasible solutions, the MOOP through some techniques is able to find one optimal solution, keeping a balance between the values of the M metrics.
- **Multi Constraint Optimal Path** (MCOP) [6]: refers to the problem of finding the optimal path subject to M objectives and M constraints.

The above problems have been proven to be NP-complete[7]. The definition of each problem follows:

Consider a network that is represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of links. Each link $(i, j) \in E$ is associated with A additive QoS parameters $w_a(i, j)$, where $a = 1, 2, ..., A$ and B boolean QoS parameters $w_b(i, j)$, where $b = 1, 2, ..., B$. All parameters are not negative. Given A additive constraints c_a, $a = 1, 2, ..., A$, and B boolean constraints c_b, $b = 1, 2, ..., B$ we can define the different kinds of path computation problems:

Definition 1:Multi-Constrained Optimal Path (MCOP) Problem: find a path p from a source node s to a destination node t such that:

4. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j) \le c_a, \forall\ a \in (1,2,...,A)$
5. $w_b(i,j) \le c_b, \forall\ (i,j) \in p\ \ \forall\ b \in (1,2,...,B)$
6. $w(p) = \sum_{(i,j)\in p} a * w_1(p) + b * w_2(p) + ... + M * w_A(p)$ is minimized over all feasible paths satisfying (1) and (2), $a = 1,2,...,A$

Definition 2: Restricted Shortest Path (RSP): find a path p from a source node s to a destination node t such that:

4. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j) \le c_a, \forall\ a \in (1,2,...,A)$
5. $w_b(i,j) \le c_b, \forall\ (i,j) \in p\ \ \forall\ b \in (1,2,...,B)$
6. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j)$ is minimized over all feasible paths satisfying (1) and (2), $a \in (1,2,...,A)$.

Definition 3: Multi Constraint Path (MCP): find a path p from a source node s to a destination node t such that:

3. $w_a(p) = \sum_{(i,j)\in p} w_a(i,j) \le c_a, \forall\ a \in (1,2,...,A)$
4. $w_b(i,j) \le c_b, \forall\ (i,j) \in p\ \ \forall\ b \in (1,2,...,B)$

3.3.1 Multi Constrained Path (MCP)

In MCP, path computation must be performed under the constraint that multiple QoS requirements have to be jointly satisfied. In most of the cases the search is reduced to the minimum-cost path with respect to only one constraint. The main drawback of this approach is that, since only one path is found, if that path does not satisfy all the constraints the flow is not admitted to the service.

Some algorithms can be found in the literature are:

3.3.1.1 Jafee's Algorithm [8]

Jaffe's algorithm solve the MCP problem under two constraints ($m = 2$). For each link $(u,v) \in E$, the algorithm assigns a composite weight $w(u,v)$ that is obtained by linearly combining the original weights w_1 and $w_2 : w(u,v) = d_1 * w_1(u,v) + d_2 * w_2(u,v)$, where d_1 and d_2 are positive multipliers. The algorithm then returns the path that minimizes the w weight. The minimization process is illustrated pictorially in Figure 3. In this figure, all possible paths between the source and destination nodes are indicated by black circles. Equal-length paths w.r.t. the composite weight w are indicated by a line. The search for the minimum length path is equivalent to sliding this indication line outward from the origin until a path is hit. This path is the one returned by the algorithm. The figure also shows that the returned path does not necessarily reside within the feasibility area defined by the constraints. In fact, Jaffe proposed using a nonlinear function whose

minimization guaranteed finding a feasible path. But there is no shortest path algorithm to minimize such a nonlinear function. Instead, Jaffe provided this algorithm and showed how to determine d_1 and d_2 based on this nonlinear function.

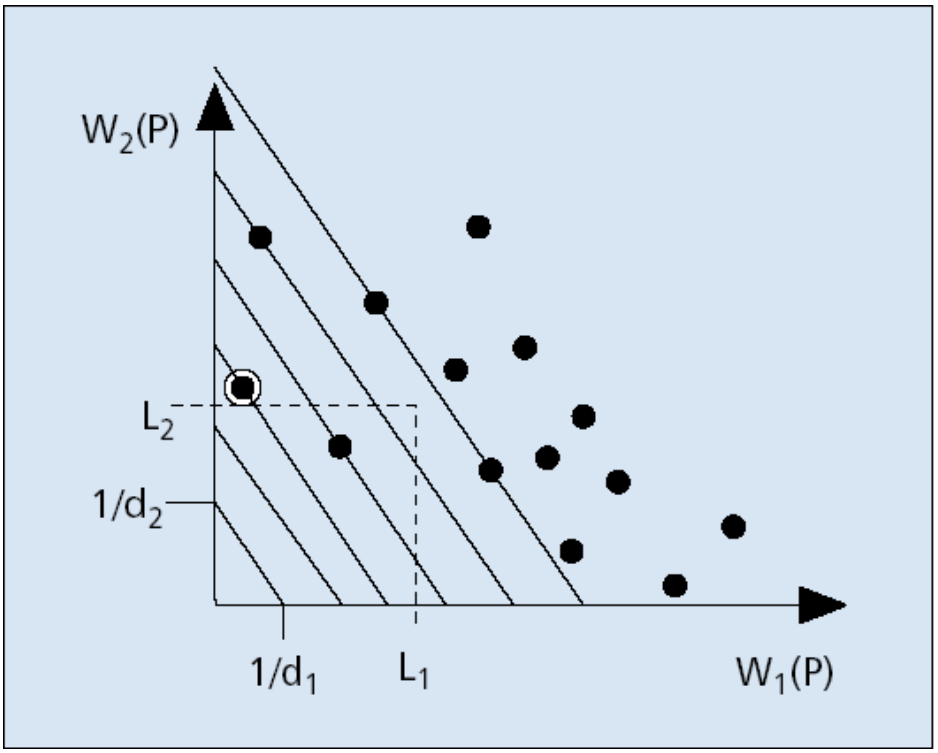

Figure 3: Search process in Jaffe's algorithm

3.3.1.2 Chen's Algorithm [9]

Chen proposed an algorithm for MCP problem with a polynomial time complexity. With this algorithm, first the NP-complete is reduced to a basic one, which can be solved in polynomial time, and then it is solved by using Dijkstra's or Bellman-Ford algorithm. For a directed graph $G(V,E)$, from source vertex s, to destination vertex t, two weight functions which are additive $w_1 : E \to R^+$ and $w_2 : E \to R^+$, two constants $c_1 \in R^+$ and $c_2 \in R^+$ are defined. The problem is described as for $MCP(G,s,t,w_1,w_2,c_1,c_2)$ finding a path p from s to t where $w_1(p) \le c_1$ and $w_2(p) \le c_1$.

In multi-constrained routing, for example, two constraints delay and bandwidth can be taken into account as two weight functions. Given a source node s and a destination node t, delay and bandwidth constrained routing problem is to find a path p from s to t such that delay $(p) \le D$ and bandwidth $(p) \le B$ where D and B are required to have end to end delay and bandwidth bounds respectively.

3.3.1.3 Extended Dijkstra's (ED) [10]

This heuristic algorithm focuses on those cases when the shortest path does not meet all the constraints but at least one sub-optimal path does, and this sub-optimal path often re-uses the shortest path to one of the destination's neighbor nodes.

More in detail, ED starts by running Dijkstra's algorithm with respect to a particular metric $c^* \in \{C\}$, where C is the set of the considered M metrics. This yields the

shortest path P between s and t. If P satisfies the constraints on all the metrics, then the solution is found and ED ends. Otherwise, ED considers the one-hop neighbors of t. For each neighbor $w_j (j = 1,...,D)$, it computes the shortest paths P_j between s and w_j, and then it builds the complete s-t path by concatenating P_j with the link $(w_j, t): P' = P_j \oplus (w_j, t)$. If P' is feasible, the algorithm stops, otherwise it repeats the same operations with the next neighbor, until a feasible path is found (success) or all neighbours have been examined (failure).

The complexity depends from the average network degree, which is constant with the number of nodes.

A disadvantage is that the shortest path to every neighbor must be available. This can be achieved by running a Dijkstra's shortest path for each examined neighbor, but actually the simplest way is to let the first Dijkstra's shortest path run to its completion, i.e. find the whole tree of shortest paths, instead of stopping when the destination node is reached (or at least stopping when all the neighbors are reached). In computational terms, this does not adds much overhead.

The final time complexity of ED can therefore be written as $O(DSP + MD)$, where D is the network degree. The space complexity is the same of DSP plus D_n to store the extra paths.

3.3.1.4 Multi-Metric Extended Dijkstra's (ED) [10]

The shortest path according to a given metric may not be the same if the other metrics are considered. If the search using the considered metric does not yield a positive result (i.e. all involved metrics are satisfied), then DSP is repeated with another measure until a feasible path is found or all metrics are examined. This allows finding up to M different paths, which are optimum with regard to at least one constraint.

The main drawback of this approach is that the maximum number of potential paths, and hence the probability of success, is equal to the number of metrics M.

The complexity of this algorithm can be straightly evaluated as $O(M * O(DSP))$, where $O(DSP)$ is the complexity of Dijkstra's shortest path algorithm.

3.3.2 Restricted Shortest Path (RSP)

The RSP problem is also known in the literature as a Constraint Shortest Path First (CSPF) problem. By its name, CSPF is an extension to the traditional shortest-path algorithm with a set of constraints attached. It tries to satisfy these constraints and optimize one metric.

3.3.2.1 A*Prune [11, 12]

The A* algorithm is one of the most famous algorithms and is widely used in path finding and graph traversal, the process of plotting an efficiently traversable path between points, called nodes. A* achieves a very good performance by using heuristics. Each node has a heuristic cost function which is the combination of the path cost

function, the cost from the starting node to the current node x, and the heuristic estimate of the distance from x to the goal. *Manhattan Method* (Fig. 4) is a very good example of finding the heuristic cost in an environment separated by squares, but several methods exist. By this way the algorithm reaches faster to the destination.

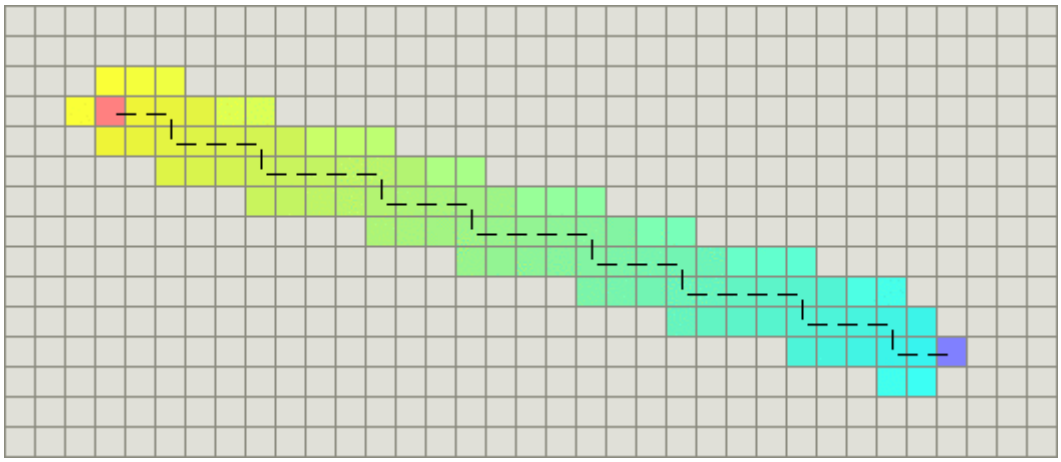

Figure 4: Manhattan Method

In terms of networking, the A*Prune is capable of finding feasible paths satisfying a set of either additive or boolean constraints and the same time optimizing one metric which is called Traffic Engineering metric.

The algorithm consists of two major steps: pre-computation and path expanding/pruning. To deal with the additive constraints, it performs a pre-computation of their associated Dijkstra distances from node v to t, $D_r(v,t)$. This pre-computation can be done in the background and stored for use by multiple path computations as long as the topology remains unchanged.

In expanding/pruning stage, A* Prune algorithm, keeps a priority queue of paths. The priority queue keeps the feasible paths found up to now. Initially, it contains the path $p(s,s)$, which consists of the single node s. Looking at the shortest path $p(s,u)$ contained in the queue, the algorithm adds the path to the list of feasible paths CSP_List if the path already reaches the destination node t. Otherwise, it attempts to expand the path using each $u's$ outgoing links (u,v) to generate a path $p(s,v)$. It prunes the path either when a loop is detected or a boolean constraint is violated. For an additive constraint, it combines the associated Dijkstra distance to node $t-D_r(v,t)$ obtained in pre-computation and the path's constraint value $W_r(p(s,v))$ to have an estimate of the projected distance and compares it against the constraint C_r. If the projected distance is greater than the constraint, any path expanded beyond this point will violate the constraint such that $p(s,v)$ can be safely pruned or removed from further consideration. This process continues until sufficient number of paths has been found for the objective.

The pseudocode of the algorithm follows:

Input:

1: $G=(V,E)$, a graph with node set V and edges set E.

2: (s,t): a node pair with source s and destination t.

3: K : number of paths to be found.

4: Ra : number of additive constraints.

5: Rb : number of Boolean constraints.

6: $C_a(i)$: the i_{th} additive constraint.

7: $C_b(i)$: the i_{th} Boolean constraint.

8: $w_k(e)$: weight related to k_{th} constraint associated to link $e \in E$.

9: $m(e)$: TE metric of link $e \in E$.

Precomputation:

10: $for \quad \forall v \in V$ and $\forall r \in (1,2,...,R_a)$, compute:

11: $D_r(v,t)$: length of Dijkstra path from v to t associated with r_{th} additive constraint.

Initialization:

12: $k = 0;$ // number of feasible paths found so far.

13: $W_r(p(s,s)) = 0; 1 \le r \le R_a$ //path's current constraint value, // used for additive constraint pruning

14: $M(p(s,s)) = 0;$ // path's TE metric, objective function

15: $pathQueue = \{p(s,s)\};$

16: $CSP\_list = \{\};$

Expanding and Pruning:

17: $while((k < K) and (pathQueue \neq null))\{$

18: $p(s,u) = extract\_\min(pathQueue);$

19: $if(u == t)\{$

20: $insert \ p(s,u) \ into \ CSP\_list;$

21: $k = k + 1;$ //find feasible path, add it to the list

22: $continue;$ // and continue with next path

23: }

24: $foreach \ u's \ outgoing \ edge \ e = (u,v)\{$ //Go through each outgoing link…

25: $if(v \in p(s,u))\{$// Check the loop existence and prune

26: $continue;$

27: }

28: $for(i = 1; i \le R_b; i++)\{$ //Boolean constraint pruning

29: *if* $(test(C_b(i),(u,v)) == fail)$ {
30: *break*;
31: }
32: }
33: *if* $(i \le R_b)$ {
34: *continue*;
35: }
36: $p(s,v) = append(p(s,v),(u,v))$; // Expand the path
37: *for* $(i = 1; i \le R_a; i++)$ { // Update the additive constraints
38: $W_i(p(s,v)) = W_i(p(s,u)) + w_i(u,v)$;
39: }
40: $M(p(s,v)) = M(p(s,u)) + m(u,v)$; // Update TE metric
41: *for* $(r = 1; r \le R_a; r++)$ { //Additive constraint pruning
42: *if* $(W_r(p(s,v)) + D_r(v,t) > C_r)$ {
42: *break*;
43: }
44: }
45: *if* $(r \le R_a)$ {
46: *continue*;
47: }
48: $insert\_queue(pathQueue, p(s,v))$; // Add the new path in pathQueue
49: }
50: }

The worst time complexity of A* Prune is O(dQ(R+h+logQ)), where Q is the number of expanded paths, R is the number of constrained metrics and h the maximum hops of the K shortest paths. The complexity grows exponentially with the size of the network.

3.3.3 Multi Objective Optimal Path (MOOP)

Multi-objective optimization problems deal with the presence of different conflicting objectives. Given that it is not possible to obtain a single solution by optimizing all the objectives simultaneously, a common way to face these problems is to obtain a set of efficient solutions called the non-dominated frontier.

The idea of non-dominating paths is based on the fact that if a path provides a better value in at least one objective parameter than the other paths found so far, it can not been rejected. Precisely, if we consider two paths P1 and P2 between the source and an intermediate node, for an m=2 objective problem with values (x1, y1) and (x2, y2) respectively, if x1<x2 and y1<y2, this means that y2 dominates y1. If x1<x2 and y1>y2 or x1>x2 and y1<y2, this means that P1 and P2 are non-dominated paths.

The most of the MOOP algorithms let the user to decide the most appropriate path between all the non-dominated ones according to the needs. It is worth referring that it is more useful to use the solutions can be found in the middle of the curve than the others in the end, since the middle solutions provide a balance between the metrics are being optimised.

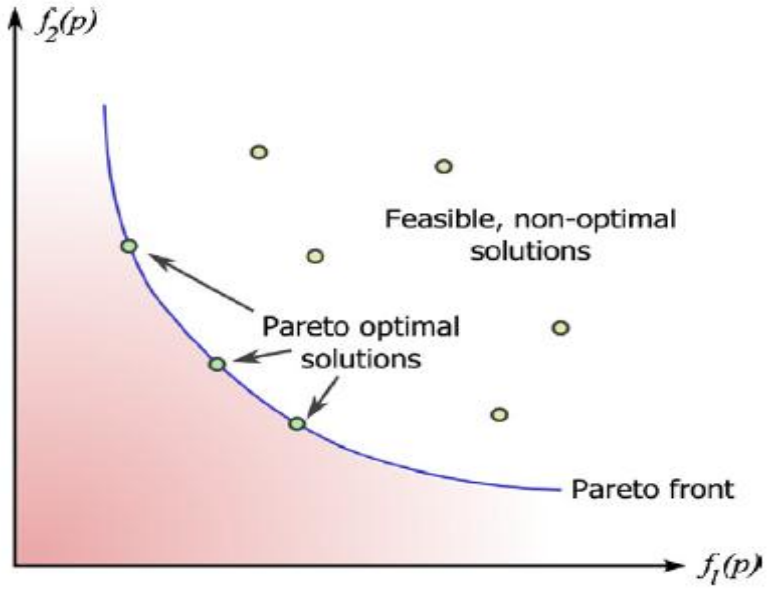

Figure 5:Pareto Optimal Solutions

3.3.3.1 Skriver and Andersens's LC algorithm [2]

Skriver and Andersens's LC algorithm is based in the non-dominated paths approach and selects all the Pareto optimal solutions. It is also based on Dijkstra with the difference that each node keeps all the non-dominated sub paths until reaching the destination. It can be used for M objectives.

Let Λ_u be the set of non-dominated labels at node u and K_u be its size. Each label is a $(c^u_{1,k},...,c^u_{M,k}), k \in [1, k_u]$, representing the cost of each metric along the currently non-dominated k_u paths from s to u. We assume that the labels are sorted so that $c^u_{1,k} < c^u_{1,k+1}$ and $c^u_{2,k} < c^u_{2,k+1}$, $\forall k, u$. In other terms $c^u_{1,1}$ is the smallest value for the first metric and c^u_{1,k_u} is the biggest, whereas the ordering on the second metric is reversed. Let now examine an edge (u, v) and evaluate whether reaching node v through node u improves any of the existing labels at node v.

Let consider the conditions:

$$c^u_{2,k_u} + c_2(u,v) \geq c^v_{2,1} \wedge c^u_{1,1} + c_1(u,v) \geq c^v_{1,1}$$

$$c^u_{1,1} + c_1(u,v) \geq c^u_{1,k_v} \wedge c^u_{2,k_u} + c_2(u,v) \geq c^v_{2,k_v}$$

If any of them is satisfied, then all labels of the new set $\Lambda_u + c(u,v)$ are dominated by the existing Λ_v. This occurs because either the first or the last label of Λ_v dominates all labels of the new set. In such a case, edge (u,v) can be removed from the graph, since it gives rise to dominated paths only. Otherwise, the labels of the two sets are merged and the dominated labels are removed.

A graphical example is reported in Fig. 6 in which all labels of $\Lambda_u + c(u,v)$ are dominated by the last label of Λ_v, $(c^v_{1,k_v}, c^v_{k_v})$. The above observation is applied to M=2 objectives but it can be extended easily for M>2.

Let $C_j \min$ and $C_j \max$ be the minimum and maximum value of each metric in the label set of node u, where $j \in [1, M]$. These values can be used to create a set of virtual labels Λ_u*that represents the extreme points of the original set Λ_u. For example, a possible virtual label could be $(c_1^{u,\min}, ..., c_{M-1}^{u,\min}, c_M^{u,\max})$. Clearly, when $M = 2$, the virtual labels corresponds to the actual extreme labels, hence the bi-dimensional case is also included in this extension.

If the set Λ_u* is dominated by the set Λ_v*, then all labels in $\Lambda_u + c(u,v)$ are also dominated by any label in Λ_v. This can be seen in Fig 6 (right), which illustrates the case in which the third metric is the one that determines the domination of Λ^*_v over all labels in $\Lambda^*_u + c(u,v)$. For each virtual label we can write a set of M equations:

$$c_i^{u,\min} + c_i(u,v) \geq c_i^{v,\max}, \forall i \in I^{\max},$$

$$c_k^{u,\min} + c_k(u,v) \geq c_k^{v,\min}, \forall k \in I^{\min},$$

Where I_{max} and $I_{\min}$ are two subsets of indices that indicate the metrics for which the considered virtual label of v assumes, respectively, the maximum and minimum values.

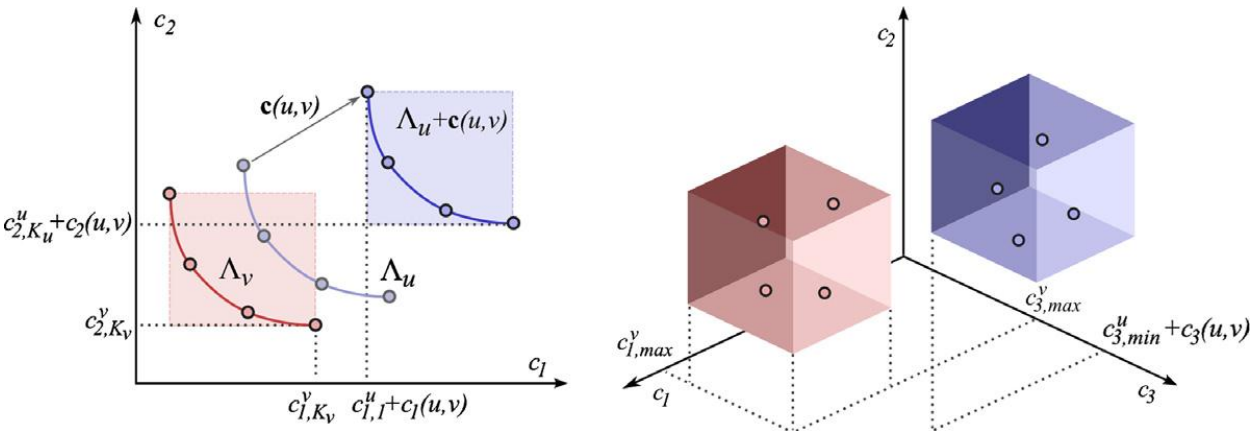

Figure 6: Exemplification of Skriver and Andersen's LC algorithm for the bi-dimensional (left) and three-dimensional (right) cases. The small circles are the current labels, the lines are the Pareto

fronts, and the shadowed areas are the hypercubes determined by the extreme points of the label sets.

The pseudocode of the final Skriver and Andersen's algorithm for M objectives is given below:

1: $\Lambda_s = \{0,0\}$

2: $X = \{s\}$

3: $while X \neq 0 do$

4: $u = extract\ \ first\ element\ \ from X$;

5: $X = X \setminus \{u\}$

6: $for\ \ all (u,v) \in E do$

7: $\quad if\ (\Lambda_v = null)\{$

8: $\quad\quad \Lambda_v = \Lambda_u + c(u,v)$

9: $\quad\quad continue$

10: $\quad else$

11: $\quad\quad if \exists label\ in\ \Lambda_v^*\ such\ that\ Eq.(1) is\ satisfied\ then$

12: $\quad\quad\quad remove(u,v)\ \ from E$

13: $\quad\quad\quad continue$

14: $\quad\quad endif$

15: $\quad endif$

16: $\Lambda_{tmp} = Merge(\Lambda_v, \Lambda_u + c(u,v))$

17: $\quad if\ (\Lambda_{tmp} \neq \Lambda_v) then$

18: $\quad\quad \Lambda_v = \Lambda_{tmp}$

19: $\quad\quad X = X \cup \{v\}$ (only if $v \notin X$, to avoid duplicates)

20: $\quad endif$

21: $\quad endfor$

22: $\quad endwhile$

23: Λ_t Holds the set of non-dominated paths from s to t

In the pseudo code, X is the set of labeled nodes to be checked for dominance, and is handled with a FIFO discipline. The dominance check is concisely reported at line 11. In case no complete dominance is found, the function Merge() at line 16 merges the two sets of labels and removes the dominated labels from the combined set. In accordance to the typical LC strategy, every time a label set changes, the correspondent node must

be reconsidered in the next iteration (lines 17-20).

The overall complexity of the algorithm is $O(DM[2^M \omega + \omega^2 \log(\omega)])$, which means that it depends on the specific implementation and problem instance.

3.3.4 Multi Constraint Optimal Path (MCOP)

The MCOP approach aims to find solutions which minimize a cost function and satisfy the set of constraints.

Minimizing a set of metrics is a scalar concept and it must be applied to the multi-dimensional space which is not straight-forward. For instance, the intuitive interpretation of asking for the simultaneous optimisation of all the objective functions:

$$\min_{p \in \omega} \begin{bmatrix} f_1(p) \\ f_2(p) \\ . \\ . \\ . \\ f_M(p) \end{bmatrix}$$

is rarely feasible. Typically, the multiple objectives are conflicting with each other and there is no single global solution. As a consequence, the trivial extension of the contemporary and independent minimisation of each $f_j(p)$, the ideal objective vector is impossible. One solution is to export a set of optimal solutions (Pareto), but it is often convenient to further simplify it using some optimization techniques. Two of them can be found in [2]:

- **Convex sum**: linear aggregation of all metrics and then searching for a path p such that:

$$F(p) = \sum_{j=1}^{M} a_j f_j(p), a_j > 0, \forall j .$$

This method gives a particular trade-off solution, called supported efficient solution, and the weights a_j indicates the importance of the particular objective function.

- **Lexicographic order**: prioritisation of the objectives, each objective function is assigned a priority that determines the order in which the functions are optimised. Assuming

$$\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_M\}, \ \gamma_i \in [1, M], \gamma_i \neq \gamma_j \forall i \neq j,$$

defines an order in the evaluation of the objective functions, a path p is said to be lexicographically better that a path q with respect to the Γ ordering.

3.3.4.1 Heuristic MCOP (H_MCOP) [6]

H_MCOP is based on Dijkstra algorithm and tries to find a feasible path subject to K additive constraints and, simultaneously, minimizing the cost of that path. It focus only on additive QoS link parameters because the non-additive ones can be easily dealt with a pre-processing step by pruning all links that do not satisfy the constraints. The optimality requirement can be imposed through a primary cost function (administrative

weight, hop count), according to which the selected feasible path is optimal.

H_MCOP considers the following function for any path p from the source to the destination:

$$g_\lambda(p) = (\frac{w_1(p)}{c_1})^\lambda + (\frac{w_2(p)}{c_2})^\lambda + ... + (\frac{w_K(p)}{c_K})^\lambda \quad (1) \quad \text{where } \lambda \geq 1.$$

Firstly, it tries to minimize the objective function g_λ for $\lambda > 1$ to ensure the feasibility part. In doing so, it first exactly finds the best path w.r.t. g_1 from each node u to t. It then starts from s and discovers each node u based on the minimization of $g_\lambda(p)$, where p is a complete $s-t$ path passing through node u. This $s-t$ path is heuristically determined at node u by concatenating the already travelled segment from s to u and the estimated remaining segment (the above best path w.r.t. g_1) from u to t. Since the algorithm considers complete paths, it can foresee several paths before destination. For the optimality part, if some of these foreseen paths are feasible, H_MCOP selects the one that minimizes the primary cost function rather than the one that minimizes the nonlinear cost function. Using this preference rule (i.e., minimize the primary cost function if the foreseen path is feasible; otherwise, minimize the nonlinear cost function), H_MCOP can be implemented as simple as single-objective algorithms.

Pseudocode of H_MCOP

$H\_MCOP(G=(V,E), s, t, c_k, k=1,2,...,K)$

1 $\text{Re}verse\_Dijkstra(G=(V,E),t)$;

2 *if* $r[s] > K$ *then*

3 *return failure* // there is no feasible path

4 *endif*

5 $Look\_Ahead\_Dijkstra(G=(V,E),s)$;

6 *if* $G_k[t] \leq c_k \forall k = 1,2,...,K$ *then*

7 *return the path* // a feasible path is found

8 *endif*

9 *return failure*

$Look\_Ahead\_Dijkstra\_\text{Re}lax(u,v)$

1 *Let tmp be a temporary node*

2 $c[tmp] := c[u] + c(u,v)$

3*a* *if* $(\lambda < \infty)$*then*

$$g[tmp] := \sum_{k=1}^{K} (\frac{G_k[u] + w_k(u,v) + R_k[v]}{c_k})^\lambda$$

3*b* *if* $(\lambda = \infty)$*then*

$$g[tmp] := \max\{\frac{G_k[u] + w_k(u,v) + R_k[v]}{c_k} | 1 \le k \le K\}$$

4 $G_k[tmp] = G_k[u] + w_k(u,v)\ fork = 1,2,...,K$

5 $R_k[tmp] = R_k[v]\ \ fork = 1,2,...,K$

6 $if\ (\Pr efer\_the\_best(tmp,v) = tmp) then$

7 $c[v] := c[tmp]$

8 $g[v] := g[tmp]$

9 $G_k[v] := G_k[tmp]\ fork = 1,2,...,K$

10 $\pi_g[v] := u$

11 *endif*

$\Pr efer\_the\_best(a,b)$

1 $if\ \ c[a] < c[b]\ \ and\ \ \forall k\ \ G_k[a] + R_k[a] \le c_k\ \ then\ return(a)$

2 $if\ \ c[a] > c[b]\ \ and\ \ \forall k\ \ G_k[b] + R_k[b] \le c_k\ then\ return(b)$

3 $if\ \ g[a] < g[b]\ \ then\ return(a)$

4 $return(b)$

The time complexity of H_MCOP is is equal to that of Dijkstra's, since at most two modified versions of Dijkstra's algorithm are executed with the complexity of $O(n\log(n) + m)$.

3.3.4.2 Tamcra and Samcra [13]

Tamcra is based on three fundamental concepts: (1) a nonlinear measure for the path length, (2) the k-shortest path approach, and (3) the principle of non-dominated paths. The first concept can be explained pictorially using Figure 7 (with m=2). Part (a) of the figure depicts the search process using a linear composition function, similar to the one used in Jaffe's algorithm. If the two path weights are highly correlated, them the linear approach tends to perform well. However, if that is not the case, then a nonlinear function is more appropriate.

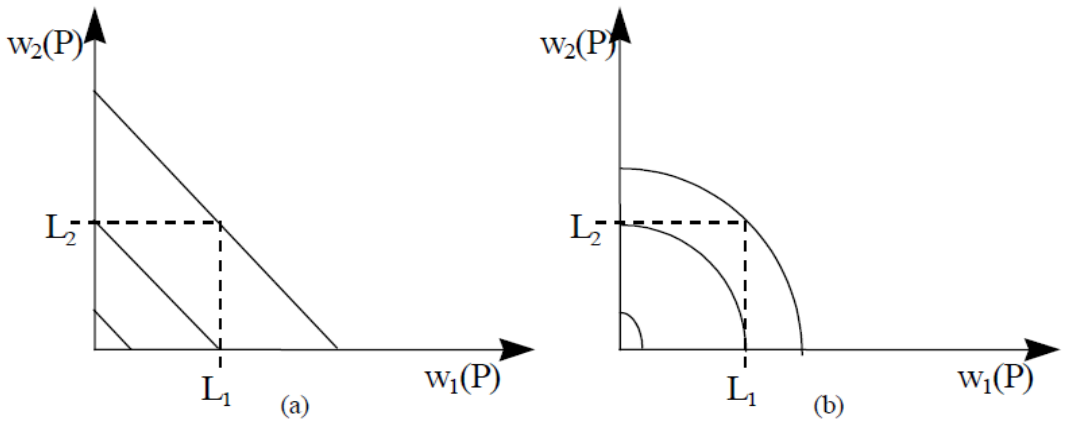

Figure 7: Searching for a feasible path by minimizing: (a) a linear composite function, (b) nonlinear composite function

Part (b) of the figure depicts the search process using a nonlinear function. Ideally, the equal-length lines should perfectly match the boundaries of the constraints, scanning the constraint area without ever selecting solution outside the constraint area. This can be achieved by taking:

$$l(P) = \max_{1 \le i \le m} \left(\frac{w_i(P)}{L_i} \right)$$

where $w_i(P) = \sum_{(u,v) \in P} w_i(u,v)$. Any path P that satisfied $l(P) \le 1$ is a feasible path, and hence is an acceptable solution to the MCP problem. The obtained path, however, may not be optimal in terms of its length.

An important characteristic of nonlinear path-length functions such as the one in (1) is that sub-paths of shortest paths are not necessarily shortest paths. In the path computation, this suggests considering more paths than only the shortest one, leading us to the k-shortest path approach.

The pseudocode of TAMCRA is given below:

TAMCRA(G, s, d, L, k)

G : graph, s : source, d : destination, L : constraints, k : tunable k- parameter

1: counter=0, for all nodes

2: length $(s[1]) = 0$

3: ADD $s[1]$ to queue

4: while (queue $\neq$ empty)

5: $u[i] =$ EXTRACT_MIN from queue

6: if ($u =$ destination) $\rightarrow$ STOP

7: else

8: for each $v \in$ adjacent_list(u)

9: if($v \neq$ previous node of $u[i]$)

10: PATH= $u[i]+(u,v)$

11: LENGTH = length of PATH

12: check if PATH is non-dominated

13: if(LENGTH $\leq$ and non-dominated)

14: if (counter(v) $<$k)

15: counter(v) = counter(v) + 1

16: j = counter

17: $v[j]=$counter(v)

18: length($v[j]$) = LENGTH

18: ADD $v[j]$ to queue

20: else

21: add $v[j]$ = path in queue with maximum length to v

22: if (LENGTH < length (old $v[j]$))

23: new $v[j]$ = PATH

24: REPLACE in queue old $v[j]$ with new $v[j]$

The worst-case complexity of TAMCRA is $O(kN\log(kN)+k^2mE)$. The allocated buffer space k is predefined and fixed, and therefore its worst case complexity is polynomial.

3.4 Resilience Techniques

As more and more mission critical services emerge on the Internet, there is a growing demand for the Internet to provide availability and reliability. More and more domains adopt multi-homed connection to expect increasing reliability. In [14] a fast recovery technique called Fast Reroute upon Multi-homed Domains (Fremd) is proposed. A Fremd path is pre-established between the providers of multi-homed domains. When links fails, the provider could fast react to failure with the help of alternative path which is pre-established. When the link failure is not restored for a certain period of time, normal BGP (Border Gateway Protocol) convergence is invoked. Once the network converge to another stable state, the Fremd path is withdrawn and the network come back to normal forwarding.

A more generic approach is given by [15] proposing harmonized set of capabilities for site multi-homing, traffic engineering, end-to-end security and support for mobile systems and networks.

Multi-homing is a good solution to achieve a minimum level of resilience but when two sites are not directly interconnected (e.g through an LSP) , another edge / node that has a direct link with the source and destination can neither not be used to maintain the connectivity. So, we can not simply rely on this technique.

Other restoration approaches can be found in the literature combined with path computation algorithms. In [16] a set of Pareto solutions are found, the primary and

backup path are selected randomly. Trying to find suitable backup path some improvements are applied directly to the solution or two (or more) solutions are combined in order to get better ones and to obtain some diversity. For the first case, a solution can be improved in one of the following ways:

1. Rebuild the existing path (primary or backup) from a given node.
2. Replace an existing path with another one from the set of paths generated in the beginning of the search process.

For the second case, the two solutions which are to be combined need to have a common node different from the first and second one.

In [17], the path diversification mechanism is presented that can be used to select multiple paths between a given ingress and egress node pair using a quantified diversity measure to achieve maximum flow reliability.

Since the primary motivation for implementing the path diversification mechanism is to increase resilience, paths should be chosen such that they will not experience correlated failures. To this end, a measure of diversity that quantifies the degree to which alternate paths share the same nodes and links is introduced in [17].

Definition 1 (Path): Given a (source s, destination d) node pair, a path *P* between them is a vector containing all links *L* and all intermediate nodes *N* traversed by that path

$$P = L \cup N$$

and the length of this path |*P*| is the combined total number of elements in *L* and *N.*

Definition 2 (Path diversity): Let the shortest path between a given (s,d) pair be P_0. Then, for any other path P_k between the same source and destination, we define the diversity function $D(x)$ with respect to P_0 as:

$$D(P_k) = 1 - \frac{|P_k \cap P_0|}{|P_0|}$$

The path diversity has a value of 1 if P_k and P_0 are completely disjoint and a value of 0 if P_k and P_0 are identical. For two arbitrary paths P_a and P_b the path diversity is given as:

$$D(P_b, P_a) = 1 - \frac{|P_b \cap P_a|}{|P_a|}$$

, where $|P_a| \leq |P_b|$.

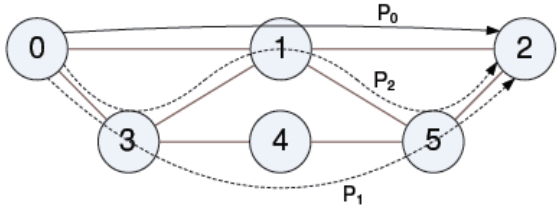

Figure 8: Shortest Path P_0 and alternatives P_1 and P_2

Given a failure of node 1, both Po and P2 will fail. P1 on the other hand has both a novelty of 1 and a diversity of 1, and does not share any common point of failure with Po.

Finally, in [18] a new restoration path computation algorithm is introduced called “Shortest Backup” method. This method can be used when paths are pre-computed, but not pre-established. The basic idea is to select strictly shortest paths for the working and restoration paths as well. Common links between the two paths are allowed, to achieve high efficiency. In order to prepare for the failure of the common links between the primary path and the first (shortest) restoration path, an additional restoration path is computed for a working path.

The above approaches can partially solve the problem of the present work but the restoration schemes are not cost effective because of the capacity reservation of two or three paths.

4. THE ADOPTED APPROACH

4.1 Studied Scenario

Since reliability and resilience can be enhanced by multi-homing, our work is based in a multi-homing context as a first level of resilience. We consider many different sites, each of which is a Local Area Network (LAN), connected at least to 2 different available Wide Area Networks (WAN), which is the usual situation for almost all the organizations. Each of WAN is replaced by links with specific characteristics in terms of availability, security, trust and protection.

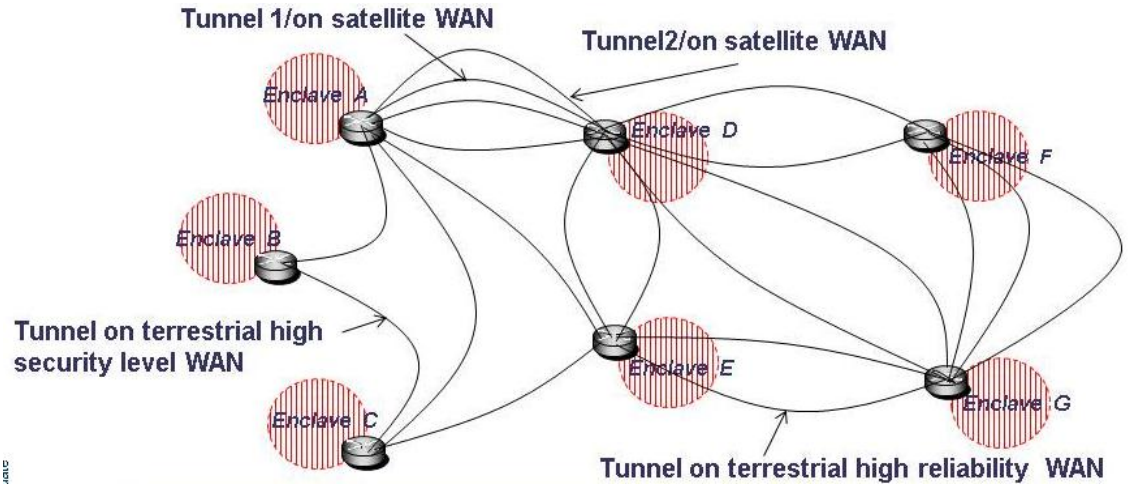

Figure 9: Overlay View

An overlay network is considered which tries to address the end-to-end quality of service (QoS), and to facilitate the deployment of value-added Internet services and QoS-sensitive applications. The initial (pro-active) routing can be pre-computed and possible SLAs (Service Level Agreement) can be negotiated by organizations or enterprises with the network access providers for building a logical end-to-end service delivery infrastructure in top of the existing data transport networks. According to the SLA, an amount of bandwidth is reserved and a set of guarantee is provided to the end user (e.g. reliability of five 9 = 99.999%).

The overlay network of the present work is built by the reserved paths have been defined on the different networks. These paths are based on the end user requirements, the user class constraints and the need of the high reliability for critical applications. As we can see in Fig. 9 some links provide high reliability, other links high security etc.

Different user classes are considered (Fig. 10) to deal with differentiated resilience according to the QoS demand. The paths are not designed only on a source – destination basis but in the user criticality basis. This means that depending the level of their criticality, two different end users can use different paths to research the same destination and also different protection scheme can be applied to their traffics. Differentiated resilience depends on:

- Bandwidth guarantee with respect to recovery
  - Hard Reservation

- Soft Reservation
- No Reservation

- Link characteristics
- Recovery time

Because of the offline path computation, in our work we cannot consider the recovery time.

Full recovery is the most expensive approach as it provides hard reservation of capacity. Full capacity provisioning is done in both primary and backup paths. This means that in case of failure, there are always enough resources to pass the traffic through the backup path.

Partial recovery is similar to full recovery in that full capacity provisioning is done for the primary paths but the backup paths share the same resources. The possibility to have two failures the same time is low, so if the backup paths use the same links, the capacity provisioning is done only once, so that a lot of bandwidth is released. This kind of recovery has quite high performance and reasonable cost.

No recovery means no backup paths and so no capacity reservation. This approach has the lowest cost and is the one is used today. By this class differentiation, a quite good balance between QoS and cost is achieved.

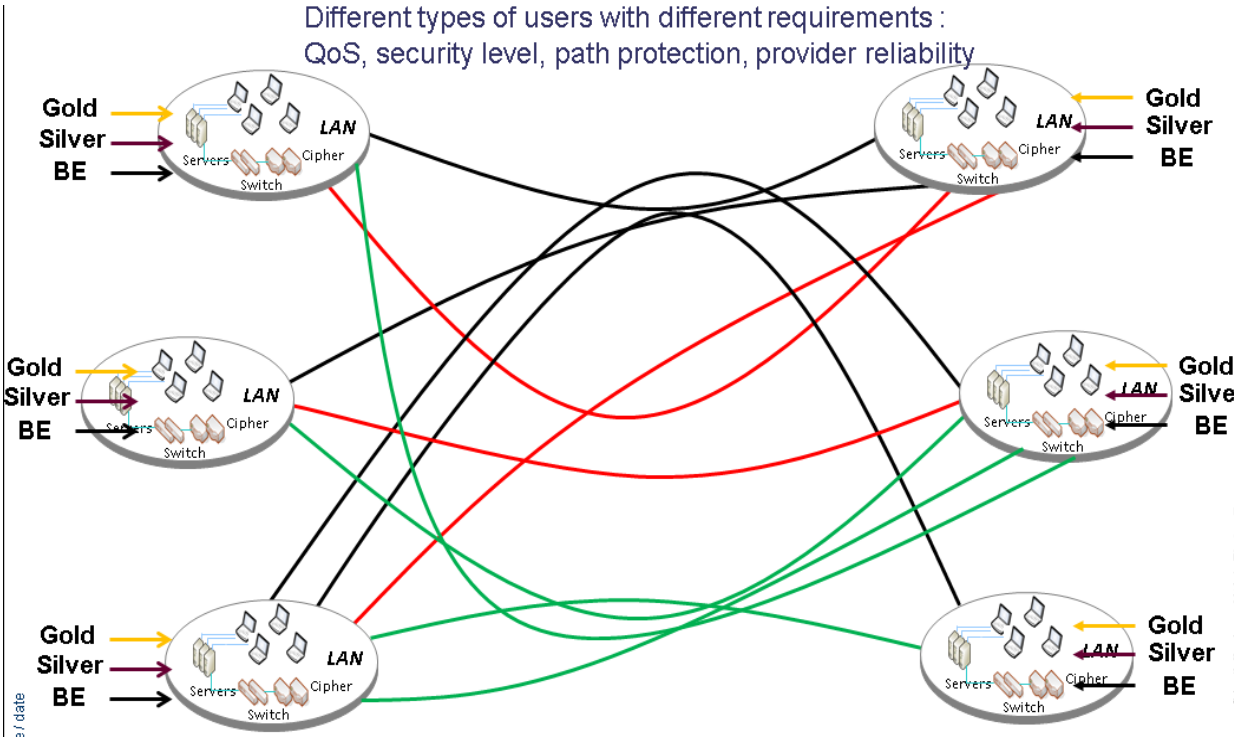

Figure 10: Adopted Approach

Primary and backup paths should be chosen such that they will not experience correlated failures. To this end, the paths must not share the same nodes and links so that in case of failure in the primary path, the backup path will not include the infected link or node. Path diversification mechanism is included in the backup paths selection.

4.2 Implemented Routing Algorithms

One algorithm of each path computation problem has been chosen and implemented. The algorithms that have been chosen are:

- MCP**: Extended Dijkstra** (ED)
  - It can be applied in M constraints - Interesting approach.
- RSP: **A* Prune**
  - It can be easily adopted in our work as it returns K shortest paths and is able to solve the protection scheme (backup path definition) which most of the other algorithms fail to address.
- MOOP: **Skriver and Andersens's LC algorithm**
  - Simplicity-Dijkstra based algorithm
- MCOP: **Heuristic MCOP** (H_MCOP)
  - Low complexity (same as Dijkstra)

The algorithms studied in this work do not solve completely the problem and for this reason they have been extended and adopted in the needs of the present work.

- Extended A* Prune

The extended A* Prune tries to find both the primary and the backup path, reserving in the same time the appropriate bandwidth demand. Firstly, it finds K shortest paths and the one with the minimum TE metric is defined as the primary path. After the hard reservation of the capacity, is runs A* again finding new K shortest paths considering the new capacity of the links. In that case the number of K can have a higher value in order to select more paths that probably provide higher path diversity in comparison to the primary path. The next step of Extended A* Prune is to estimate the diversity metric - which is explained in chapter 2 - of each shortest path compared to the primary path and define the path with the maximum diversity metric as the backup path. The last step is the reservation of the capacity according to user class. The flow chart and the pseudo code of the extended A* Prune is:

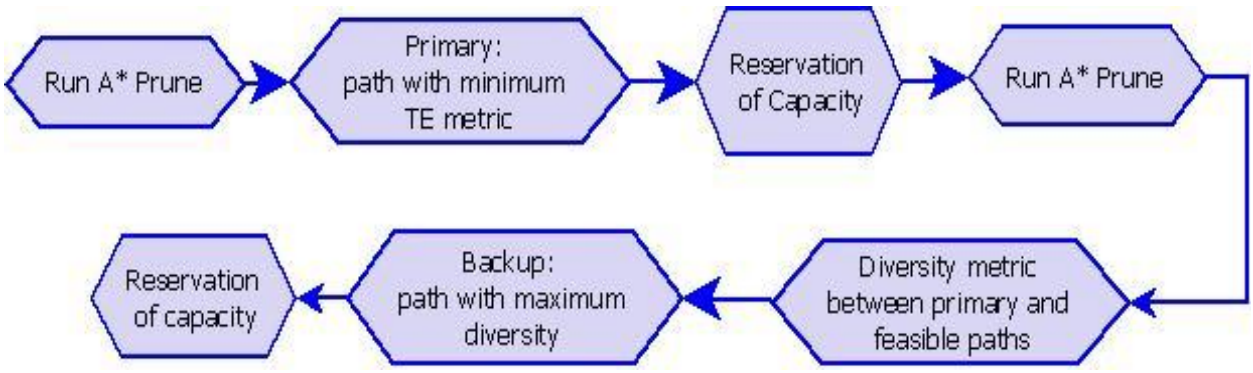

Figure 11: Flowchart of Extended A* Prune

Pseudo code:

Input:

$G = (V, E)$, a graph with node set V and edges set E.

demandList: list with the demands, each demand has the source, the destination, the additive and boolean constraints.

CSP_List: the list which keeps all the feasible, shortest paths

minTEmetric: minimum TE metric of all the feasible shortest paths

primaryPath: the primary path

backupPath: the backup path

fs: feasible shortest path in CSP_List;

diversity_metric: the diversity metric between the primary and a feasible path

common: list which keeps the common elements between the primary and a feasible path

dList: list which keeps the feasible paths and its diversity metric

max_diversity_metric: the maximum diversity metric

Initialization:

minTEmetric = Double_max_value; // is equal to the maximum number of a double

max_diversity_metric = 0;

Primary Path Selection:

```
1.  for each feasible path fp of CSP_List{
2.      if (TEmetric of fp < minTEmetric)
3.          minTEmetric = TEmetric of fs;
4.      }
5.  }
6.  primaryPath = fs with the minTEmetric;
```

Backup Path Selection:

```
1.  PrimaryPath L∪N ; // Links and nodes of the primary path without the source and
                      the destination
2.  for each feasible path fp in CSP_List{
3.      fp_L∪N ; // Links and nodes of the feasible path without the source and
                   the destination
4.      common(PrimaryPath L∪N , fp_L∪N ); // find the common elements
5.      diversity_metric = 1 – (size_of_common \ size_of_primaryPath L∪N );
```

```
6.          add in dList the fp with its diversity metric;
7.      }
8.      for each feasible path fp of dList{
9.          if (diversity metric of fp > max_diversity_metric){
10.             max_diversity_metric = diversity metric of fp;
11.         }
12.     }
13.     backupPath = fp with max_diversity_metric;
```

- Extended H_MCOP and ED

H_MCOP and ED are Dijkstra based and there is a difficulty to discover a path with high path diversity, since every node keeps the shortest path from the source to that node. Consequently, trying to find a diverse secondary path we prune the links (not the nodes) that constitute the primary path. Moreover, these algorithms work only with additive constraints or optimize only additive metrics and therefore an initialization process exists in the beginning in which the links that do not satisfy the boolean constraints are pruned. The flowcharts of the extended algorithms are presented in Fig 12:

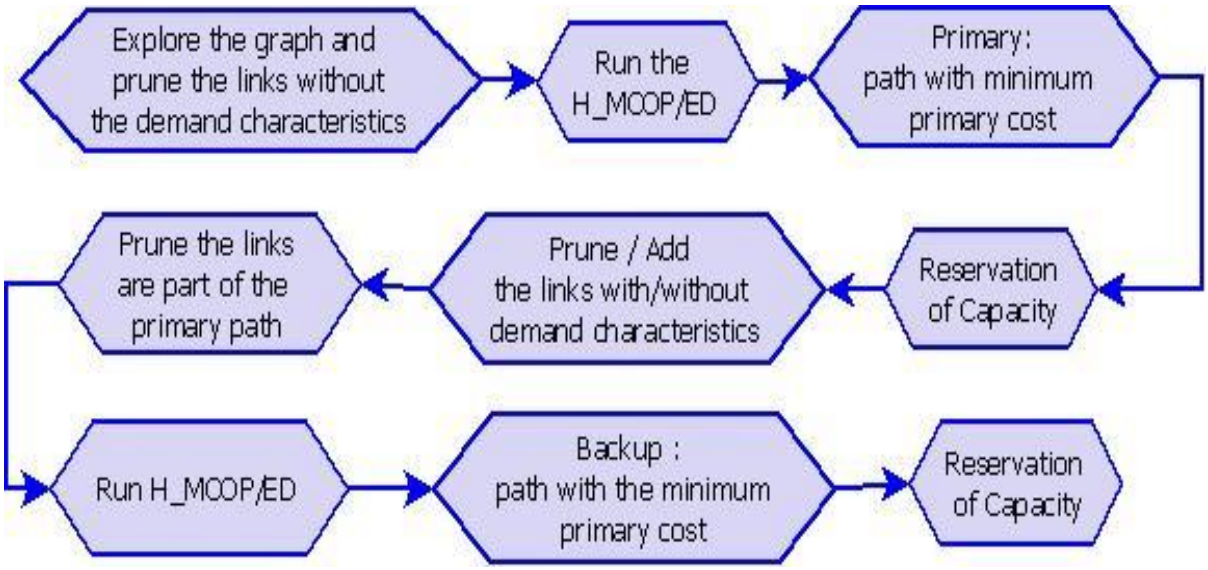

Figure 12: Flowchart of Extended H_MCOP / ED

4.3 Adopted Approach

Our adopted approach is displayed in Figure 12. The input data is the overlay network with the link capacity and characteristics and the traffic demands with the resiliency requirements. In the main process, primary and secondary paths are selected exporting the output data.

The three algorithms have been implemented optimize different metrics but satisfy the same constraints.

Constraints

- **Boolean Constraints:** Bandwidth, Color (link characteristic representation)
- **Additive Constraints:** Hop Limit (delay representation)

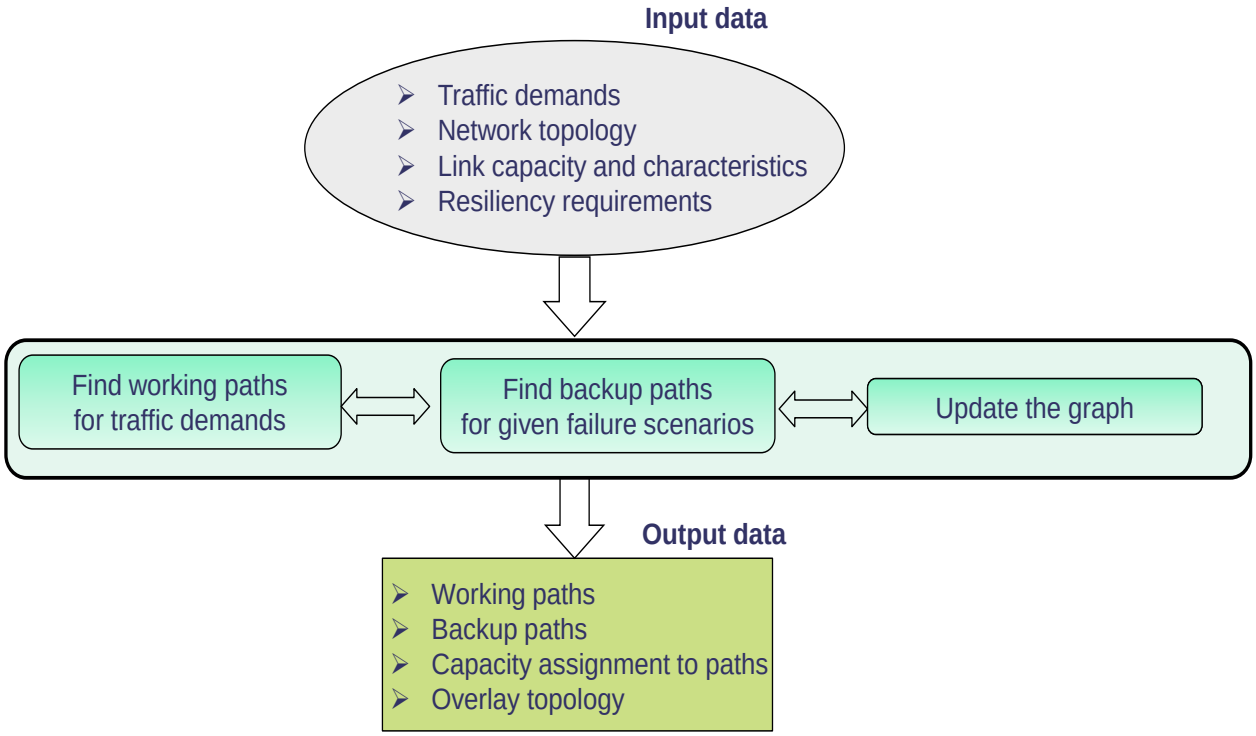

Figure 13 : Adopted approach

Optimization Metrics

- **ED:** Hop count
- **A* Prune:** Inverse of the capacity
- **H_MCOP:** Inverse of the capacity + hop count

It is worth pointing out that these metrics can be replaced from any additive metrics. Their choice was done for defining a concrete example and testing our scenario.

Primary and Backup Paths Selection

ED

- Primary: path found in case of success.
- Backup: path found in case of success after pruning the links of the primary path.

Extended A* Prune

- Primary: path with the minimum TE metric.
- Backup: path with the maximum path diversity.

Extended H_MCOP

- Primary: path with the minimum primary cost.
- Backup: path with the minimum primary cost, after pruning the links of the primary path.

Five topologies have been generated with a different degree distribution. Each topology is comprised of three wide area networks which have different network characteristics. A network generator is used which exports random simple connected graphs with prescribed degree sequence. Given an undirected graph, a degree sequence (Fig. 15) is a monotonic non-increasing sequence of the vertex degrees of its graph vertices. The degree sequence is imported by a power-low distribution of exponent alpha.

Generally, a power law, also know as a scaling law, is a relation of the type $Y = K * X^{-a}$, where Y and X are variables of interest, a is called the power law exponent, and K is a constant. The curve of a power low distribution is presented in Fig 14.

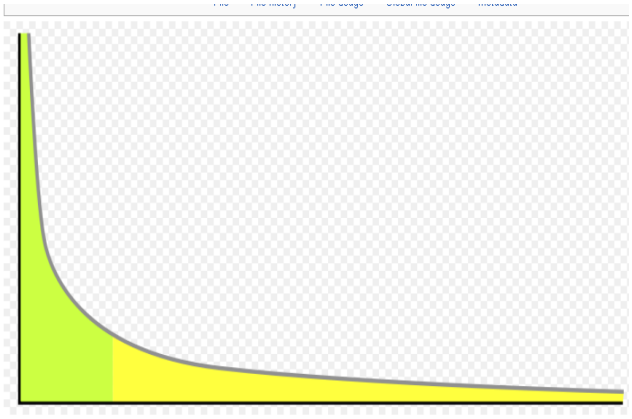

Figure 14: An example of power low graph.

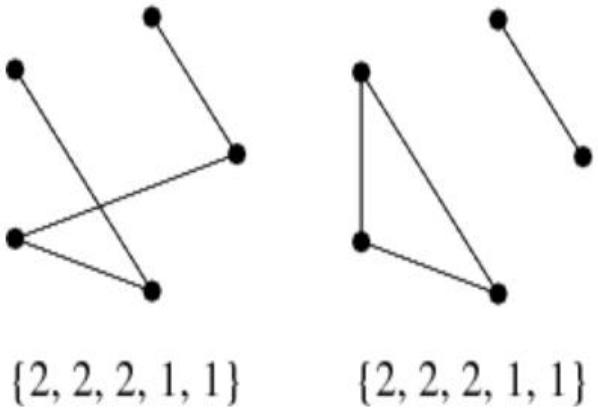

Figure 15: Degree Sequence

The values of alpha (a) are 2, 2.5, 3, 3.5 and 4 and each graph has 20 nodes. The network characteristics are presented in Table 1.

Managing very important information, the WAN 1 requires high security but lacks in the coverage level having a reasonable cost and reliability level. Even if the WAN 2 has a high coverage area and a very good level of security it is very costly to use it and

Table 1: Network Characteristics

Characteristics	Networks		
	WAN 1	**WAN 1**	**WAN 3**
Security	High	Medium	Low
Coverage Area	Low	High	Medium
Cost	Medium	High	Very Low
Reliability	Medium	Low	High

provides low reliability. As for WAN 3, it provides the best effort service without any significant requirement from the side of user and therefore the cost is very low.

Table 2: Resiliency Rules

User Classes	Primary Path		Backup Path	
	Networks	**Reservation**	**Networks**	**Reservation**
Gold Users	WAN 1	Hard	WAN 1 + WAN 2	Hard

Silver Users	WAN 1 + WAN 3	Hard	WAN 1 + WAN 2 + WAN 3	Soft
Best Effort Users	WAN 1 + WAN 3	Hard	-	-

In Table 2 the resilience rules are determined. As the gold users have high reliability and security requirements they are able to use the safest network, the WAN 1 for the primary path selection. Generally, WAN 2 is used for the backup path selection for both gold and silver users due to its high cost. The silver users can use WAN 1 and WAN 3 for primary paths as the security its not the first priority and all the available networks for secondary paths. As for the best effort users, WAN 1 and WAN 3 are dispensable for the primary but there is no protection scheme. This approach is just a hypothesis, which can be modified according to the needs.

Hard reservation of resources is done for the primary paths of all the kind of users which quite normal as the first path must be guaranteed. For the secondary paths only gold users have a guarantee of protection. Silver users have partial recovery scheme, which means that if they share a common link in their backup paths, only the maximum value of the bandwidth demand is reserved. This approach provides a good balance between QoS and cost. Reserving one path for each primary is very costly and not feasible, but separating the users according to their needs, suitable resilience level can be provided with the respective cost.

The demand table is random and common for all the five networks (Table 3):

Table 3: Traffic Demands

Source	Destination	Constraints		
		User Type	**Bandwidth**	**Hop Limit**
5	3	Gold	15	4
5	3	Silver	10	4
5	3	Best Effort	5	6
4	8	Gold	15	4
4	8	Silver	10	3
4	8	Best Effort	5	3
1	0	Gold	10	3
1	0	Silver	5	3
1	0	Best Effort	14	4
2	7	Silver	13	4
2	6	Silver	10	4
7	8	Gold	5	5
7	8	Silver	5	6
1	2	Silver	10	4
1	2	Best Effort	5	6
9	8	Silver	10	4
9	8	Best Effort	10	4
4	6	Silver	7	3
7	5	Silver	10	3

6	7	Silver	10	3

In our work, the paths of each demand are computed in the order is presented in Table 3. According to the user type, specific resilience level and protection scheme is adopted. This means that networks with specific link characteristics can be used and respective capacity reservation is done. For the silver users, the capacity reservation of the common links of the backup paths is done only once and the reservation is equal to the maximum value. For example, every time we try to reserve the capacity of a silver user, we check the previous backup links of silver users that have already been reserved. If there is any equal link and the value is higher, no reservation is done; otherwise the difference of the capacity is reserved.

5. EVALUATION

5.1 Evaluation Metrics

The problem is trying to be solved of the present work is how to build paths in a way that the robustness of the network will be increased. For this reason, the evaluation metrics have to estimate the level of the resilience our approach provides. Moreover, the complexity of the path computation algorithms and the convergence time is another crucial factor. The metrics have been used are:

- Path Diversity
- Quantitative Robustness Metric (QNRM)
- Gain of Bandwidth Reservation
- Number of satisfied connections
- Convergence time

1) Path Diversity

The path diversity metric [17] has been described in Chapter 2. For two arbitary paths P_a and P_b the path diversity is given as:

$$D(P_b, P_a) = 1 - \frac{|P_b \cap P_a|}{|P_a|}$$

, where $|P_a| \leq |P_b|$. In the present work, the average path diversity of the demands is estimated. The equation is presented below:

$$\text{Average\_Path\_Diversity} = \frac{\sum_{i=1}^{K} D_i}{K}$$

, where K is the number of the demands.

2) QNRM [19]

Assuming that a network is more robust if the service on the network performs better, the performance of the service is assessed when the network is either (a) in a conventional state or (b) under perturbations (failures, virus spreading, etc.), the robustness does depend on the type of impairment that occurs. The term impairment refers to any kind of attack, multiple or cascading failure that can occur within a network.

Impairments or multiple failures are basically divided into two groups: static and dynamics. The former is related to the idea of affecting a network permanently and just once, while the latter is related to an impairment that has a temporal dimension.

Static

Static impairments are essential one-off attacks that affect one or more nodes at any given point. There are, in essence, two forms of static impairments:

Random (SR (Static Random))

In the SR case, nodal attacks occur indiscriminately selecting nodes at random Fig. 16 shows this kind of impairments.

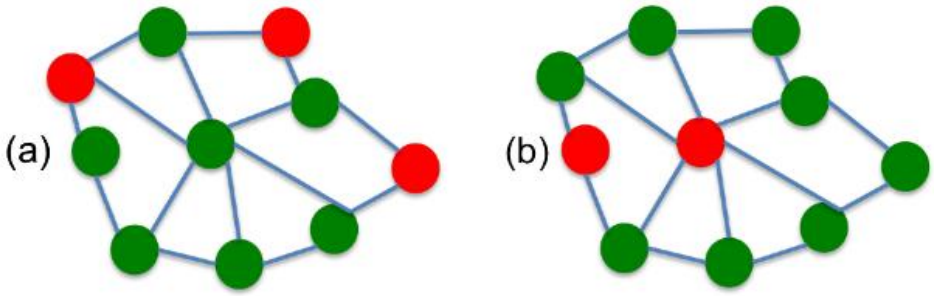

Figure 16: Examples of SR impairment. (a) and (b) show that nodes are chosen randomly.

Target (ST (Static Target))

Nodes in an ST attack are chosen in order to maximize the effect of that attack; there is an element of discrimination in the impairment. The choice of attack target may be a function of network-defined features such as nodal degree, betweenness centrality as well as other "real-world" features, such as the number of users potentially affected and socio-political and economic considerations. Fig. 16, Fig. 17 and Fig. 18 show some examples of ST attacks, considering different elements of discrimination.

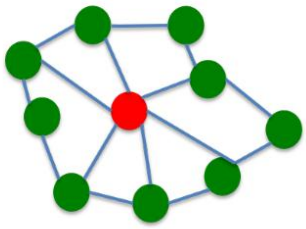

Figure 17: Example of a ST impairment. The element of discrimination is the nodal degree.

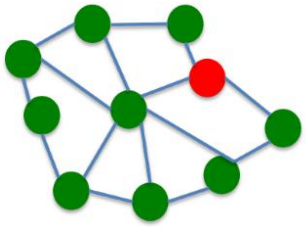

Figure 18: Example of a ST impairment. The element of discrimination is the betweenness centrality

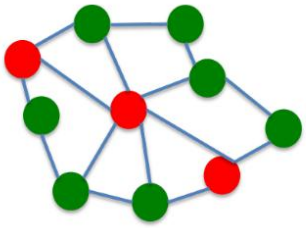

Figure 19: Example of a ST impairment. The element of discrimination is in this case to disconnect the network.

Dynamic

Epidemical (DE (Dynamic Epidemical))

This second type of failures (commonly related to multiple failures such as cascading failures) has a temporal dimension. Two types are defined Epidemical (DE (Dynamic Epidemical)) and Periodical *(DP (Dynamic Periodical)).* Considering a DE, a failure occurs in a node (or a set of nodes of the network) and the failure can spread through the network (becoming an epidemic) or not. The rise and decline in epidemic prevalence of an infectious disease (or failure) is a probability phenomenon dependent upon the transfer of an effective dose of the infectious agent from an infected individual to a susceptible one. Fig. 20 shows an example of how an epidemic can act in a network.

This type of failures is based on epidemic modes (EM) and there are several forms of them. The first type, called the Susceptible-Infected (SI) considers nodes as being either susceptible (S) or infected (I). This type assumes that the infected nodes will remain infected forever and, so, can be used for “worst case propagation”. Another type is the Susceptible-Infected-Susceptible (SIS), which considers that a susceptible node can become infected on contact with another infected node, then recovers with some likelihood of becoming susceptible again. Therefore, nodes will change their state from susceptible to infected, and vice versa, several times. The third kind is the Susceptible-Infected-Removed (SIR), which extends the SI model to take into account the removed state. In the SIR group, a node can be infected just once because when the infected nodes recover, they become immune and will no longer pass the infection onto others. Finally there are two models that extend the SIR one: SIDR (Susceptible Infected Detected Removed) and SIRS (Susceptible Infected Removed Susceptible). The first one adds a Detected (D) state, and is used to study the virus throttling, which is an automatic mechanism for restraining or slowing down the spread of diseases. The second one considers that after a node becomes removed, they remain in that state for a specific period and then go back to the susceptible state.

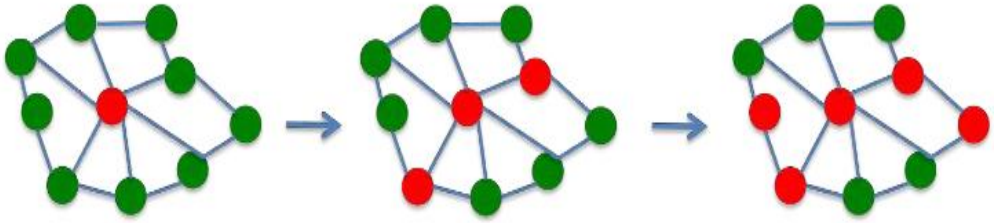

Figure 20: Example of a DE impairment. A failure occurs on a node, and after a period of time, it spreads to its neighbors.

Periodical (DP (Dynamic Periodical))

A DP is simply any kind of impairment that occurs periodically following its characteristic cycle.

Computing a TE LSP (Label-Switched Path) can be done by two options: offline and online path computation. With offline path computation, an offline tool is used to compute the path of each TE LSP, taking into account the constraints, the network topology and the resources. Because the computation is simultaneously performed for all the TE LSPs in the network, offline tools try to achieve a global network optimization with multiple criteria such as maximum link utilization, minimized propagation delay, and so on, and with the objective of maximizing the amount of traffic the network can carry. This can be achieved thanks to the global view of the network characteristics and traffic demands.

The online path computation method relies on distributed path computation, whereby each router is responsible for computing the path(s) of the TE LSP(s) it is the headend for. No central server computes the TE LSP's path in the network.

Online path computation is more dynamic, more reactive to network and traffic changes, and more robust (it does not rely on a single centralized server) because of its distributed nature. It also yields less-optimal paths. In contrast, the offline approach usually allows for a higher degree of optimality at the price of less dynamicity, scalability, and increased management overhead.

In our work, an offline path computation is studied, and therefore only the static impairments can be considered. The dynamic impairments require the time parameter for spreading the failures in the network, and so on the online path computation.

QuaNtitative Robustness Metric

The *Quantitative Robustness Metric* or QNRM analyses how an impairment of any kind (SR, ST, DE, or DP) affects the number of connections established on a network. In this metric, the number of Blocked Connections (BC) in each time step is analyzed. We define a BC as a connection that should have been established at time t but could not be established as a consequence of nodal failures.

Define BC(t) as a number of BC in a given time step, TTC(t) as the number of connections that should have been established in the same time step. The quotient

shown in the following equation:

$$QNRM[t] = \frac{BC(t)}{TTC(t)}$$

The average of all values obtained during the interval of interest is the QNRM.

$$QNRM = \frac{\sum_{t=1}^{Total} QNRM[t]}{Total}$$

In offline path computation, the parameter of time can not be used so the equation of the Quantitative Robustness Metric is differentiated as:

$$QNRM = \frac{BC}{TTC}$$

In the present work, QNRM is estimated in case of maximum nodal degree and maximum betweenness centrality (node and link).

3) Gain of Bandwidth Reservation

This metric is related to the cost and represents the percentage of bandwidth is not needed to be reserved using our approach compared to the approach of having a secondary path for each primary path with hard reservation in both of them (one plus one technique). The equation of the metric is:

$$G = 1 - \frac{RB}{DB}$$

, where G is the percentage of the gain, RB the reserved bandwidth according to the approach of this work and DB is the reserved bandwidth using the one plus one technique.

4) Number of satisfied connections

What is very important in this work is to observe which algorithm manages to satisfy all the connections or the bigger number of them, which depends on the metrics they optimize and the way they work. The first priority of path computation is the ability to connect the sites each other and then to provide a good level of resilience.

The formula of this metric is the connections are able to be satisfied by the specific algorithm compared to the whole amount of the demands. As bigger is the number of satisfied connections, as more efficient is the specific algorithm.

5) Convergence Time

The convergence time has been defined as the time that each algorithm needs to select the paths of all the traffic demands. This useful metric is related to the complexity, as more complex is the algorithm as longer is the convergence time.

5.2 Results

In this part, the results of the present work are displayed. The three algorithms ED, A* Prune and H_MCOP are compared in five values of alpha. As greater is the value of alpha, as lower is the connectivity and more links are available. It is worth referring that we do not compare the performance of each algorithm in the values of alpha because it is not feasible. The graphs are random, so it is not safe to give a conclusion scrolling the value of alpha. We compare the three algorithms each other in each different value of alpha.

Diagram 1 depicts the average of path diversity metric (%) of each algorithm in each value of alpha. What is noticeable is that the values of all three algorithms are very close and quite high (over 70%). This means that our approach is very effective in path diversification and provides a very good level of resilience. The values of A* Prune are slightly higher than ED's and H_MCOP's because of its advantage to select a set of feasible paths so that it can choose the one with the maximum diversity metric. Even if the path diversity peaked at alpha equal to 3, we can observe that in the higher value of alpha the path diversity has the lowest value which is quite normal since as lower the connectivity is as less links are available and as lower is the path diversity.

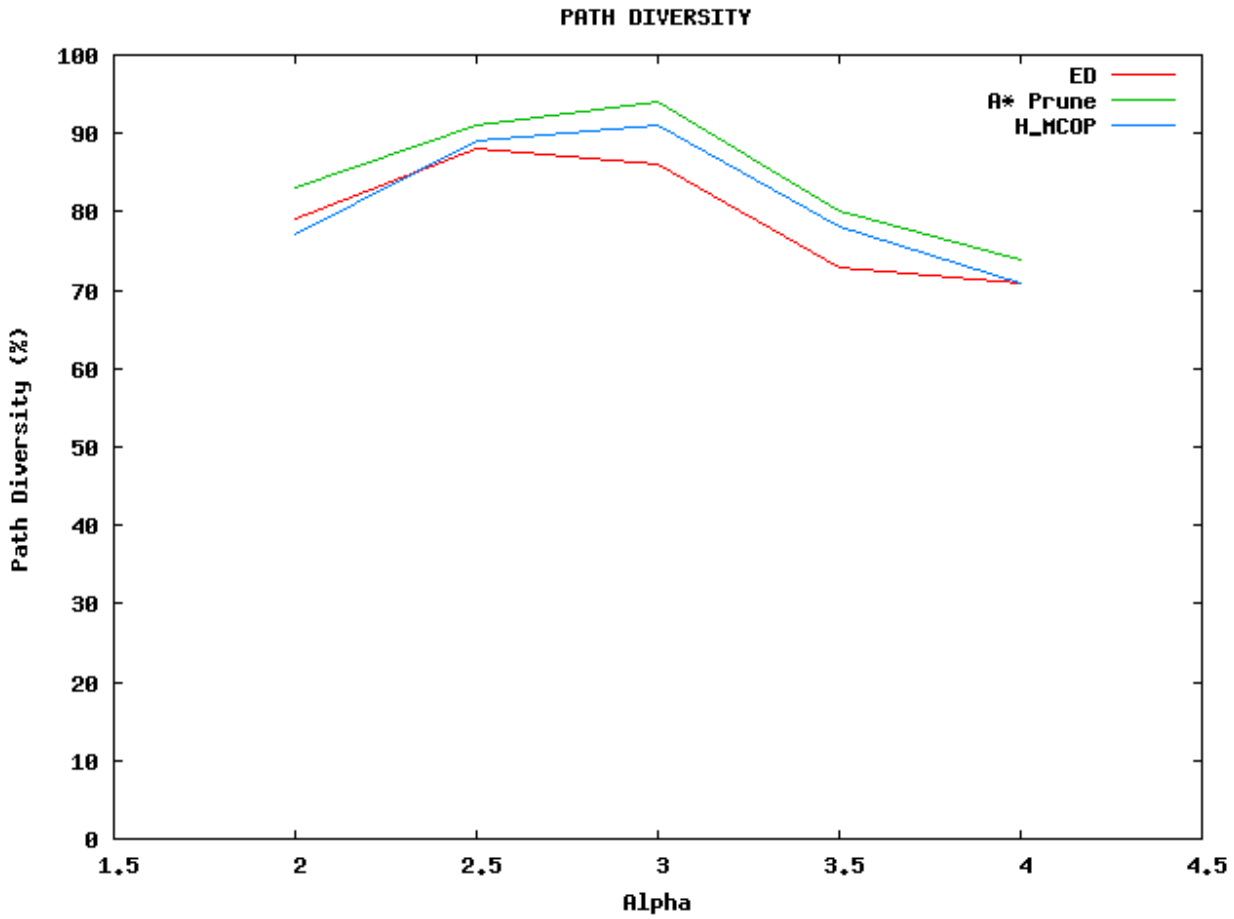

Diagram 1: Path Diversity VS routing algorithms

Even if the A* Prune seems to provide a better resilience level, the convergence time is dramatically higher as we can see in Diagram 2. This happens because A* Prune explores a big part of the graph trying to find a set of shortest feasible paths. In contrast, ED and H_MCOP are Dijkstra based trying to find one shortest path. The convergence time of A* Prune bottomed out in alpha equal to 2 and this happens because of the high connectivity, as more links available for checking as more the convergence time.

ED has the lowest complexity so as the lowest convergence time. In the best case, ED finds the shortest path applying Dijkstra in respect to one metric and if this path satisfies the other constraints, the problem has been solved with Dijkstra's complexity and convergence time. H_MCOP has a middle performance in terms of resilience level and convergence time and it can be considered as a balanced solution.

Therefore, if an application needs high level of resilience, A* Prune seems to be more suitable but in this case the cost of convergence time is very high. From the other hand, if the priority of the service is a quick computation, ED and H_MCOP are more appropriate algorithms.

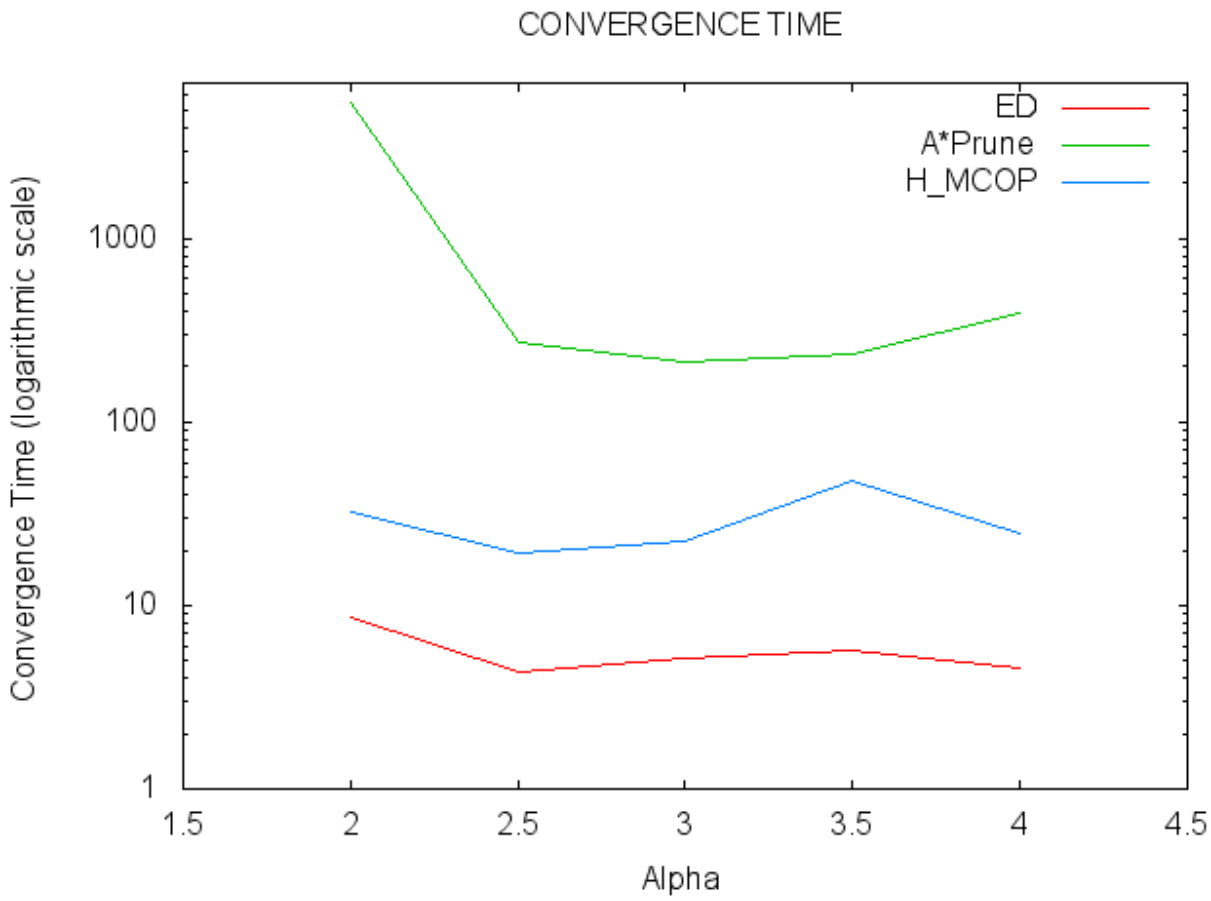

Diagram 2: Convergence time VS routing algorithms

The following 3 diagrams depict the convergence time of each algorithm for 10 and 20 nodes respectively. For A* Prune and H_MCOP is quite obvious that in case of 20 nodes the algorithm needs to explore a bigger number of nodes and therefore the convergence time is higher than in case of 10 nodes. As for ED, the convergence time depends from the time the algorithm finds the first feasible path, and so the solution. In detail, if the algorithm returns success without examining the neighbours, the convergence time is too low in comparison to the case of examining one or more neighbours.

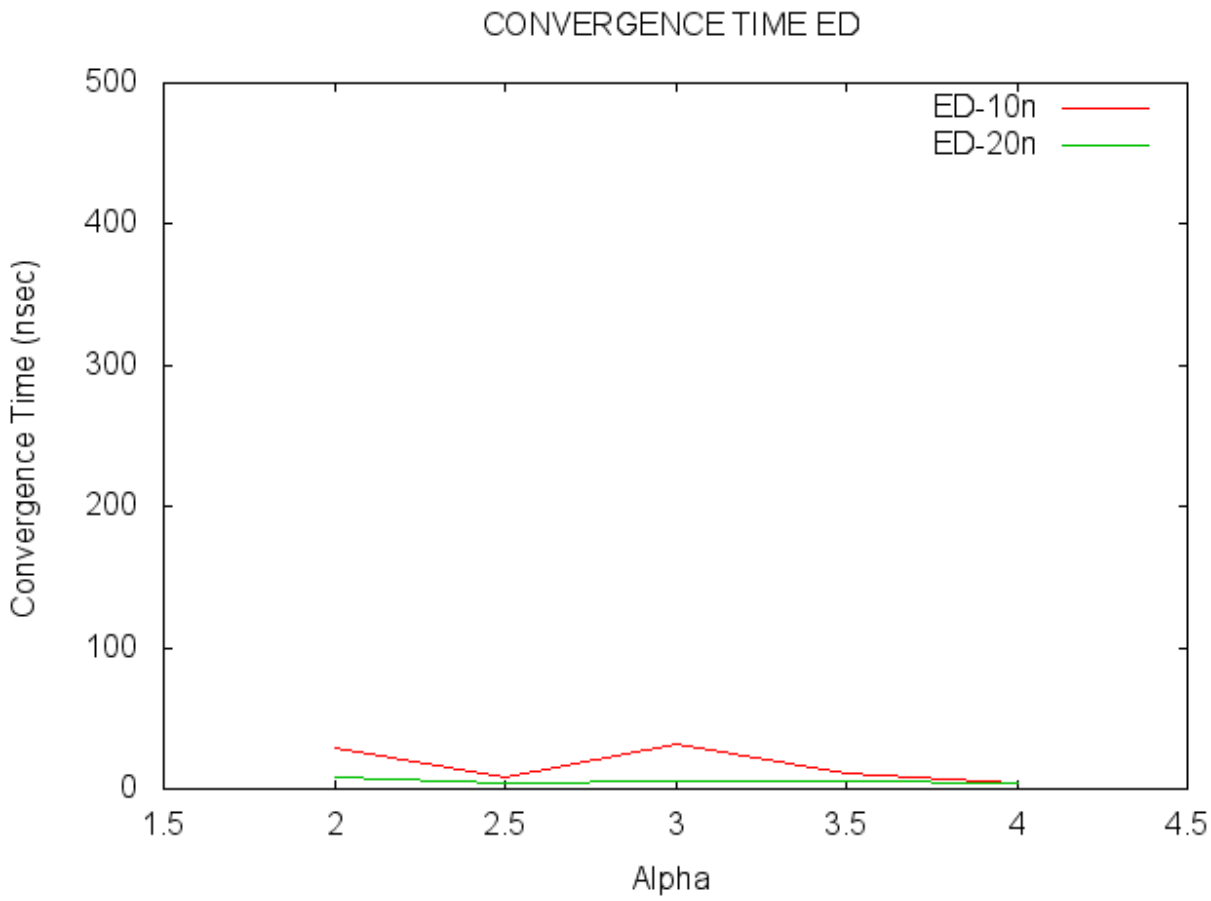

Diagram 3: ED Convergence time of a graph with 10 and 20 nodes

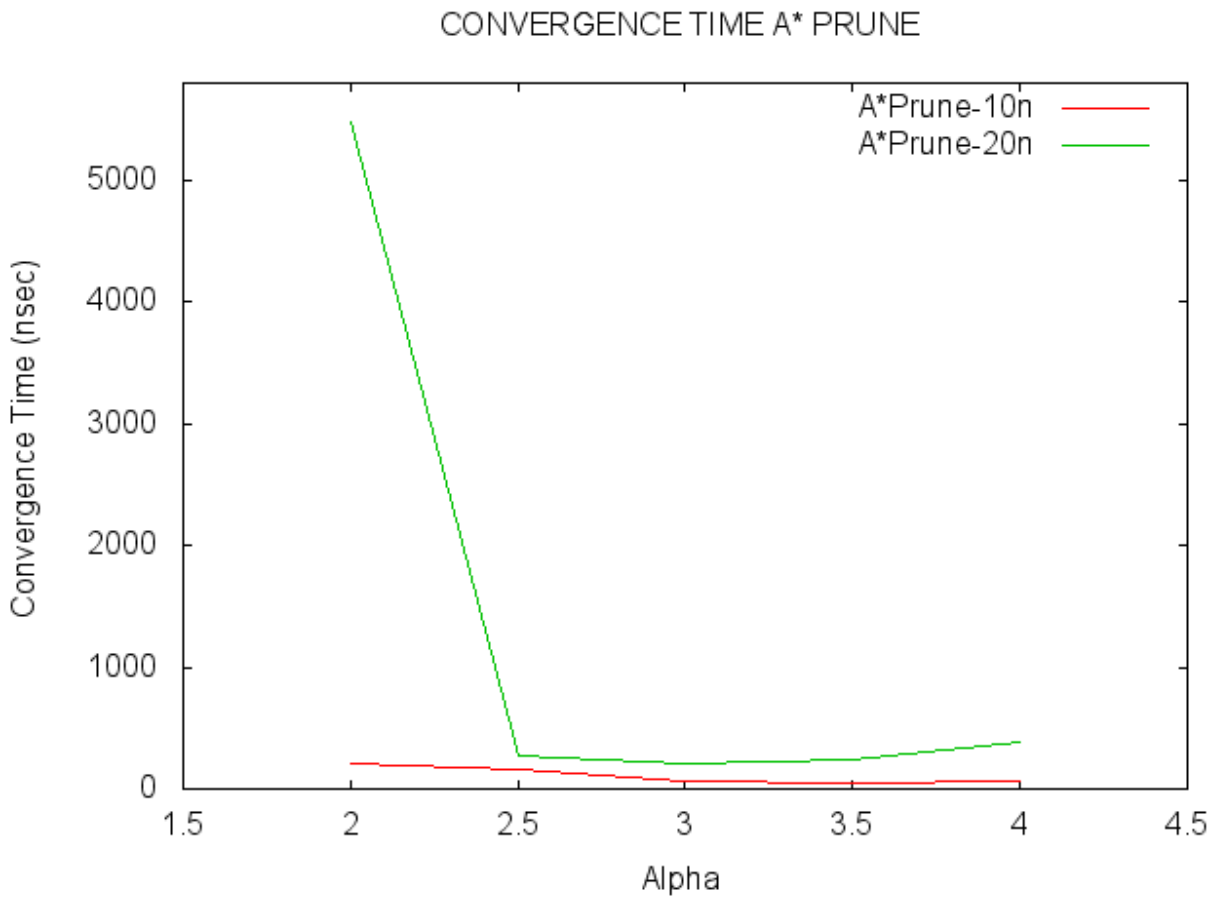

Diagram 4: A* Prune Convergence time of a graph with 10 and 20 nodes

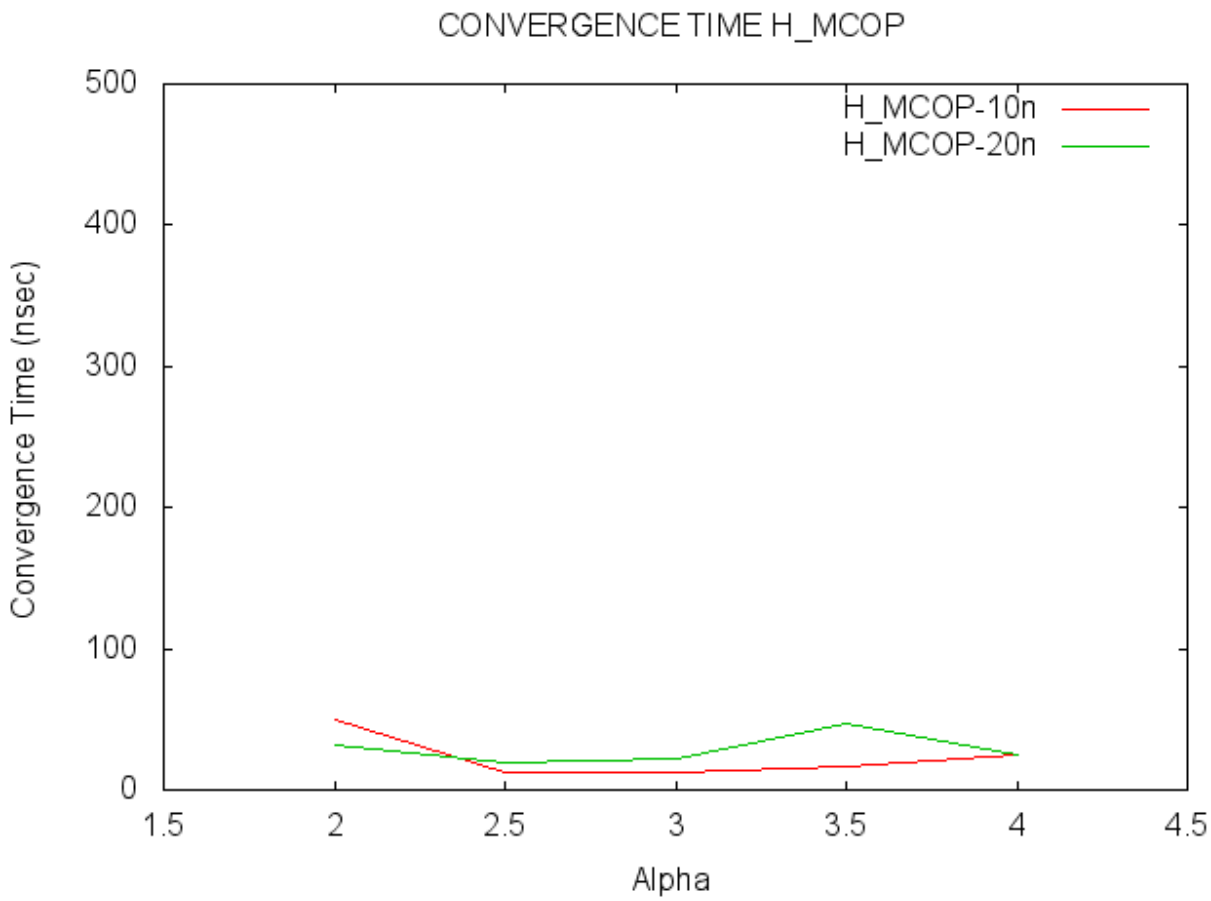

Diagram 5: H_MCOP Convergence time of a graph with 10 and 20 nodes

As far as the QNRM is concerned, as lower is the value as more robust is the network and more paths can be saved using the backup paths approach. Diagram 6 outlines the percentage of robustness of the network between the algorithms. In our graphs the node with the max in-degree, max out-degree and max betweenness centrality is the same node so the results are presented in the same graph.

ED and H_MCOP have competitive performance which is lower compared to the A* Prune. This caused because of its lower level of path diversity. ED takes into account one metric for the optimization, so it does not share the traffic in the links of the network and H_MCOP, optimizing many metrics, can not expand the path so much so the selected path is more restricted. Especially in our case H_MCOP optimizes the inverse of the capacity and the hop count. The hop count represents the delay so the path must not only have a low cost but also a small delay. From the other hand, A* Prune can expand the path, if the cost is low, without considering the hop count. A* Prune seems to have a little better performance and this is due to the higher level of path diversity.

Better performance of all three algorithms is presented in QNRM of link failures with the maximum betweenness centrality. The comparison of the algorithms is quite the same but the values are much smaller. This happens because the generated graphs have fewer nodes and more links, so if a node fails more paths are in danger than in a link failure. Furthermore, it is good to keep in mind that in ED and H_MCOP the links of the primary path are pruned and the backup path is selected without considering those

links. Consequently, a node is more important or critical in a network than a link.

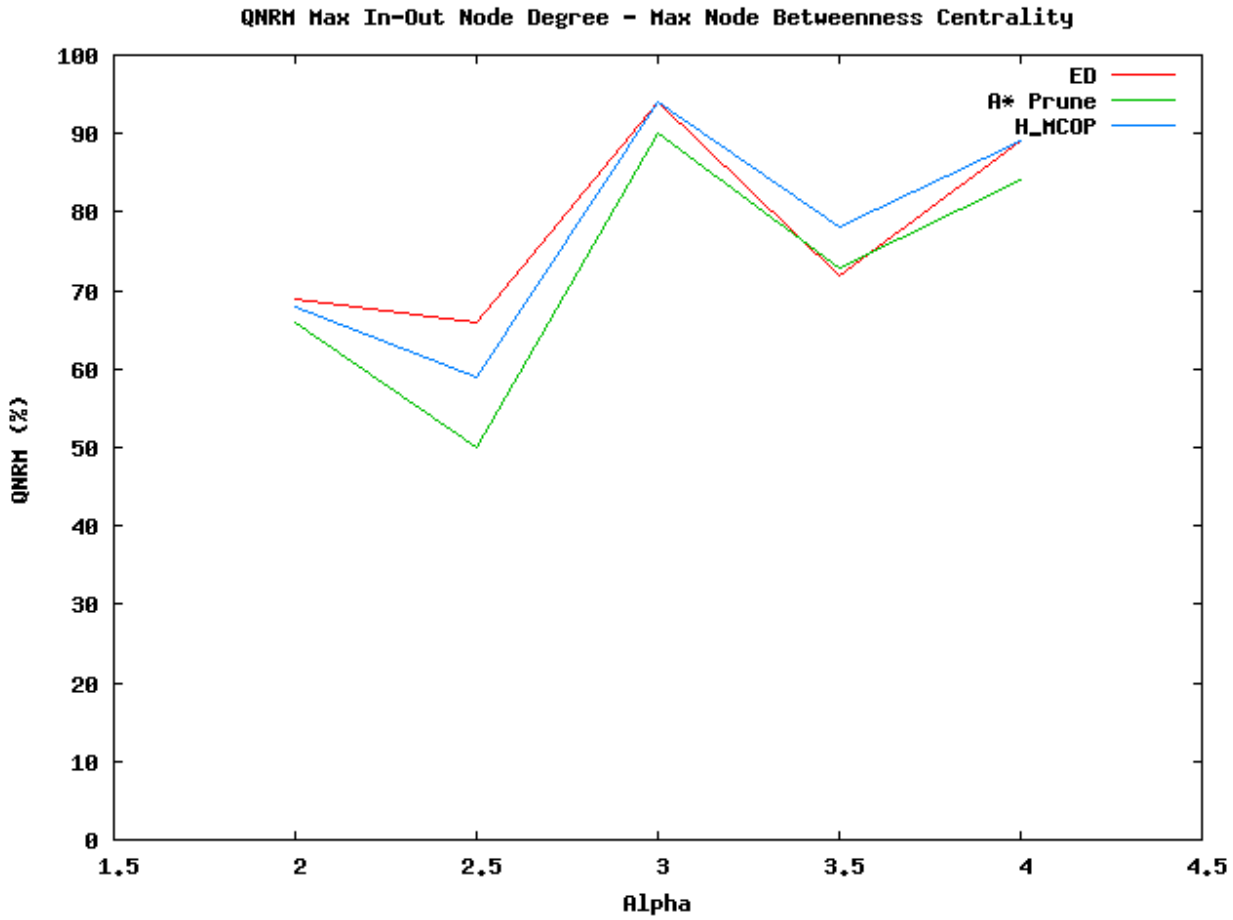

Diagram 6: QNRM – max in-out node degree, max node betweenness centrality VS routing algorithms

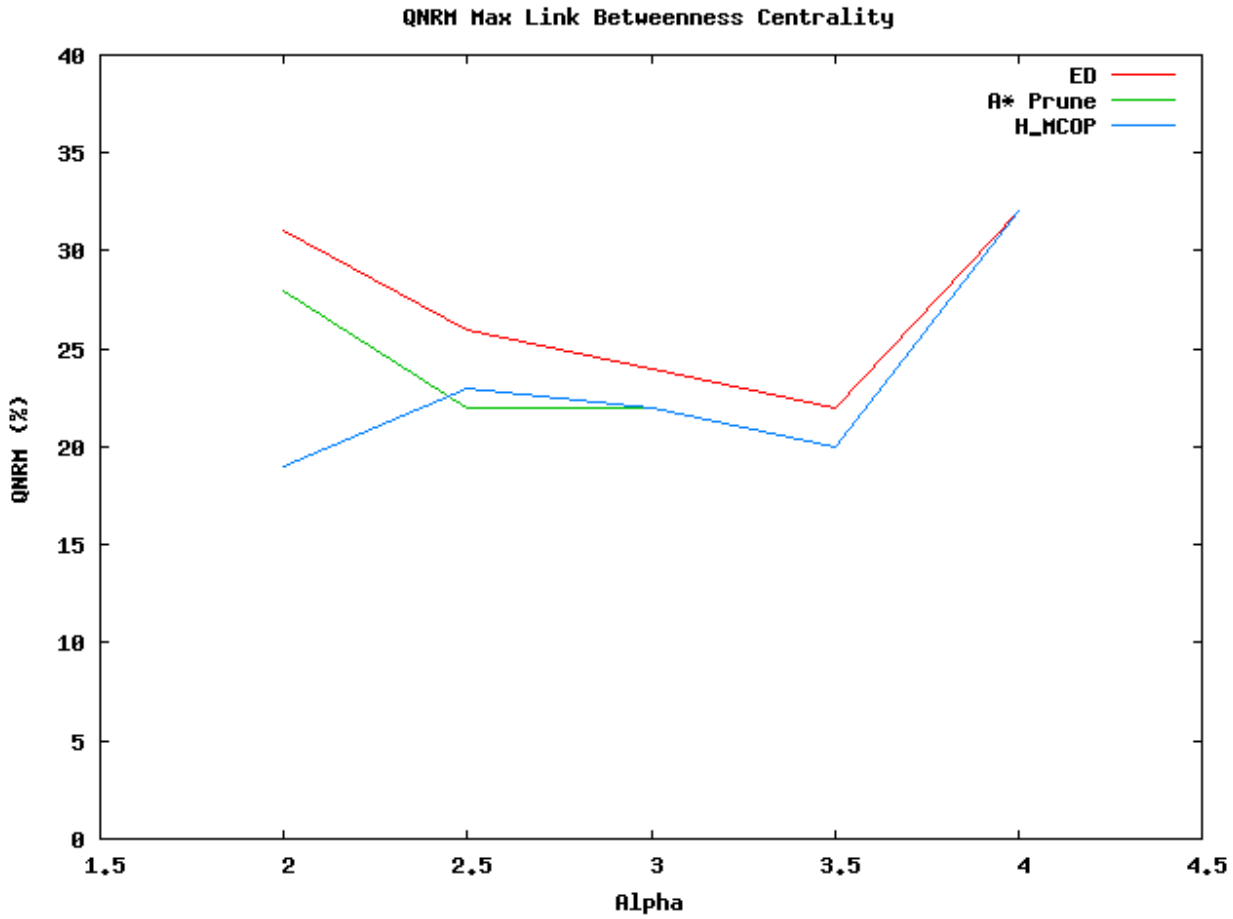

Diagram 7: QNRM – max link betweenness centrality VS routing algorithms

The goal of our work is to provide a balance between the QoS and the cost of the services. The diagram 8 depicts the gain in bandwidth reservation using our approach in comparison to which we have a secondary path for each primary path with hard reservation in both of them.

Clearly, ED requires the minimum amount of bandwidth reservation. This caused due to the low path diversity. Few different links means that the silver users share the same links and therefore the partial capacity reservation is done. ED and H_MCOP are in the same level.

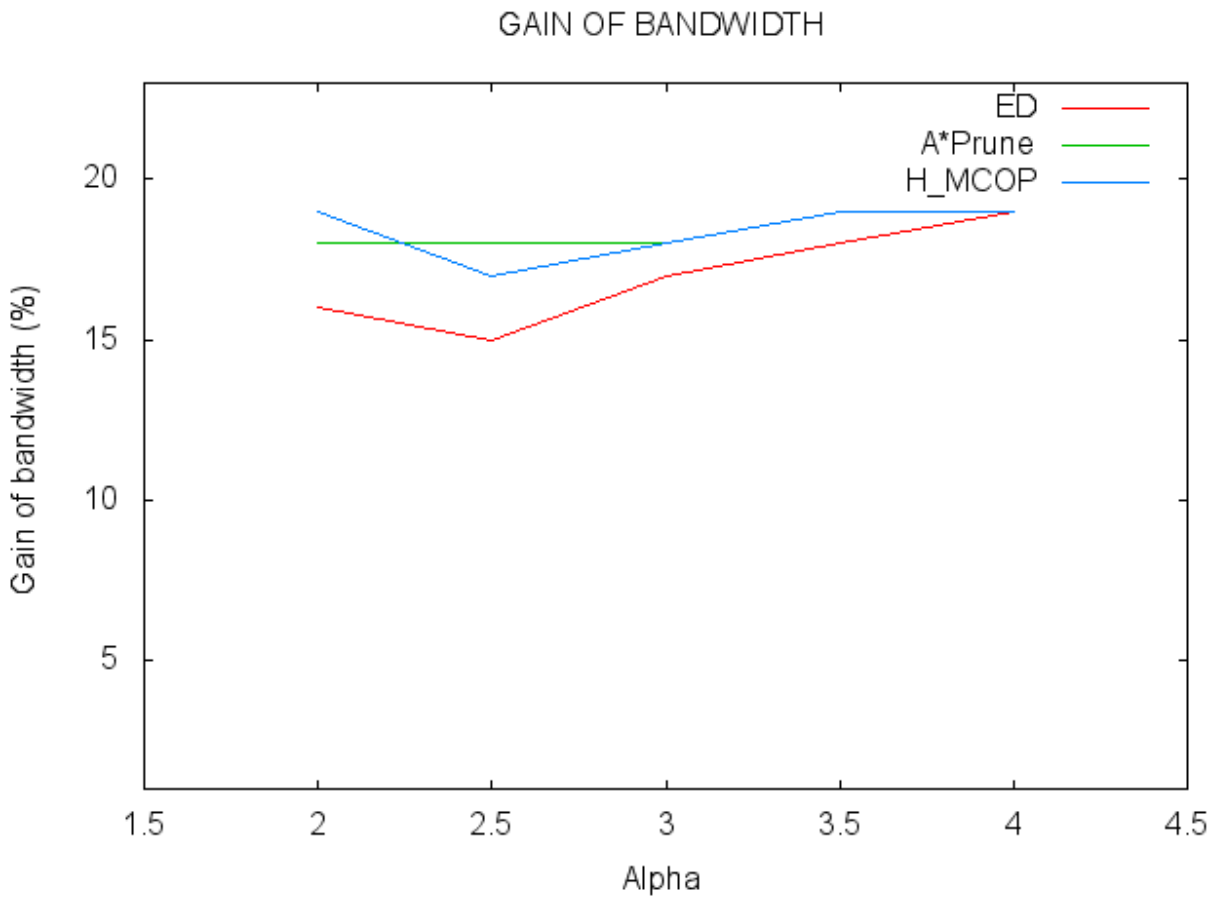

Diagram 8: Percentage of Bandwidth Reservation Gain VS routing algorithms

Except of the level of resilience, it is of paramount importance for the algorithm to be able to connect as more paths as it is possible. The metric that estimates those parameters is the number of satisfied connections in Diagram 9. It is clear that ED is not as powerful as H_MCOP and A* Prune because of the way it works. Running Dijkstra means that it uses the same links so it can not share the traffic in the network and some links are easily lack for resources.

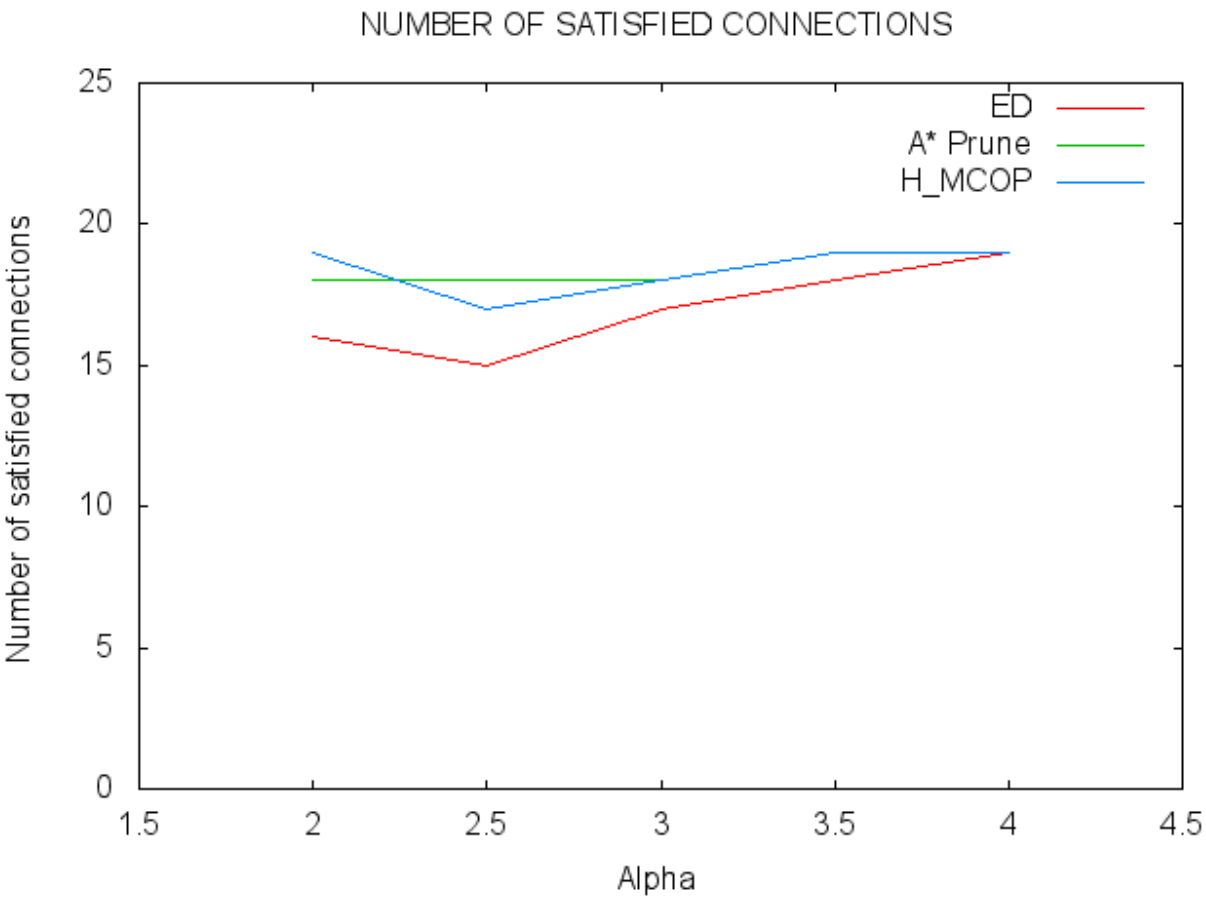

Diagram 9: Number of satisfied connections VS routing algorithms

6. CONCLUSION

In this thesis, a new approach on building paths between the sites in an overlay view is proposed, enhancing at the same time the resilience and the robustness of the network. Different user classes have been proposed for different level of QoS in terms of reliability, security, protection and cost. Three path computation algorithms (ED, A* Prune, H_MCOP) have been implemented and extended to cover our needs.

Our results indicate that all the extended algorithms provide high path diversity (over 70%). Moreover, in the worst case of failure of the most critical links and nodes, many paths are managed to be saved. This means that our approach is effective on creating fault tolerant networks while keeping a reasonable cost in terms of bandwidth provisioning.

Depending on the needs the most appropriate algorithm can be chosen. A* Prune seems to provide the highest level of reliability and security, having the highest values in path diversity, QNRM and number of satisfied connections. Nevertheless, using A* Prune the cost in time convergence is extremely high.

If many metrics are needed to be considered (cost, delay, jitter), H_MCOP is the most suitable algorithm. Generally, its performance is quite close to A* Prune but with much lower convergence time.

Finally, ED can be used if the priority is just to find a feasible path. Only one metric may be optimized because of the way it works, but the convergence time is significantly low and also the gain of bandwidth is very high. The drawback of ED is that it does not provide so high resilience level as the other algorithms because of the lower path diversity, QNRM and number of satisfied connections. Therefore, it can be used for the less QoS–sensitive applications or fast path computation.

In future work we will test our scenario in a big scale network of 100 nodes trying to observe any differences with the present results. Since our work is offline and we can not take into account many parameters, another interesting test could be the implementation of an online path computation.

REFERENCES

[1] Pragyansmita Paul and S V Raghavan, “Survey of QoS Routing” .
[2] R. G. Garroppo, S. Giordano, L. Tavanti “A survey on multi-constrained optimal path computation: Exact and approximate algorithms”. Dip. Ingegneria dellÍnformazione, Universita de Pisa.
[3] G. Xue, A. Sen, W. Zhang, J. Tang, K. Thulasiraman, Finding a path subject to many additive QoS constraints.
[4] Salman Yussof and Ong hang See, Finding Multi-Constrained Path Using Genetic Algorithm.
[5] F. A. Kuipers, T. Korkmaz, M. Krunz, P. Van Mieghem, “Overview of Constraint-Based Path Selection Algorithms for QoS Routing”
[6] T. Korkmaz M. Krunz “Multi-Constrained Optimal Path Selection”, Department of Electrical & Computer Engineering, University of Arizona
[7] M. R. Garey and D.S. Johnson. Computers and Intractability, A Guide to the Theory of NP-Completeness. Freeman, 1979.
[8] Jaffe, J. (1984) Algorithms for Finding Paths with Multiple Constraints: Networks, 14, 95-116
[9] Chen, S., Nahrstedt, K. (1998) On Finding Multi Constrained Paths: Proceedings of ICC’98, 884-879.
[10] D. Adami, R. G. Garroppo, S. Giordano, L. Tavanti “Multi-Constrained Path Computation Algorithm for Traffic Engineering over Wireless Mesh Networks”
[11] “Implementing A Constraint-based Shortest Path First Algorithm in Intelligent Optical networks”
[12] Gang Liu, K. G. Ramakrishnan, A* Prune: An algorithm for Finding K Shortest Paths Subject to Multiple Constraints
[13] H. D. Neve, P. V. Mieghem, “TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm”
[14] H. Ma, Y. Guo, J. Zhang, “Fast Reroute upon Multi-homed Domains Link Failures”
[15] R. Atkinson, S. Bhatti, S. Hailes, “Harmonised Resilience, Security, and Mobility Capability for IP”
[16] Crina Grosan, Ajith Abraham, Aboul Ella Hassainen, “Designing resilient networks using muticriteria metaheuristics” .
[17] Justin P. Rocher, Abdul Jabbar, James P.G. Sterbenz, “Path Diversification: A Multipath Resilience Mechanism”
[18] B. Szviatovszki, A. Szentesi, A. Juttner, “On the Effectiveness of Restoration Path Computation Methods”.
[19] Marc Manzano Castro, “Metrics to Evaluate Network Robustness in Telecommunication Networks”.