A Link Failure Recovery Algorithm For Virtual Network Function Chaining

Oussama Soualah, Marouen Mechtri, Chaima Ghribi and Djamal Zeghlache
Institut Mines-Telecom, Telecom SudParis, CNRS UMR 5157 and Universit Paris Saclay, France
Email: {oussama.soualah, marouen.mechtri, chaima.ghribi, djamal.zeghlache}@telecom-sudparis.eu

***Abstract*—This paper addresses Virtual Network Functions (VNFs) placement and chaining in the presence of physical link failures. A decision tree approach to the NP-Hard VNF placement and chaining problem is used to minimize the penalties induced by service interruptions due to link outages. Formulating the problem as decision tree reduces the complexity significantly and leads to a new reliable algorithm, named R-SFC-MCTS, that builds incrementally the decision tree to efficiently search for good placement and chaining solutions. Execution time is improved thanks to the Monte-Carlo Tree Search strategy. The proposed link failure recovery algorithm selects and assigns reliable paths to prevent and avoid the negative effects of link failures and reactively re-maps impacted virtual links in safer physical paths once an outage occurs in the infrastructure. The performance of R-SFC-MCTS is compared via extensive simulations with a baseline and a reactive solution in terms of: i) acceptance rate, ii) induced penalties iii) provider revenue loss and iv) the final provider's profit.**

Keywords: Service Function Chaining, Network Function Virtualization, Monte-Carlo Tree Search, Link Failure, Reliability.

I. Introduction

Network Functions Virtualization (NFV) has emerged as an innovative concept to simplify the deployment and management of networking services using virtualization and Cloud technologies. Deploying network services on virtual machines reduces time and cost compared with the dedicated traditional hardware implementations (using servers, switches, etc.). According to a recent SDN and NFV survey [1], the factors driving companies interest in NFV are: reduced deployment time, reduced OPerating EXpense (OPEX) and improved management flexibility. This survey reports also that within four years, about 83% of IT organizations are likely to have made significant NFV deployments in their physical infrastructures.

The NFV architecture, a European Telecommunications Standards Institute (ETSI) coordinated initiative, is based on the concept of Virtualized Network Functions (VNFs) representing the software implementation of network functions. VNFs, act as network services building blocks that can be connected and chained to provide an end-to-end communication service. In this context, the efficient placement and chaining of VNFs (such as load balancers, firewalls, IPS/IDS, etc.) while steering traffic across the VNFs is becoming essential. As network function virtualization relies on virtualization for partitioning and sharing physical cloud and network resources, ensuring failure robustness and recovery is as important to respect the Service Level Agreements (SLA) and contracts established between consumers and providers. The providers will pay penalties for SLA violations on the one hand and the quality of service and experience of the customers will degrade on the other hand. Statistics from [2] highlight revenue losses due to unplanned IT outages and indicate: i) IT businesses in North of America are collectively losing 26.5 billion USD in revenue each year through IT downtime and data recovery; ii) North American businesses collectively suffer from $1,661,321$ hours of IT downtime each year. That is an average of 10 hours per company yearly. A deep study of data center failures [3] shows that the daily mean number of links outage is 40.8. Evidently, some survivable mechanisms are mandatory in order to guarantee a reliable service. A failure could be caused by either a link or node failure. Link failures are very common [3] and cause significant service interruptions while commodity switches are more reliable. Modern data centers typically implement node redundancy to greatly improve reliability. We consequently focus and consider, in this paper, only link failures that are of more concern than node failures.

The VNFs placement and chaining problem is known to be NP-Hard and has been addressed based on exact (i.e., for small instances) and approximation methods (e.g., [4], [5]). In our work, we add failure recovery to an efficient VNF chain placement strategy to avoid SLA violation due to link failures and to improve reliability.

We aim at providing a scalable and efficient algorithm for VNFs placement and chaining combined with preventive and reactive mechanisms to address physical **link failures**. A way to compute the optimal embedding is to enumerate all the candidate hosts for each virtual resource (i.e., virtual node and/or link) within the physical network (i.e., hardware nodes and/or paths). Clearly, this is typically computationally intractable in large scale networks because of the combinatorial explosion leading to exponential execution times. To reduce complexity, we formulate the survivable (to link failures) Service Function Chaining (SFC) placement as a decision tree problem. Since building the entire decision tree is not feasible for large infrastructures, we propose a new Reliable Service Function Chaining (R-SFC) algorithm using the Monte-Carlo Tree Search (MCTS) strategy [6] and name it R-SFC-MCTS. The key advantage of MCTS is the **incremental construction** of the decision tree when **searching** for an optimized solution. This characteristic of MCTS fits very well our problem since only partial tree construction is needed to make decisions. MCTS is also known to perform well in many combinatorial Computer Games (e.g., computer Go [7]) and complex real-world planning and optimization problems [6].

R-SFC-MCTS algorithm uses five main steps for initial embedding: i) Tree initialization, ii) Selection of the node to explore during tree navigation, iii) Generation of a new sub-branch, iv) Updating of nodes' relevance and finally v) Selection of the best solution among generated branches. As already mentioned, R-SCF-MCTS continuously and gradually builds the decision tree. At tree initialization, the root node, which is an abstract node that represents the system status before embedding any resource of the current SFC request, is created. Starting the navigation from the root node, R-SFC-MCTS checks at each visited node if all its children are already developed and added to the decision tree. If this condition is satisfied, the second stage (i.e., selection) is performed to select the most suitable next node to continue the navigation process.

If the condition is not satisfied, the third stage (i.e., sub-branch generation) is started and one of the non developed child is generated at each subsequent tree-level until arriving to a leaf node. The generation of a new child corresponds to the mapping of one VNF node and the directly connected virtual link of the requested chain. A leaf node corresponds to: i) the mapping of the entire request or ii) a blocking status where it is not possible to further embed virtual resource(s) due to a lack of available physical resources. These stages are repeated until a computational budget limit, used to stop the execution of R-SFC-MCTS, is reached. Finally, the branch providing the best embedding is selected based on mapping quality. The selected branch has maximum computed path reliability. This information is directly found in the leaf node and avoids parsing all the tree branches. By selecting the most reliable physical links during embedding we minimize the impact of link failures. In addition to this **preventive** approach, we **reactively** re-embed impacted virtual links in safer paths. Performance evaluation based on extensive simulations show that R-SFC-MCTS achieves good performance in terms of i) acceptance rate, ii) induced penalties iii) provider revenue loss and iv) final provider revenue.

Section II of this paper presents related work and Section III describes the system model. Section IV provides details on our link failure management approach. Section V reports the results of performance evaluation.

II. Related work

There have been recent work on VNF placement and chaining which is gaining more and more attention in the literature. In [8], authors proposed an eigendecomposition based approach for joint VNFs placement and traffic steering of their associated forwarding paths and graphs. A heuristic based on a greedy algorithm is also presented to solve the problem iteratively. The Greedy solution is based on bipartite graph construction and matching techniques and solves the problem in two steps (mapping VNFs then steering traffic between them). In [9], authors formulated a Genetic Programming based approach to solve the VM allocation and network management problem. Authors in [10] solved the problem of network service chaining using an ILP and a heuristic algorithm. Their proposed heuristic is based on a decomposition selection with backtracking phase and on a mapping phase. In [5], authors proposed an ILP and a heuristic for VNF placement and chaining based on a transformation of the problem by adding new virtual switches. The idea is to model the problem as a Multi-Stage directed graph and to run the Viterbi algorithm [11] on it. All this prior art addresses VNF placement and chaining does not consider resource failures and there have been no attempts to handle failure recovery automatically.

Failure recovery for the Virtual Network Embedding (VNE) problem received more attention and some survivable algorithms for VNE have been proposed in [12], [13], [14], etc. The proposed strategies be classified using: i) centralized or distributed, ii) proactive or reactive, iii) the type of the protected resource (i.e., router and/or link, geographic region) iv) backup use, v) curative or preventive, vi) batch or online principles. In [15], authors proposed a new Survivable Virtual Network Embedding algorithm denoted by SVNE. First, SVNE proactively computes a set of possible backup detours for each substrate link. Then, SVNE embeds each new Virtual Network (VN) request by calling the VN embedding strategy D-ViNE [16]. Finally, when a link failure occurs, a reactive mechanism is invoked to re-embed the affected virtual link on the backup detours computed in the first step. The authors formulated the problem of re-mapping the virtual links as a linear program. The main objective is to minimize the bandwidth consumption and the penalties due to link failures.

In [17], the authors proposed a new algorithm, named RMap, dealing with the mapping of VNs by considering failures of substrate links. To ensure some levels of reliability, RMap allocates backup links. Indeed, once a link stress is higher than a predefined threshold, RMap computes its backup detour based on Loop Free Alternate resilience approach. When a link failure occurs, virtual links transiting over this substrate link will migrate to the backup links. However, RMap ensures protection only for stressed links. In other terms, if a failure occurs within a non-stressed link, all clients will be impacted.

However, these approaches are not applicable in the NFV context and especially VNF chain placement as claimed in [18], [4], [19] because of some dissimilarities. For instance, in the VNE context, requests are modeled by simple undirected graphs, whereas VNF chains are more complex components that contain both the VNFs to place and the traffic flows to steer between end points. Besides, the sharing of VNFs among several clients is not supported in the VNE environment. Hence, any comparison with the survivable algorithms proposed in the VNE context will neither be fair nor meaningful.

In the NFV context, authors in [20] addressed the problem of a VNF (node) failure. The proposed solution selects an alternative VNF, in a greedy manner, to replace the failed one then ensures the communication by allocating a path between the new VNF and its neighbor(s). As claimed by the authors, their solution is temporary and sub-optimal. The work is mostly experimental and does not propose a mathematical model we can compare our proposal with.

In this work, we formalize the VNF placement as a tree decision problem and we propose a reliable algorithm for VNF placement and chaining which deals with link failure recovery. To the best of our knowledge, this is the first work that solves the VNF placement and chaining problem using a Monte-Carlo Tree Search strategy and that seeks to optimize link failure recovery.

III. Problem formalization

This section formalizes the reliable service function chaining problem and presents the physical network and VNF chain request models.

A. Substrate and virtual networks models

The substrate or physical network, defined by the ETSI [21] NFV Infrastructure (NFV-I), is modeled as an undirected weighted graph, denoted by $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$ where $E(\mathcal{G})$ is the set of the physical links and $V(\mathcal{G})$ is the set of the physical nodes. Each substrate node, $w \in V(\mathcal{G})$, is characterized by

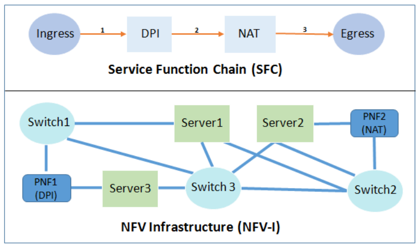


Fig. 1: The SFC and NFV-I topologies

its i) available processing power (i.e., CPU) denoted by $C(w)$, and ii) type $T(w)$: switch, server or physical network function (PNF) [22], where PNF are traditional dedicated physical hardware that implement network functions. The physical links, $e \in E(\mathcal{G})$, are characterized by their i) available bandwidth $B(e)$ and ii) reliability at time t.

As in [23], we formulate the Mean Time Between Failures (MTBF) for the physical links as a Weibull distribution. Accordingly, it is possible to estimate and predict the equipment's reliability at any time t. Consequently, the Cumulative Distribution Function (CDF) of the MTBF is expressed as:

$$F(x) = 1 - \exp(-\left(\frac{x}{a}\right)^b), \ x \geq 0 \qquad \text{(III.1)}$$

where a and b are the parameters of the Weibull distribution. We assume heterogeneous physical links in the physical network (i.e., links have different failure probability). To do so, each physical link $e \in E(\mathcal{G})$ has its own initial age denoted by $A(e)$. Accordingly, we classify the links within the substrate network into three groups: i) **young**, ii) **adult** and iii) **old**. Formally, the probability to fail at a time t of a substrate link e is equal to $\mathcal{P}_r(e,t) = F(A(e)+t)$. It is worth pointing out that the substrate links' reliability is inversely proportional to its initial age.

The client request (i.e., requested service function chain) is modeled as a directed graph, denoted by $\mathcal{D} = (V(\mathcal{D}), E(\mathcal{D}))$ where $V(\mathcal{D})$ is the set of the virtual nodes and $E(\mathcal{D})$ is the set the virtual links. Each virtual nodes, $v \in V(\mathcal{D})$, is characterized by its i) required processing power $C(v)$ and ii) its type $T(v)$: switch (i.e., ingress or egress) and VNF. Each virtual link $d \in E(\mathcal{D})$ is described by its required bandwidth $B(d)$. Figure 1 depicts the SFC and the NFV-I topologies. The virtual topology is a chain of VNFs and switches. Examples of VNFs are: firewalls, DPIs, IDSs, load balancers, NATs, etc., as described in IETF [24] specifications. Regarding the NFV-I depicted in Figure 1, we note the existence of two PNFs and some interconnected servers beside the two switches serving the ingress and/or egress flows in the SFC topology.

B. SFC optimization problem

We now summarize the main constraints related to the service function chaining optimization problem. Each virtual node $v \in V(\mathcal{D})$ may be embedded in a substrate node, $w \in V(\mathcal{G})$, that has enough physical resources and for which $C(w) \geq C(v)$. Note that a virtual switch is mapped only onto a physical switch, and a VNF may be embedded in a PNF or a physical server based on the ETSI recommendations [21]. This simplifies the identification for each virtual node type, $v \in V(\mathcal{D})$, a set of physical candidate hosts $Cand(v) \subset V(\mathcal{G})$. Each virtual link, $d \in E(\mathcal{D})$, should be mapped to one physical path P that meets the required bandwidth. In other terms, each physical link $e \in E(\mathcal{G})$ forming the path P should have enough remaining bandwidth to serve the required bandwidth. Formally, $\forall e \in P, B(e) \geq B(d)$. Note that splittable link mapping is not considered in this paper.

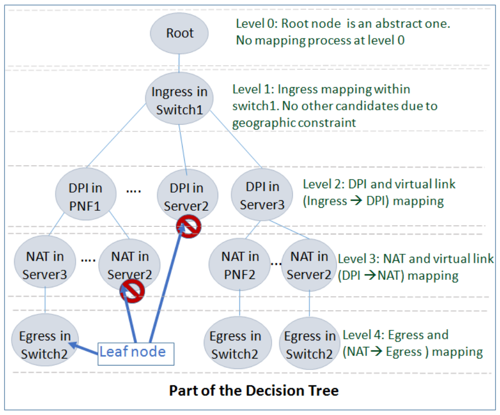

Root
Level 0: Root node is an abstract one. No mapping process at level 0
Ingress in Switch1
Level 1: Ingress mapping within switch1. No other candidates due to geographic constraint
DPI in PNF1
....
DPI in Server2
DPI in Server3
Level 2: DPI and virtual link (Ingress → DPI) mapping
NAT in Server3
....
NAT in Server2
NAT in PNF2
...
NAT in Server2
Level 3: NAT and virtual link (DPI → NAT) mapping
Egress in Switch2
Leaf node
Egress in Switch2
Egress in Switch2
Level 4: Egress and (NAT → Egress) mapping
Part of the Decision Tree

Fig. 2: The decision Tree of the SFC optimization problem

In order to reduce complexity by avoiding time consuming and unnecessary exhaustive search, we model the reliable SFC mapping problem as a decision tree denoted by $\mathcal{T}$. Each tree-node represents the embedding of one virtual node, $v \in V(\mathcal{D})$, in one of its physical candidate hosting node $w \in Cand(v)$. The connection between two tree-nodes $n1$ and $n2$ (i.e., the father and its child) describes a valid embedding of the virtual node $v1 \in V(\mathcal{D})$ and its successor $v2 \in V(\mathcal{D})$ as well as the mapping of the virtual link between them within a physical path. Accordingly, generating a new child $n2$ in the decision tree $\mathcal{T}$ from the father tree-node $n1$, that matches with the embedding of a virtual node $v1$, requires a valid mapping of i) the virtual node $v2$ (i.e., the successor virtual node of $v1$ in the SFC), and ii) the connection link. The tree root node is defined as an abstract node that reflects the state of the NFV-I before handling the current request. A tree leaf node corresponds to a placement outcome that can be either: i) the successful mapping of the entire SFC request or ii) the rejection of the request due to a lack of available physical resources. A branch in the decision tree $\mathcal{T}$ corresponds to the mapping of the SFC virtual resources.

An example of a portion of the decision tree $\mathcal{T}$ is illustrated in Figure 2. Each tree level corresponds to the mapping of one virtual node except level 0 that contains and defines only the "abstract" root node. In level 1 of this example, the ingress node must be mapped only in $switch1$ due to geographic constraints. The DPI may be mapped in any of the three servers or $PNF1$ according to available computing resources and the optimization criteria. The NAT function should be mapped into a server or $PNF2$ while meeting the requested computing resource. At the last level, the egress point should be embedded only on $switch2$ to respect also geographic constraints.

This example highlights the combinatorial complexity and the need to devise judicious strategies that can incrementally build the decision tree $\mathcal{T}$ when searching an optimized mapping. In Section IV, we will describe the proposed Monte-Carlo Tree Search to meet these objectives.

IV. Proposal: R-SFC-MCTS strategy

This section presents our proposal named **Reliable Service Function Chaining based on the Monte-Carlo Tree Search**

Algorithm 1: R-SFC-MCTS psuedo-code

```
1  Inputs: NFV-I, SFC
2  Output: $\mathcal{B}_b$
3  Initialization ($\mathcal{T}$)
4  while Computational Budget do
5      $n \leftarrow$ Selection-Node-Explore($\mathcal{T}$)
6      $\mathcal{B} \leftarrow$ SubBranch-Generation($\mathcal{T}, n$, NFV-I, SFC)
7      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{B}$
8      Update-Relevance($\mathcal{T}$)
9  $\mathcal{B}_b \leftarrow$ Selection-Best-Solution($\mathcal{T}$)
10 return $\mathcal{B}_b$
```

strategy (R-SFC-MCTS) to address VNF chain placement including reactions to link failures. Since we formulate the SFC placement or embedding problem as a decision tree model, we present the main decision tree search approaches from the literature and justify the adoption of the MCTS strategy. We also describe our SFC-MCTS algorithm steps serving as a basis to address failures via the R-SFC-MCTS extension.

A. Tree search optimization algorithms

One of the most challenging issues in decision tree search is scalability since the problem becomes computationally intractable for large size trees. Some approaches were introduced to reduce search complexity such as Best-First-Search [25], Branch&Bound [26], A* [27], etc. These strategies assume that the decision tree is already entirely built and the remaining task consists in finding the best solution in the complete or full tree. They consequently continue to suffer from combinatorial explosion and experience unacceptable execution time with increasing problem size. In our case the decision tree is not entirely generated, and the construction is dynamic and incremental and this reduces significantly complexity.

Our proposal R-SFC-MCTS mainly relies on the Monte-Carlo Tree Search [6] (MCTS) strategy. In fact, MCTS has one key advantage that allows coupling i) incremental tree building with ii) an efficient search mechanism. This fits well the SFC problem because the decision tree is not already built. MCTS has in addition good performance in the Computer Game area such as computer Go [7] as well as some other optimization problems [6]. Recently, MCTS was successfully applied to solve the batch virtual network embedding problem [28]. Taking all these elements into consideration, we adopt the MCTS strategy to efficiently guide the decision policy. The suitable game category that best fits our problem description is the **Single Player** game [29] where there is no opponent. In fact, the objective is to maximize the single player's payoff that matches well the objective of enhancing the provider revenue. We present next, the different stages of our proposal.

B. R-SFC-MCTS description

R-SFC-MCTS incrementally builds the decision tree $\mathcal{T}$ and in parallel searches for the best mapping. R-SFC-MCTS is devised around five main stages: i) Tree initialization, ii) Selection of the node to explore during the tree navigation, iii) Generation of a new sub-branch, iv) Update of nodes' relevance (i.e., back propagation) and finally v) Selection of the best solution. We summarize our proposal in Algorithm 1.

1) Tree initialization: At this step, the root of the decision tree $\mathcal{T}$ is initialized and created. The root node is an abstract node that represents the system status before embedding any resource of **the current SFC** request.

2) Selection of the node to explore: This stage is mandatory for the exploration process. Once arrived to a tree-node (i.e., starting from the abstract root node) one child should be selected, among its children, to continue the navigation. In order to select one child, R-SFC-MCTS ensures balance between i) exploitation of promoted branches and ii) exploration dealing with less promising ones to avoid local optima. For this aim, we define for each tree-node $n \in \mathcal{T}$ a relevance tree-function, χ_n, that indicates the relative importance of tree-nodes, n, candidate for selection:

$$\chi_n = \bar{\mathcal{R}}_n + \alpha \cdot \sqrt{\frac{\ln(\hat{\mathcal{V}}_n)}{\mathcal{V}_n}} + \sqrt{\frac{\sum_l {\mathcal{R}_n^l}^2 - \hat{\mathcal{V}}_n \cdot (\bar{\mathcal{R}}_n)^2 + \beta}{\mathcal{V}_n}} \tag{IV.2}$$

where i) $\bar{\mathcal{R}}_n$ is the average revenue generated by all the branches crossing the tree-node n, ii) $\mathcal{V}_n$ is the number of visits to n, iii) $\hat{\mathcal{V}}_n$ is the number of visits of n's parent (i.e., predecessor in the tree), iv) $\mathcal{R}_n^l$ is the revenue generated by the branch $\mathcal{B}_l$ transiting over n.

Regarding α and β, they are calibration constants. Note that $\mathcal{R}_n^l$ is computed once the leaf node is generated (i.e., an entire branch is built). In fact, $\mathcal{R}_n^l$ quantifies the quality of the mapping corresponding to the generated branch. It is defined based on the offered reliability of the selected physical links to embed the different virtual links of the client request. In order to quantify $\mathcal{R}_n^l$, we first define a substrate path P reliability $Rel(P, t)$ as follow:

$$Rel(P,t) = \frac{\sum_e \frac{1}{\mathcal{P}_r(e,t)}}{len(P)}, e \in P \tag{IV.3}$$

where $len(P)$ is the path P length in term of hops and t is the arrival time of the current request. By doing so, $Rel(P, t)$ describes a reliability average of the current path. Formally, $\mathcal{R}_n^l$ is defined as follow:

$$\mathcal{R}_n^l = \sum_i Rel(P_i, t) \tag{IV.4}$$

where P_i is the selected path to embed the i-th virtual link. We should highlight that the third term in the equation IV.2 quantifies a possible search deviation of node n [29]. Indeed, it includes the sum of the squared revenues $\mathcal{R}_n^l$ corrected by the expected revenues $\hat{\mathcal{V}}_n \cdot (\bar{\mathcal{R}}_n)^2$. The β constant is useful to push for the exploration of rarely visited nodes. The above equation IV.2 allows R-SFC-MCTS to control balancing between the exploitation of promising nodes and exploration of rarely visited nodes. By doing so, R-SFC-MCTS tries to avoid local optima. We should highlight that the selection stage is applied only when **all the children** of the visited tree-node are already generated. Otherwise, the next stage is processed.

3) Generation of a new sub-branch: Arriving to one tree-node that has at least one child not already developed in $\mathcal{T}$, R-SFC-MCTS generates **one** new tree-node, **at each subsequent level**, until reaching a leaf node. At each level, one candidate, from the $Cand(v_i)$ set, is choosen to host the corresponding virtual node v_i (i.e., VNF, Ingress/Egress), then the attached virtual link is embedded as well in a substrate path. This physical path is the one that maximizes the reliability among the

Algorithm 2: SubBranch-Generation

```
1 Inputs: $\mathcal{T}, n$, NFV-I, SFC
2 Output: New sub-branch $\mathcal{B}$
3 Initialization ($\mathcal{B}$)
4 while isLeafNode($n$) = False do
5     $n \leftarrow$ Compute-Child-Node($n$)
6     $\mathcal{B} \leftarrow \mathcal{B} + n$
7 return $\mathcal{B}$
```

paths connecting the two extremities. Algorithm 2 summarizes the sub-branch generation stage. We recall that the generation of a new tree-node requires the embedding of the next virtual node (i.e., the following node in the SFC chain) as well as the mapping of the virtual link connecting both of them. The process of new node generation at each level will end by creating a leaf node based on two cases:

- Mapping all the virtual nodes and links of the SFC, or
- Blocking due to a lack of physical resource. In other terms, it is impossible to embed further virtual node(s) and/or link(s) because of hardware resource shortage.

At the end of this stage, a new sub-branch $\mathcal{B}$ is generated. Accordingly, the decision tree $\mathcal{T}$ is enhanced by adding the new sub-branch $\mathcal{B}$. Formally, $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{B}$.

4) Update of nodes' relevance: Once the decision tree $\mathcal{T}$ is enhanced by the new sub-branch $\mathcal{B}$, R-SFC-MCTS computes the mapping quality at the leaf node n (i.e., $\mathcal{R}_n^l$) then it updates the parameters of the selection function (i.e., χ_{n^i}), defined in equation IV.2, for any tree-node n^i belonging to $\mathcal{B}$. More precisely for all nodes $n \in \mathcal{B}$, i) $\bar{\mathcal{R}}_n$ and $\mathcal{R}_n^l$ are updated with respect to the cumulative revenue induced by $\mathcal{B}$ and ii) the visit frequency register $\mathcal{V}_n$ is incremented.

5) Selection of final solution: R-SFC-MCTS repeats steps 2, 3 and 4 until the expiration of a computational budget denoted by δ. Note that δ can be based on i) a time period or ii) the maximum number of loops or iii) any other metrics fitting the problem definition and the execution environment.In a large scale scenario, R-SFC-MCTS will end without building the entire decision tree but hopefully the generated branches will include an optimized solution thanks to the previous stages. Once the δ period is expired, our proposal searches for the best branch $\mathcal{B}_b$ in the decision tree $\mathcal{T}$. This search process is optimized by R-SFC-MCTS because during the tree $\mathcal{T}$ building process we save the information related to the relevance of the branch (i.e., mapping quality) at the leaf nodes. Looking for the best solution is equivalent to finding the leaf node with the best mapping quality. The selected leaf tree-nodes denoted by n_b have to satisfy this equality:

$$\chi_{n_b} = \max_{n \in LeafNodes(\mathcal{T})} (\mathcal{R}_n^l) \quad \text{(IV.5)}$$

Then, $\mathcal{B}_b$ is the best sequence in $\mathcal{T}$ containing n_b. Note that $\mathcal{B}_b$ is unique since each node in the tree is attached to one and only one parent thanks to the tree topology. Finally, a bottom-up navigation computes the best branch $\mathcal{B}_b$.

C. Reliability support

Our proposal deals with survivability using **two stages**. First, during the embedding process, our algorithm priviledges the use of reliable physical links to minimize the impact of physical link failures, as long as the bandwidth availability constraint is respected. This is achieved via the **preventive** mechanism from the payoff function $\mathcal{R}_n^l$ (i.e., equation IV.4) defined for the decision tree search to initially embed the service function chain. In fact, the selected branch that matches with computed initial mapping is the one that maximizes the reliability based on the equation IV.5. Consequently, R-SFC-MCTS prevents the service interruption by avoiding the use of the non-reliable physical links.

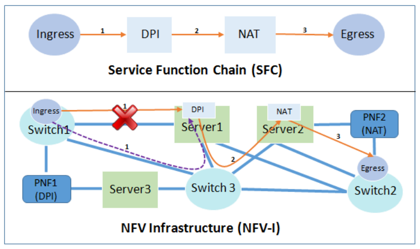


Fig. 3: The Recovery Process

Second, once a link failure impacts some virtual links, R-SFC-MCTS tries to re-embed the affected virtual links in safer physical paths while respecting the bandwidth requirement. We make use of the same initial mapping process, as described previously (i.e., Algorithm 1), to compute the new paths. This re-embedding mechanism is fast since the majority of the virtual resources are already mapped. Hence, the tree search process will be optimized since some candidates are already predetermined. We should highlight that this **recovery** mechanism can be defined based on different variants of our algorithm considering backups or not. For the first variant, in each physical link a backup proportion of its bandwidth capacity is dedicated for the recovery mechanism. Hence, the initial mapping will be performed only within the first proportion of each selected substrate link, and the remaining (i.e., second) proportion will host only the recovered virtual links after a substrate link outage. In the solution variant, without backup resources, the initial embedding as well as the recovery mechanism will use the residual bandwidth of the entire link without any separation. For the sake of clarity, we introduce Fig. 3 to illustrate the recovery result. In this figure, we note that the initial provisioning (virtual link 1) used the physical link connecting switch1 and server1. During the client service, an outage occurs on this physical link. Hence our algorithm re-embeds the virtual link 1 on the new computed path: switch1$\rightarrow$switch3$\rightarrow$server1.

The complexity of our proposal, R-SFC-MCTS, is: $O(I \times \mid Cand(v) \mid \times \mid V(\mathcal{D}) \mid \times \mid V(\mathcal{G}) \mid^2)$, where $\mid Cand(v) \mid$ is the size of the candidates set of a virtual node v (i.e., VNF or Ingress/Egress). The computational budget, $\mathcal{B}_b$, is defined here as the maximum number of loops and denoted by I. The complexity of our proposal, R-SFC-MCTS, is hence polynomial thanks to the use of the MCTS approach.

V. Performance Evaluation

In this section, we assess the performance of our proposal R-SFC-MCTS based on extensive simulations. We describe the simulation settings and the used performance metrics to conduct a comparison of the solutions with and without back up resources to address the link failures for reliable and failure tolerant VNF chain placement.

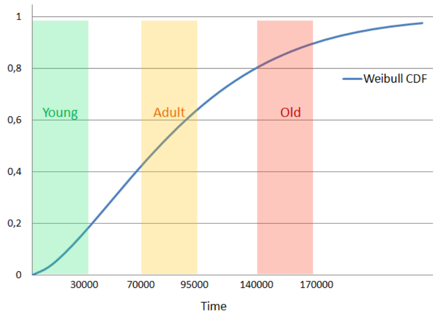


Fig. 4: Weibull CDF and Initial Ages

A. Simulation Environment

To evaluate our approach, we run simulations using realistic topologies and parameters. We used the same conditions and settings of simulation scenarios in the literature to obtain meaningful comparisons [5], [30]. Requests arrive according to a Poisson process with an average rate of 5 requests per 100 time units and each resource request has an exponential lifetime with a mean of 500 time units. These settings load gradually the NFV-I that becomes fully loaded with increasing simulation time. Performance results are averaged over all simulation experiments and each reported value is an average of 100 instances ensuring a confidence interval of 95%.

The considered NFV-I sizes vary between 100 and 2000 nodes with connectivity of 50%. The available CPU capacity per physical node and available bandwidth capacity per link are randomly drawn [150, 160] units and [100, 110] units, respectively. The SFC chains are randomly generated with a number of virtual nodes per request in the range of [10, 100]. The requested CPU capacity of each VNF was generated based on the data provided in [31], [24]. The CPU is drawn randomly in [10, 15] with random requested bandwidths in the [10, 12] interval.

Physical links failure probabilities are set using the parameters a and b of the Weibull distribution (see equation (III.1)) to obtain a lifetime for each physical equipment ($\approx 6 \times 10^5$ time units) roughly equal to 10 times the simulation duration ($\approx 6 \times 10^4$ time units). Furthermore, the proportion of adult links in the NFV-I is fixed to 20%. In our simulations, we vary the proportion of young links and consequently the old proportion to assess performance with variable infrastructure reliability. The initial age of physical links follows a uniform distribution based on defined bounds of its group (i.e., young, adult or old). Fig. 4 illustrates these bounds and the CDF modeling the Mean Time Between failure. In fact, this figure describes the initial ages of the three link types as well as the corresponding initial (i.e., simulation start) failure probability. As a result of these Weibull settings, we highlight in Fig. 5 the cumulative number of failed physical links while varying the proportion of young links from 10% to 70%. The number of occurring failures is lower when the majority of the network links are young (i.e., young rate = 50% or 70%). The number of link outages or failures is very high when most of the links are old (when the proportion of young links = 10%). These failures events serve as input to our simulation scenarios.

B. Performance Metrics

We first define the performance evaluation metrics before we report the results of the performance assessment.

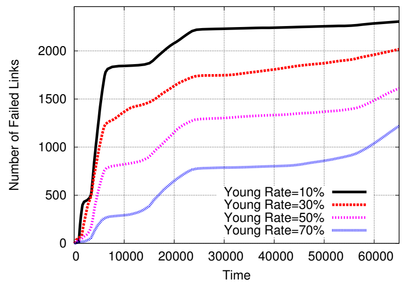


Fig. 5: Outages in Physical Links

1) $\mathbb{Q}$: is the rejection rate of the incoming requests during the simulation that corresponds to the fraction of client requests that are not accepted for physical resources shortage.

2) $\mathbb{R}$: we define $\mathbb{R}(t)$ that measures the total revenue generated by the embedded VNF requests at time t :

$$\mathbb{R}(t) = \sum_{\mathcal{D} \in \mathcal{AR}_t} \mathbb{R}(\mathcal{D}) \tag{V.6}$$

where $\mathcal{AR}_t$ is the set of accepted requests until time t and $\mathbb{R}(\mathcal{D})$ is the acquired revenue defined as in [32]:

$$\mathbb{R}(\mathcal{D}) = \left(\sum_{v \in V(\mathcal{D})} C(v) \right) * U_C + \left(\sum_{d \in E(\mathcal{D})} B(d) \right) * U_B \tag{V.7}$$

where U_C and U_B are the revenue gained by allocating a unit of computing resource and bandwidth respectively. The acceptance revenue at the end of the simulation is given by $\mathbb{R}$.

3) $\mathbb{P}$: describes the penalties induced by link failures and downtime during the recovery mechanism. In our study, we define two kinds of penalties. The first one P_1 considers the induced penalty when it is impossible to recover a virtual link. For this case the whole virtual network will be impacted and hence released. This penalty is proportional to the remaining lifetime of the impacted service function chain. Formally, for each released request denoted by $\mathcal{D}$:

$$P_1(\mathcal{D}) = \frac{T_D^d - \Delta_T}{S_D} \times \delta \times \mathbb{R}(\mathcal{D}) \tag{V.8}$$

where T_D^d is the assumed departure time of the request $\mathcal{D}$, Δ_T is the failure instant, S_D is the sojourn time of the request $\mathcal{D}$ and δ is the applied penalty. The first term in the equation represents the fraction of time the request D is not served compared with the expected failure free sojourn time.

The second penalty P_2 is the micro interruption during successful failure recovery of an impacted virtual link $d \in E(\mathcal{D})$.

$$P_2(d) = \Delta_{Rec} \times \delta \tag{V.9}$$

where Δ_{Rec} is the needed time to perform the recovery of the impacted virtual link d.

$$\mathbb{P} = \textstyle\sum_{\mathcal{D} \in \mathcal{F}} P_1(\mathcal{D}) + \sum_{d \in \mathcal{R}ec} P_2(d) \tag{V.10}$$

where $\mathcal{F}$ is the set of the non recovered requests and $\mathcal{R}ec$ is the set of the recovered virtual links.

4) $\mathbb{L}$: is the revenue loss caused by link failures during the entire simulation. It is evaluated as the ratio of induced penalties to the acceptance generated revenue. $\mathbb{L}$ describes the percentage of the penalties due to SLA violations compared to the initial earned revenue from the acceptance process:

$$\mathbb{L} = \frac{\mathbb{P}}{\mathbb{R}} \tag{V.11}$$

5) $\mathbb{G}$: is defined as the profit (final gain) the provider earns when accounting for both acceptance generated revenue and penalties (loss) caused by service interruptions. $\mathbb{G}$ is evaluated as the subtraction of the induced penalties $\mathbb{P}$ from the acceptance revenue $\mathbb{R}$:

$$\mathbb{G} = \mathbb{R} - \mathbb{P} \tag{V.12}$$

C. Simulation results

We evaluate performance of our proposal by comparing also with a baseline solution that does not implement any recovery from failures. We define a no-backup variant and backup solutions (that set a fraction of resources aside for exclusive use by failure management). For the backup variants, we vary the percentage of the dedicated backup (20%, 30% and 40%) to check our algorithm performance behavior.

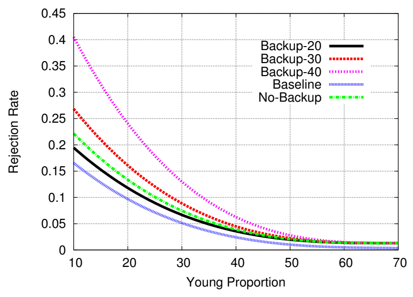


Fig. 6: Rejection Rate

*1) **Rejection rate and acceptance revenue**:* The first evaluation concerns rejection rate illustrated in Fig. 6. As intuitively expected, all algorithms reject more requests when the young population is low or when there are more unreliable links in the infrastructure. The baseline strategy has the lowest rejection rate but this is rather misleading since it is achieved at the expense of quality of experience since services on failed links are interrupted and can only resume if the consumer manages these degradations by restarting their applications and services themselves. This is equivalent to low infrastructure availability.

On the contrary the failure recovery approaches (R-SFC-MCTS variants) with and without back up will find alternate physical resources to maintain the service while minimizing as much as possible service interruptions. They will consequently use more physical links to construct back up paths for the failed virtual links and will consume more resources. The baseline, will simply violate the established SLAs and will end up with more available resources to accept future requests as seen in Fig. 7. The reported rejection rate and acceptance generated revenue results need to be analyzed carefully by realizing that the baseline will cause significant consumer churn with detrimental effect on the provider business. All the performance metrics have to be analyzed jointly to draw

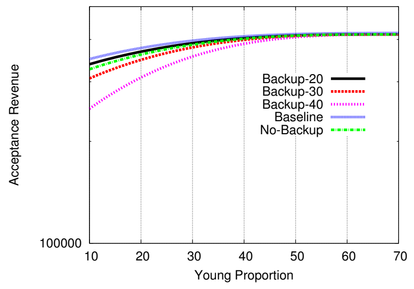


Fig. 7: Acceptance Revenue

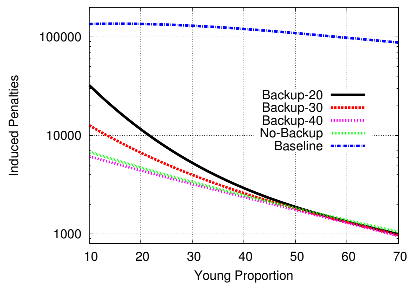


Fig. 8: Induced Penalties

reliable conclusions on the relative performance of all evaluated algorithms. Both initial gains and penalties have to be included when evaluating the final gain or achieved profit $\mathbb{G}$.

*2) **Induced penalties**:* Fig. 8 confirms the expected poor performance of the baseline algorithm (no recovery from link failures) in terms of induced penalties that is 100 times worst than the failure recovery algorithms (R-SFC-MCTS variants). The algorithm with no reserved back up resources performs best compared to the algorithms with small amounts of back up resources such as the backup-20 and backup-30 variants. To outperform the no back up solutions at least 40% of the resources need to be set aside for recovering from link failures with lower penalties. There is hence a trade-off to find for the variants with backup resources before the best overall performance can be obtained.

*3) **Revenue loss rate**:* Fig. 9 sheds some light on the required tradeoff by evaluating the revenue loss caused by penalties. The baseline solution loses more than 20% of its accepted requests generated initial revenue. This highlights the mandatory need of the recovery mechanisms. The no-backup variant outperforms the other approaches when the proportion of young links is low indicating that it is suitable when the substrate network is essentially old. When the majority of the links are reliable (young links) the failure recovery algorithms have have roughly the same performance.

*4) **Final revenue**:* Fig 10 shows the final revenue gained by the provider (or earned profit) taking into consideration what is earned from initial provisioning and what is lost in penalties due to service interruptions and SLA violations. The no-backup recovery approach achieves the best overall performance with a final additional revenue of about 20000 monetary units comparing with the backup-20 variant when the failures are very frequent in the substrate network. As a conclusion, the no-backup variant is the most appropriate to maximize revenue for the provider since it optimizes the use

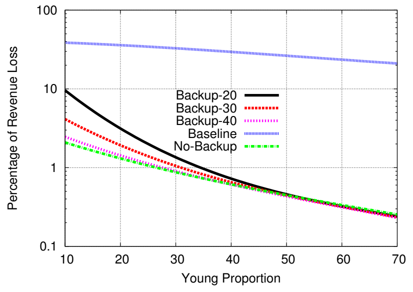


Fig. 9: Revenue Loss

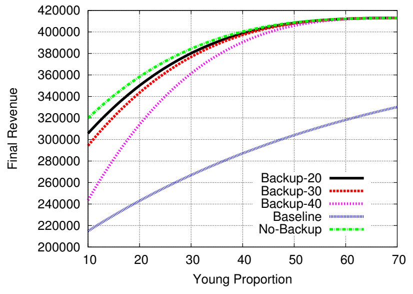
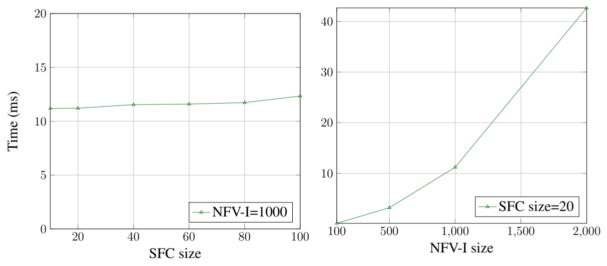


Fig. 10: Final Revenue

of physical resources. The variants that set aside resources for the failure recovery exclusive use limit the search space of possible solutions to the back-up zone only. The baseline as expected leads to very low profit compared to the failure recovery solutions.

*5) **Recovery Time**:* The last aspect that needs to be assessed is the service interruption time that has direct impact on the consumer quality of service and experience. This is evaluated using the time needed to find alternate physical paths for the failed virtual links. Since all our proposed R-SFC-MCTS variants have roughly the same recovery time, Fig. 11a depicts only one figure for all the algorithms.



(a) Varying SFC size (b) Varying NFV-I size

Fig. 11: Recovery time

In Fig. 11a, the substrate infrastructure size is fixed to 1000 and the SFC size is varied from 10 to 100. This experiment shows that our algorithm manifest "flat" recovery times (less than 13 ms for NFV-I=1000), regardless of SFC request size. The SFC size in Fig. 11b is fixed to 20 VNFs and we vary the NFV-I infrastructure size in the range [100, 2000]. The algorithms need more time to recover from link failures when increasing the infrastructure size but the time needed to find an alternate path does not exceed tens of milliseconds (below 45 ms for NFV-Is of size 2000). This recovery time is negligible compared with the time required for initial SFC placement and chaining.

Table I summarizes the ratio between the time to find a recovery solution and the time needed to find an initial embedding solution. Despite the sharply increasing recovery time, the ratio remains very low (0.13%) and this emaphsizes the efficiency of the MCTS based solution to address the link failures problem.

TABLE I: Recovery time vs Initial embedding time

NFV-I size	100	500	1000	2000
Initial embedding time (ms)	100	2150	8430	32870
Recovery time (ms)	0.13	3.21	11.2	42.66
Ratio (Percentage)	0.13%	0.14%	0.13%	0.12%

VI. Conclusion

In this paper, we proposed a reliable approach to deal with the problem of VNF placement and chaining in Cloud environments. Our proposal makes use of the Monte-Carlo Tree Search algorithm to place VNFs and simultaneously steer traffic flows across them in a reasonable time. Our approach deals with reliability using an appropriate strategy at initial embedding and another one when link failures actually occur. The first one is preventive. During the initial embedding, R-SFC-MCTS favors or selects in priority reliable links to avoid the detrimental effects of failures. Second, at link failures, our proposal reactively re-maps the failed virtual links in other substrate paths. To summarize, we addressed, in this paper, the reliable SFC chaining and placement problem in the presence of link failures. Our work shows that using MCTS as a basis for recovering from link failures is efficient when combined with strategies with no backup and backup resources and generates higher provider profit. In future work, we will enhance and extend our algorithms and strategies to deal with physical node failures, scaling and VNF migration.

References

[1] J. Metzler, 2016, "The 2016 Guide to SDN and NFV". [Online]. Available: http://www.webtorials.com/

[2] C. Technologies Inc, "The avoidable cost of downtime," *Research Report*, 2010. [Online]. Available: http://arcserve.com/

[3] G.Phillipa, J. Navendu, and N. Nachiappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 350–361, Aug. 2011.

[4] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Network and Service Management (CNSM)*, Nov 2014, pp. 418–423.

[5] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions in NFV," *CoRR*, vol. abs/1503.06377, 2015. [Online]. Available: http://arxiv.org/abs/1503.06377

[6] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[7] K.-H. Chen, D. Du, and P. Zhang, "Monte-Carlo Tree Search and Computer Go," *Springer - Advances in Information and Intelligent Systems*, vol. 251, 2009.

[8] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 1 – 14, 2016.

[9] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo, "Towards making network function virtualization a cloud computing service," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May 2015, pp. 89–97.

[10] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, Part 3, pp. 492 – 505, 2015.

[11] G. D. Forney, “The Viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[12] O. Soualah, I. Fajjari, N. Aitsaadi, and A. Mellouk, “PR-VNE: Preventive Reliable Virtual Network Embedding Algorithm in Cloud’s Network,” *IEEE Global Communications Conference (Globecom)*, 2013.

[13] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, “Adaptive virtual network provisioning,” *ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures (VISA)*, 2010.

[14] O. Soualah, I. Fajjari, N. Aitsaadi, and A. Mellouk, “A Reliable Virtual Network Embedding Algorithm based on Game Theory within Clouds backbone,” *IEEE International Conference on Communications (ICC)*, 2014.

[15] M. Rahman and R. Boutaba, “SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization,” *IEEE Transactions on Network and Service Management (TNSM)*, 2012.

[16] M. Chowdhury, M. Rahman, and R. Boutaba, “ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping,” *IEEE/ACM Transactions on Networking (TON)*, vol. 20, 2012.

[17] W. Yan, S. zhi Chen, X. Li, and Y. Wang, “RMap: An algorithm of virtual network resilience mapping,” *International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, 2011.

[18] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, Oct 2014, pp. 7–13.

[19] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, “A novel approach to virtual networks embedding for SDN management and orchestration,” in *2014 IEEE Network Operations and Management Symposium, NOMS 2014, Krakow, Poland, May 5-9, 2014*, 2014, pp. 1–7.

[20] S.-I. Lee and M.-K. Shin, “A self-recovery scheme for service function chaining,” *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 108–112, 2015.

[21] ETSI GS NFV 001: ”Network Functions Virtualisation (NFV); Use Cases”.

[22] ETSI GS NFV 003: ”Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV”.

[23] A. P. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. N. Chuah, Y. Ganjali, and C. Diot, “Characterization of failures in an operational ip backbone network,” *IEEE/ACM Transactions on Networking*, vol. 16, pp. 749–762, 2011.

[24] S. Kumar and et al., “Service function chaining use cases in data centers,” Internet-Draft, January 2016. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-sfc-dc-use-cases-04.txt

[25] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2003.

[26] J. Clausen, “Branch and Bound Algorithms - Principles and Examples,” *Depart. of Computer Science, Univ. of Copenhagen*, pp. 1–30, 1999.

[27] N. Huyn, R. Dechter, and J. Pearl, “Probabilistic analysis of the complexity of A*,” *Artificial Intell*, vol. 15, pp. 241 –254, 1980.

[28] O. Soualah, I. Fajjari, N. Aitsaadi, and A. Mellouk, “A batch approach for a survivable virtual network embedding based on Monte-Carlo Tree Search,” *IFIP/IEEE Integrated Network Management (IM)*, 2015.

[29] M. P. Schadd, M. H. Winands, M. J. Tak, and J. W. Uiterwijk, “Single-player Monte-Carlo tree search for SameGame,” *A Special Issue on Artificial Intelligence in Computer Games: AICG*, vol. 34, pp. 3–11.

[30] M. Mechtri, C. Ghribi, and D. Zeghlache, “Vnf placement and chaining in distributed cloud,” in *the 9th IEEE International Conference on Cloud Computing, June 27 - July 2, 2016, San Francisco, USA*.

[31] Q. Z. Ayyub, T. Cheng-Chun, C. Luis, M. Rui, S. Vyas, and Y. Minlan, “Simple-fying middlebox policy enforcement using sdn,” ser. SIGCOMM ’13. ACM, 2013, pp. 27–38.

[32] Y. Zhu and M. Ammar, “Algorithms for assigning substrate network resources to virtual network components,” *IEEE INFOCOM*, pp. 1–12, 2006.