A Batch Approach for a Survivable Virtual Network Embedding based on Monte-Carlo Tree Search

Oussama Soualah, Ilhem Fajjari†, Nadjib Aitsaadi and Abdelhamid Mellouk
LiSSi, University of Paris-Est Creteil Val de Marne (UPEC): 122 rue Paul Armangot, 94400 Vitry-sur-Seine, France
†VIRTUOR: 4 Residence de Galande, 92320 Chatillon, France
oussama.soualah@u-pec.fr, ilhem.fajjari@virtuor.fr, nadjib.aitsaadi@u-pec.fr, mellouk@u-pec.fr

***Abstract*—In this paper, we address the survivable batch-embedding virtual network problem within Cloud's backbone. In fact, the batch mapping of virtual networks will enhance the cumulative Cloud provider's revenue thanks to the global view of the incoming requests during a predefined time slot. Hence, the differentiation between requests can be performed and the arrival order of requests is ignored. The embedding of one virtual network is NP-hard. Adding the batch processing of the requests will further increase the complexity of the problem. In order to skirt the exponential complexity, we formulate the problem as building and researching problems within a decision tree. To resolve it, we propose a novel reliable batch-embedding virtual network strategy denoted by `BR-VNE`. It is based on Monte-Carlo Tree Search optimization method in which the upper confidence bounds can be reached in polynomial time. Based on extensive simulations, the results obtained show that `BR-VNE` outperforms the related work in terms of i) acceptance rate of virtual network requests, ii) Cloud provider's revenue and iii) rate of requests impacted by physical failures within the Cloud's backbone.**

Keywords: Cloud Computing, IaaS, Virtual Network Embedding, Reliability, Monte-Carlo Tree Search, Batch mode.

I. Introduction

Cloud Computing has known an impressive success over the last few years. This innovative technology has attracted both industrial and research communities and becomes omnipresent. The key success of Cloud Computing is virtualization [1]. This promising technology enables elasticity and improves resource utilization. Thanks to virtualization, novel adaptive strategies must be deployed in order to optimize resource allocation according to the dynamic customers' requirements. However, despite its multiple advantages, virtualization may considerably impact the performance of Cloud applications in case of physical failure. Indeed, a physical equipment outage may affect virtual instances and consequently many clients will be consequently impacted. This service interruption leads to penalties due to Service Level Agreement (SLA) violation. For instance, the average cost of unexpected physical failures in data centers is estimated to $\$5,000$ per minute in the North of America [2]. In this paper, we focus on Network as a Service (NaaS) which is a kind of Infrastructure as a Service (IaaS). With NaaS, the Cloud Provider ($\mathcal{CP}$) satisfies client requests by leasing Virtual Networks ($\mathcal{VN}$) resources (i.e., virtual routers and virtual links) within the substrate Cloud's backbone network. A client request is defined by a $\mathcal{VN}$ topology as well as the requirements of virtual resources in terms of bandwidth, processing power, memory, etc.

In this paper, we tackle the survivable $\mathcal{VN}$ **batch-embedding** problem within the Substrate Network ($\mathcal{SN}$) of Cloud's backbone. It is worth noting that in existing literature, almost all the $\mathcal{VN}$ embedding strategies consider the **online** approach which consists in handling a $\mathcal{VN}$ request as soon as it arrives. However, a $\mathcal{CP}$ prefers to handle a bundle of incoming $\mathcal{VN}$ requests periodically in order to i) efficiently exploit the residual resources in the $\mathcal{SN}$ and ii) minimize the frequency of configurations in the Cloud's backbone and hence minimize the service outage probability. In fact, a $\mathcal{CP}$ queues all the incoming client's requests during a predefined time-window, then survivable batch-embedding $\mathcal{VN}$ algorithm will be performed. In doing so, $\mathcal{CP}$ alleviates the high frequency of hardware resources' interruptions which deeply deteriorates the performance of the equipments in the $\mathcal{SN}$.

Besides, dealing with a batch-mapping of $\mathcal{VN}$s may considerably improve the $\mathcal{CP}$' revenue. Indeed, the macroscopic vision of $\mathcal{VN}$s set is exploited by the $\mathcal{CP}$ to select a subset of $\mathcal{VN}$ requests maximizing the global turnover. Note that the elected $\mathcal{VN}$s to be embedded do not necessarily follow their arrival time order. Ideally, $\mathcal{CP}$ should embed all received $\mathcal{VN}$s, whereas this may not be feasible sometimes due to the limited residual resources of substate equipments in the Cloud's backbone.

In this paper, we put forward a novel **Batch Reliable Virtual Network Embedding** (`BR-VNE`) strategy within Cloud's backbone dealing with the survivable mapping of incoming $\mathcal{VN}$ requests during a predefined time-window. The main objectives of `BR-VNE` consists in maximizing the $\mathcal{CP}$'s revenue by i) maximizing the acceptance rate of $\mathcal{VN}$ requests, ii) selecting the $\mathcal{VN}$s generating most profit and iii) minimizing the penalties impacted by physical failures in the Cloud's backbone. Due to the finite resources in the $\mathcal{SN}$, `BR-VNE` defines the sequence of $\mathcal{VN}$s that maximizes the cumulative revenue of $\mathcal{CP}$.

It is straightforward to see that the simplistic way to get the best sequence is to compute all possible combinations (i.e., sequences) of incoming $\mathcal{VN}$ requests. Whereas, this approach is computationally intractable. In order to skirt this exponential complexity, we formulate the search space by a decision tree which will be incrementally built. In the decision tree, each node models the mapping of one $\mathcal{VN}$ request. Hence, a branch in the tree depicts a sequence of $\mathcal{VN}$ embedding requests. Our objective is the generation of the optimal branch in the tree that maximizes the $\mathcal{CP}$'s revenue. Unfortunately, the construction of the whole decision tree is not conceivable, since the size of $\mathcal{VN}$ sequences is exponential. Moreover, the search process in the decision tree to find the optimal branch is computationally intractable. Beside the aforementioned exponential complexity, the mapping of **one** $\mathcal{VN}$ in each node in the decision tree is NP-hard [3] [4]. In summary, the generation of the optimal sequence is a combinational optimization problem which is NP-complete and the mapping of $\mathcal{VN}$ in each node in the sequence is NP-hard.

To overcome the above complexity, our proposal relies on the Monte-Carlo Tree Search [5] optimization method. `BR-VNE` incrementally builds the $\mathcal{VN}$ embedding decision

tree and searches for the **best** branch (i.e., sequence of $\mathcal{VN}$s). Initially the decision tree contains only the root node modeling the state of no $\mathcal{VN}$ request embedded in the $\mathcal{SN}$. BR−VNE operates as following.

First, BR−VNE selects the node among the nodes in the tree that can generate a child and simultaneously satisfy three conditions: i) its father cannot generate more children, ii) its election metric is the maximum among its brothers and iii) recursively the latter rule is satisfied for each ancestor until reaching the root. Note that the election metric controls the balance between i) exploitation of promoting branches and ii) exploration dealing with less promising branches. Then, a child (i.e., $\mathcal{VN}$ request) of the elected node is randomly generated (i.e., embedding simulation of $\mathcal{VN}$) and this process is repeated until i) all the $\mathcal{VN}$ requests are mapped or ii) the embedding process is blocked due to the congestion in the $\mathcal{SN}$. Next, BR−VNE enhances the decision tree with the newly created sub-branch. Afterwards, BR−VNE evaluates the cumulative revenue induced by the new sub-branch starting from the root of the tree to the leaf of the added sub-branch. Finally, the election metric of all the nodes in the latter branch is updated with respect to the accumulative earned income and the visit frequency. The same process is recursively repeated until the predefined computational budget is expired. The selected $\mathcal{VN}$ requests which will be mapped in the Cloud's backbone consist in the $\mathcal{VN}$ forming the branch **maximizing** the cumulative revenue. BR−VNE is validated by extensive simulations and compared with the most prominent related strategies found in existing literature. The results obtained show that our proposal outperforms related approaches in terms of i) rejection rate of $\mathcal{VN}$ requests, ii) rate of $\mathcal{VN}$s impacted by physical failures and iii) $\mathcal{CP}$'s revenue.

The remainder of the paper is organized as follows. Section II will succinctly summarize related work dealing with survivable online and batch $\mathcal{VN}$ embedding problem. In Section III, we will formulate $\mathcal{VN}$ and $\mathcal{SN}$ models as well as the reliable batch-mapping of $\mathcal{VN}$s. Subsequently, Section IV will describe the details of our proposal BR−VNE. Afterwards, simulation environment and performance evaluation will be presented in Section V. Finally, Section VI will conclude the paper.

II. Related work

In the current section, we will summarize the main related **batch-mapping** virtual network algorithms found in existing literature. Then, we will give an overview of the main survivable embedding of **one** $\mathcal{VN}$ strategies.

A. Batch-embedding of virtual networks

Few methods tackled the problem of $\mathcal{VN}$ mapping in batch mode. To the best of our knowledge, this problem is addressed only in [6] [7] [8]. Note that these algorithms do not consider reliability. Hereafter, we will briefly describe these related strategies.

In [6], the authors propounded a 2-stages algorithm denoted by Batch-Baseline (B−Base) dealing with $\mathcal{VN}$s embedding in batch mode. B−Base sorts all queued $\mathcal{VN}$ requests with respect to their revenues and handles their mapping on this order. In the first stage, B−Base maps only virtual routers for the incoming $\mathcal{VN}$ requests. Then, the virtual links are embedded based on an unsplittable approach. We notice that B−Base adopts a greedy strategy for virtual nodes and link mapping. In fact, this kind of approach may converge to a local optimum which impacts the $\mathcal{CP}$'s revenue. On the contrary, $\mathcal{CP}$ can earn more by accepting other $\mathcal{VN}$ requests with less revenue but the cumulative revenue would be higher than the one generated by accepting the requests having the maximum revenue. Moreover, the survivability of $\mathcal{VN}$ embedding is not considered.

TABLE I. Survivable embedding of one $\mathcal{VN}$

Algorithm	Centralized/Distributed	Reliability support
RMap [9]	Centralized	Link protection
CPP-VNP [10]	Centralized	Node protection
CG−VNE [11]	Centralized	Node and link protection
PR−VNE [12]	Centralized	Node and link protection
IOCM-SOUM [13]	Centralized	Regional protection
SVNE [14]	Centralized	Link protection
ORP [15]	Centralized	Link protection
K−Redundant [16]	Centralized	Node protection
Cluster [17]	Distributed	Node and link protection

In [7], the authors proposed a periodical auction-based planning of embedding process in order to handle the incoming set of $\mathcal{VN}$ requests. The propounded strategy modeled $\mathcal{VN}$s as bidders competing to win the access to the $\mathcal{SN}$ resources. In other words, each request is a selfish bidder that tries to be embedded within Cloud's backbone network. We notice that the proposed approach focuses on the competition between $\mathcal{VN}$s. However, the authors did not detail how can this approach (i.e, selfish) improves the cumulative turnover of $\mathcal{CP}$. In fact, the $\mathcal{VN}$s requests offering the maximum of bids would have more chance to be accepted regardless of the total revenue generated within the $\mathcal{SN}$.

In [8], the authors proposed a new algorithm dealing with a Multiple Virtual Network requests Embedding (MVNE). The main objectives are: i) maximizing the $\mathcal{CP}$'s revenue and ii) balancing the usage rate of resources in the $\mathcal{SN}$ in order to host more $\mathcal{VN}$ requests. MVNE algorithm is based on a Mixed Integer Program. Unfortunately, this approach is not scalable. Accordingly, this proposal cannot be exploited in a large scale $\mathcal{SN}$ network.

B. Reliable embedding of one virtual network

The proposed survivable embedding $\mathcal{VN}$ algorithms found in existing literature process only one $\mathcal{VN}$ request, this approach is called **online mode**.

We can classify the related reliable $\mathcal{VN}$ mapping methods with respect to different criteria: i) the kind of protected physical equipments (i.e., routers and/or links), ii) the use of dedicated backup, iii) the embedding of virtual links (i.e., unsplittable or splittable) and iv) distributed or centralized approach. TABLE I summarizes the main survivable virtual network online embedding algorithms found in existing literature. However, the online mode process requires a quite frequent access to the substrate network (i.e., hardware resources) which can lead to service interruption. In fact, the $\mathcal{SN}$ is a critical resource where cohabit many $\mathcal{VN}$ requests. Consequently, a physical failure or bug in the software of virtual machines may simultaneously induce to SLA violation for many clients. Hence, penalties would be introduced and $\mathcal{CP}$'s revenue would be dropped.

It is obvious to notice that scientific researchers pay more attention to the online request mapping than to batch mode. Whereas, $\mathcal{CP}$ opts for the latter mode in order to reduce the access frequency to the physical equipments. Although the batch mode provides to the $\mathcal{CP}$ a macroscopic view of incoming $\mathcal{VN}$ requests. Consequently, the most promising $\mathcal{VN}$s that generate the best cumulative turnover can be selected. In this paper, we address the survivable $\mathcal{VN}$ embedding problem in batch mode which is not addressed in existing literature.

III. Problem formalization

In this section, we will describe the reliable batch-embedding $\mathcal{VN}$ problem within Cloud's backbone network. To this aim, we will first define the substrate and virtual network models. Then, we will formulate the $\mathcal{VN}$ mapping problem.

A. Substrate and virtual networks models

The $\mathcal{SN}$ is formulated as an undirected weighted graph, denoted by $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$ where $E(\mathcal{G})$ and $V(\mathcal{G})$ are respectively the sets of substrate links and routers. Each physical router, $w \in V(\mathcal{G})$, is characterized by its i) remaining memory capacity $M(w)$, ii) residual processing power (i.e., CPU) $B(w)$, iii) type: core or access $X(w)$, and iv) reliability $\mathcal{R}^n(w,t)$ at time t. It is worth pointing out that if $X(w) = 0$, then w is a core substrate router. Otherwise, w is an access physical router (i.e., $X(w) = 1$). Each physical link (i.e., $e \in E(\mathcal{G})$) is characterized by its i) initial bandwidth capacity $\hat{C}(e)$, ii) remaining bandwidth $C(e)$, and iii) reliability $\mathcal{R}^l(e,t)$ evaluated at time t.

We assume the same model as [18] of Mean Time Between Failures (MTBF) in the $\mathcal{SN}$ which follows the Weibull distribution. Accordingly, the equipment's reliability at any time can be estimated. 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 specific parameters to the Weibull distribution. It is noteworthy that we consider heterogeneous ages classes in $\mathcal{SN}$. Hence, each substrate resource in $\mathcal{G}$ (i.e., physical router or link) has its initial age denoted by $A(w)$ and $A(e)$ respectively for physical router w and physical link e. Consequently, in the $\mathcal{SN}$ we distinguish three groups of physical resources: i) **young**, ii) **adult** and iii) **old**. Therefore, the reliability of a i) substrate link e is equal to $\mathcal{R}^l(e,t) = F(A(e) + t)$ and ii) substrate router w is equal to $\mathcal{R}^n(w,t) = F(A(w) + t)$. It is worth pointing out that the physical equipment's reliability is inversely proportional to its age. Thus, the probability of equipment's outage increases with time.

A $\mathcal{VN}$ request is also modeled as an undirected graph, denoted by $\mathcal{D} = (V(\mathcal{D}), E(\mathcal{D}))$ where $V(\mathcal{D})$ and $E(\mathcal{D})$ are respectively the sets of virtual routers and links. Each virtual router, $v \in V(\mathcal{D})$, is typified by its i) requirement in term of memory $M(v)$ and processing power $B(v)$ and ii) its type: access or core $X(v)$. Note that if $X(v) = 1$, then v is an access virtual router. Otherwise, v is a core virtual router. Furthermore, each virtual access router v is characterized by a geographical zone denoted by $\mathcal{Z}_v$. In fact, $\mathcal{Z}_v$ defines the geographic area where v should be embedded. Moreover, each virtual link $d \in E(\mathcal{D})$ is typified by its required bandwidth $C(d)$.

B. Reliable batch-embedding $\mathcal{VN}$ problem

We model the arrival rate of $\mathcal{VN}$ requests as a Poisson process with density λ. All the incoming $\mathcal{VN}$s during the predefined time window, denoted by Δ_T, are queued and their mapping is deferred to the next processing time slot. At the end of Δ_T, the $\mathcal{VN}$ batch-embedding strategy is launched. Fig. 1 illustrates the above model. Note that each $\mathcal{VN}$ has a finite lifetime and leaves the $\mathcal{SN}$ by unfreezing the allocated physical resources. Let's denote the set of received $\mathcal{VN}$ requests during Δ_T by $\Gamma = \{\mathcal{VN}_1, \cdots \mathcal{VN}_m\}$.

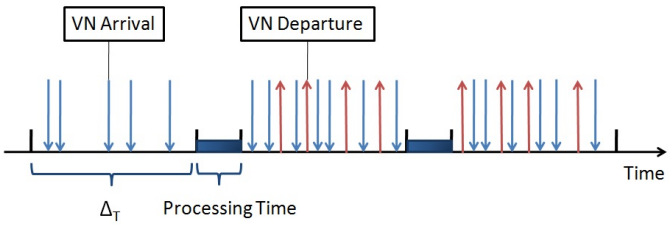


Fig. 1. Arrival and processing of $\mathcal{VN}$ requests

It is worth noting that the order of embedding $\mathcal{VN}$s directly impacts the $\mathcal{SN}$ configuration (i.e., physical residual resources). For instance, if we assume that 4 requests arrived during Δ_T, hence $\Gamma = \{\mathcal{VN}_1, \mathcal{VN}_2, \mathcal{VN}_3, \mathcal{VN}_4\}$. In one hand, the mapping of $\mathcal{VN}_1$ may consume huge $\mathcal{SN}$ resources so this would prohibit the embedding of the remaining requests. In the other hand, embedding first $\mathcal{VN}_3$ would allow the mapping of $\mathcal{VN}_4$ and $\mathcal{VN}_2$. However, starting by mapping $\mathcal{VN}_4$ cannot allow the embedding of $\mathcal{VN}_2$ and $\mathcal{VN}_3$. In summary, the embedding order strongly impacts the acceptance rate of $\mathcal{VN}$s and hence the $\mathcal{CP}$'s turnover.

For each Δ_T, the $\mathcal{VN}$ batch-embedding strategy selects the **best sequence** (i.e., order is considered) of $\mathcal{VN}$s providing the maximum of revenue. Let's denote this sequence by $\Theta_b \subseteq \Gamma$. Formally, our objective is to generate Θ_b while:

$$\forall \Theta_i \subseteq \Gamma, \mathcal{R}(\Theta_b) \geq \mathcal{R}(\Theta_i) \qquad \text{(III.2)}$$

where $\mathcal{R}(\Theta_i)$ is the sum of revenues earned by embedding the sequence of $\mathcal{VN}$s defined in Θ_i. Formally,

$$\mathcal{R}(\Theta_i) = \sum_{\mathcal{VN}_i \in \Theta_i} U(\mathcal{VN}_i) \qquad \text{(III.3)}$$

Note that $U(\mathcal{VN}_i)$ is the utility function defined in [11] and depends on the volume of resources required in terms of bandwidth, processing power, memory and lifetime.

Note that the embedding of sequence Θ_i is binding by the residual hardware resources in the $\mathcal{SN}$. Thus, Θ_i includes only $\mathcal{VN}$s that are successfully mapped in the same order defined by this sequence (i.e., Θ_i). The mapping of each $\mathcal{VN}$ is ensured by a predefined virtual network embedding strategy. The embedding problem of **one** $\mathcal{VN}$, in each node in $\mathcal{T}$, is modelized in sub-section III-C.

To compute the best sequence Θ_b maximizing the $\mathcal{CP}$'s revenue, the ideal and simplistic way is to check by simulation the embedding of all combinations (i.e., all possible orders) with regard to the $\mathcal{SN}$ resources constraints. Unfortunately, the above approach is computationally hard because of its exponential time complexity.

In order to skirt this complexity, we model the sequences of $\mathcal{VN}$s as a decision tree denoted by $\mathcal{T}$ and defined as follows. Each node in the tree represents the embedding simulation of one $\mathcal{VN}$ request. The link between two nodes in the tree exits if their associated $\mathcal{VN}$ requests are neighbors in a sequence Θ_i. The root node is abstract and represents the non-mapping of any $\mathcal{VN}$ in the Cloud's backbone. In other words, each branch in $\mathcal{T}$ corresponds to one and unique sequence Θ_i. Besides, each node in $\mathcal{T}$ located at depth d means that the mapping of $d-1$ $\mathcal{VN}$ requests is simulated since the level 1 represents the root node.

It is worth pointing out that reliable $\mathcal{VN}$ embedding problem is a non-linear and multi-objective combinatorial optimization (see sub-section III-C). It has been proved in [3]

[4] that it is NP-hard problem. Accordingly, handling the mapping of $\mathcal{VN}$ requests set in a batch mode is computationally intractable. Consequently, a smart and judicious batch-embedding $\mathcal{VN}$ strategy is mandatory to build incrementally $\mathcal{T}$ and to converge to the best branch (i.e., sequence Θ_b). In Section IV, we will detail our proposal based on the Monte-Carlo Tree Search algorithm.

C. Reliable embedding of one virtual network problem

The maximization of the acceptance rate of $\mathcal{VN}$ requests can be achieved by i) maximizing the load balancing of the $\mathcal{SN}$ and ii) minimizing the amount of physical resources allocated for each $\mathcal{VN}$ request. To reach these aims, the volume and the standard deviation of residual resources should be respectively maximized and minimized. Formally,

$$\begin{aligned} &\textbf{maximise}\left[\min\left\{C\left(e\right)\right\},\min\left\{B\left(w\right)\right\},\min\left\{M\left(w\right)\right\}\right] \\ &\textbf{minimise}\left[std\left\{C\left(e\right)\right\},std\left\{B\left(w\right)\right\},std\left\{M\left(w\right)\right\}\right] \\ &\qquad e\in E\left(\mathcal{G}\right),\ \ w\in V\left(\mathcal{G}\right) \end{aligned} \tag{III.4}$$

Moreover, in order to maximize the reliability, our objective consists in mapping the virtual **routers** and **links** within the substrate resources offering the highest reliability during the lifetime of each $\mathcal{VN}$. For each virtual router $v \in V(\mathcal{D})$ in a $\mathcal{VN}$, the reliability objective is expressed as following:

$$\textbf{maximise}\left\{\mathcal{R}^n(w^v, T_{w^v})\right\}, w^v \in V\left(\mathcal{G}\right) \tag{III.5}$$

where w^v denotes the substrate router hosting v and T_{w^v} is the age of w^v when $\mathcal{D}$ leaves the physical backbone network $\mathcal{G}$.

Besides, the objective of virtual link embedding in unsplittable manner is formally described as follow:

$$\begin{aligned} \textbf{minimize}\quad &\Big\{\prod_{e^d\in\mathcal{P}}\left(1-\mathcal{R}^l\left(e^d,T_{e^d}\right)\right)\cdot \\ &\prod_{e^d\in\mathcal{P}}\left(1-\frac{C\left(e^d\right)}{\widetilde{C}(e^d)}\right)\cdot \\ &\prod_{w^d\in\mathcal{P}}\left(1-\mathcal{R}^n\left(w^d,T_{w^d}\right)\right)\Big\}, \\ &\mathcal{P}\in \textit{Paths}(d) \end{aligned} \tag{III.6}$$

where $\textit{Paths}(d)$ denotes the set of substrate paths which can host the virtual link d, $\{e^d\}$ and $\{w^d\}$ denote the set of substrate links and routers forming the substrate path $\mathcal{P} \in \textit{Paths}(d)$. As defined above, T_{w^d} and T_{e^d} are respectively the ages of the physical router w^d and link e^d when the virtual network $\mathcal{D}$ leaves the backbone network $\mathcal{G}$. Note that the above cost function quantifies the reliability provided by substrate path's equipments (i.e., routers and links) and residual bandwidth. It is straightforward to see that reliable $\mathcal{VN}$ embedding problem is a non-linear and multi-objective combinatorial optimization.

IV. Proposal: BR-VNE strategy

In this section we detail our proposal named **Batch Reliable Virtual Network Embedding** (BR-VNE). As described above, the batch reliable mapping problem of $\mathcal{VN}$s is transformed as a tree search problem. So first of all, we will introduce the main optimization strategies found in existing literature to resolve this kind of problem. Then, we will present the main stages of our proposal BR-VNE.

Algorithm 1: BR-VNE psuedo-code

```
Inputs: SN, Γ
Output: Θ_b
Initialization (T)
while Computational Budget do
    n ← Selection-Node-Explore(T)
    B ← Generation-Sub-Branch(T, n)
    T ← T + B
    Update-Relevance(T)
Θ_b ← Selection-Best-Solution(T)
```

A. Tree search optimization algorithms

The main problem of tree search problem is the scalability. In fact, the size of solution space is finite but increases exponentially with the number of nodes in the tree. Consequently, the problem becomes computationally intractable in large-scale trees. Many research strategies were investigated to tackle its complexity such as: A* [19], Best-First-Search [20], Branch&Bound [21], etc. Unfortunately, computing time and the relevance of the solution are the main weaknesses of the aforementioned algorithms. In fact, most of them are using evaluation functions to guide the navigation process in the tree. Whereas, even if the chosen evaluation function is an admissible under-estimator one, the quality of the generated solution is not appreciated [22]. Moreover, the above methods assume that the tree search is built and the problem consists only in finding the best solution. Unfortunately, it is not easy to generate the whole solution tree at the beginning due the exponential size of the solution space.

Our proposal BR-VNE is based on Monte-Carlo Tree Search [5] (MCTS) optimization algorithm. It couples the i) tree building and ii) optimum searching process. This fits to our batch-embedding problem since the decision tree is not fully built. Indeed, MCTS combines the precision of tree search with the generality of random sampling [5]. MCTS has been applied with spectacular success in computer Games such as computer Go [23] as well as some combinatorial optimization problems [5]. Accordingly, MCTS is an efficient tool that guides a decision maker in different kind of problems.

The determination of the best sequence Θ_b of $\mathcal{VN}$s in the set Γ of incoming $\mathcal{VN}$s during the time slot Δ_T is similar to some computer games' family. Fortunately, the latter kind of problem is solved efficiently with the MCTS strategy. The suitable game category that best fits our problem description is the **Single Player** one [22]. Its objective is to maximize the player's payoff regardless to any other opponent. Consequently, the $\mathcal{CP}$ can be considered as the player who is looking to maximize the global benefit. For this reason, our proposal BR-VNE is based on MCTS. Hereafter, we will describe the main BR-VNE's stages.

B. BR-VNE description

BR-VNE incrementally builds the decision tree $\mathcal{T}$ and searches for the optimized mapping sequence Θ_b matching with the best branch in $\mathcal{T}$. The main stages of BR-VNE are: i) *Tree initialization*, ii) *Selection of node to explore*, iii) *Generation of sub-branch*, iv) *Update of nodes' relevance* and finally v) *Selection of best solution*. Algorithm 1 summarizes the pseudo-code of BR-VNE.

1) Tree initialization: In this stage, the root of decision tree $\mathcal{T}$ is created. This node does not contain any $\mathcal{VN}$ requests. The

root node means that no $\mathcal{VN}$ request within Γ is mapped in $\mathcal{SN}$. The root is an abstract node.

2) Selection of node to explore: The main objective of this stage is to select one node within decision tree to expand its branch in the next stage. In fact, the main idea is to make a balance between i) exploitation of promoting branches and ii) exploration dealing with less promising branches. In the other hand, BR−VNE should also check non promising nodes in order to avoid local optima. To do so, we associate to each node $n \in \mathcal{T}$ a relevance function denoted by ψ_n. It quantifies the attractiveness of n to be elected in Θ_b. Formally,

$$\psi_n = \bar{\mathcal{R}}_n + \alpha \cdot \sqrt{\frac{\ln(\hat{\mathcal{V}}_n)}{\mathcal{V}_n}} + \sqrt{\frac{\sum_l {\mathcal{R}_n^l}^2 - \hat{\mathcal{V}}_n \cdot (\bar{\mathcal{R}}_n)^2 + \beta}{\mathcal{V}_n}} \tag{IV.7}$$

where i) $\bar{\mathcal{R}}_n$ is the average revenue generated by all the sequences Θ_i transiting over node n, ii) $\mathcal{V}_n$ is the number of sequences passing through n, in other words it is the number of visits, iii) $\hat{\mathcal{V}}_n$ is the number of visits of n's parent (i.e., predecessor), iv) $\mathcal{R}_n^l$ is the revenue generated by the sequence Θ_l transiting over n. Note that $1 \leq l \leq \mathcal{V}_n$. Finally, α and β are constants.

It is worth noting that ψ_n is inspired from [24] which defines the **upper confidence bounds** applied to Trees. Thanks to Monte Carlo Tree Search approach, the upper bound can be reached in polynomial time [24]. It is clear that the third term [22] in the above metric quantifies a possible search deviation of node n. 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 big constant β is useful to promote rarely explored nodes. The defined metric IV.7 allows BR−VNE to balance the control between exploitation of frequently visited nodes and exploration of rarely visited nodes. Accordingly, local optimums are avoided.

BR−VNE selects one node n_s in $\mathcal{T}$ which satisfies the following constraints:

- n_s can generate new children. That means, there exists at least one $\mathcal{VN}$ request in Γ which is not mapped by all the ancestors of n_s. Formally, assuming that $n_s \in \Theta_i$, where Θ_i defines the sequence from the root till n_s, then:
$$\Gamma - \Theta_i \neq \emptyset \tag{IV.8}$$
- The father of n_s cannot generate other children. That means, the upstream levels are fully explored and cannot add any new sub-branch.
- The relevance metric ψ_{n_s} of n_s is the maximum among its brothers. Formally,
$$\psi_{n_s} = \max_{n \in Brother(n_s)} (\psi_n) \tag{IV.9}$$
- Recursively the latter rule must be satisfied for each ancestor until reaching the root. In other words, all the nodes in sequences are the best among their small family (i.e., only brothers).

3) Generation of sub-branch: Once n_s is located, a sub-branch denoted by $\mathcal{B}$ is randomly and iteratively generated. Note that randomness guarantees the convergence to the upper bound in polynomial time [24]. We assume that $n_s \in \Theta_i$ and $\mathcal{B} = \emptyset$. Then at each iteration one virtual network request $\mathcal{VN}_r$ is randomly elected in $\Gamma - (\Theta_i + \mathcal{B})$. Afterwards, the mapping simulation of $\mathcal{VN}_r$ is launched. If the embedding process is successful, then $\mathcal{VN}_r$ is added to $\mathcal{B}$. The same process is repeated until i) all virtual network requests are mapped (i.e., $\Gamma - (\Theta_i + \mathcal{B}) = \emptyset$) or ii) the embedding process fails to map the randomly elected virtual network request due to physical resources shortage. Finally, the new generated sub-branch $\mathcal{B}$ is pasted to the decision tree $\mathcal{T}$ at node n_s. Formally, $\mathcal{T} \leftarrow \mathcal{T} + \mathcal{B}$.

It is worth noting that in this paper, we make use of our previous online virtual network embedding strategy denoted by CG−VNE [11]. In fact, we have demonstrated that CG−VNE is efficient compared to the most prominent related strategies. CG−VNE relies on a cooperative game framework to reach a social optimum that maximizes $\mathcal{CP}$'s revenue. CG−VNE is formulated as two interleaved Identical Interest Games. The first game deals with the virtual routers' embedding. Since the embedding of each virtual router strongly depends on the mapping of its attached virtual links. Thus, the second game is called to map the virtual links. It is noteworthy that with both games, fictitious players are defined to cooperate in order to reach Nash Equilibrium of which we have proven the existence. We recall that Nash Equilibrium in CG−VNE matches with a social optimum. It is noteworthy that CG−VNE deals with reliability based on 2 steps. First, it urges the use of reliable resource during the mapping process in a preventive manner. Secondly, CG−VNE has a reactive mechanism to handle physical failures as soon as they occur in the $\mathcal{SN}$.

4) Update of nodes' relevance: Once the sub-brach $\mathcal{B}$ is added to the decision tree $\mathcal{T}$, BR−VNE updates the relevance function defined in equation IV.7 for all nodes in the sequence Θ_i containing $\mathcal{B}$. More precisely for all nodes $n \in \Theta_i$, i) $\bar{\mathcal{R}}_n$ and $\mathcal{R}_n^l$ are updated with respect to the cumulative revenue induced by Θ_i (i.e., the sum of virtual network revenue associated to each node in Θ_i) and ii) the visit frequency register $\mathcal{V}_n$ is incremented. In other words, BR−VNE propagates the revenue to all ancestors nodes till the root node in the new created sub-branch $\mathcal{B}$. Consequently, in doing so, the relevance function value of each node ψ_n is updated. Note that ψ_n decreases with respect to the frequency of visits and increases with the benefit induced by node n.

5) Selection of final solution: The above stages 2, 3 and 4 are repeated during a predefined period named computational budget and denoted by δ. Note that δ can be expressed by i) time period or ii) maximum number of loops. δ is calibrated with respect to the i) the arrival rate λ density of $\mathcal{VN}$ requests, ii) the duration of batch period Δ_T, and iii) the hardware performance of Cloud provider controller in which the embedding strategy is executed.

Once δ period is over, BR−VNE looks for the best branch Θ_b in $\mathcal{T}$. In order to minimize the research convergence time to this best sequence, the idea is to check only the leaf nodes in $\mathcal{T}$. In fact, the relevance function ψ of any leaf node interprets more than the induced revenue of its sequence (i.e., $\bar{\mathcal{R}}_n$) since it is visited only once (i.e., $\mathcal{V}_n = 1$). We consequently reduce the convergence time to select the best sequence. Formally, the best leaf node denoted by n_b must satisfy:

$$\psi_{n_b} = \max_{n \in LeafNodes(\mathcal{T})} (\bar{\mathcal{R}}_n) \tag{IV.10}$$

Then, Θ_b is the best sequence in $\mathcal{T}$ containing n_b. It is worth noting that Θ_b is unique since each node in the tree is attached to one and only one parent.

V. Performance Evaluation

In this section, we will gauge the performance of our proposed algorithm BR−VNE based on extensive simulations.

First of all, we describe our discrete event simulator that takes into consideration different level of reliability by distinguishing different class of substrate equipments' ages. Then, we define the performance metrics to assess our proposal and the related strategies. Afterwards, we calibrate the parameters of BR−VNE. Finally, we discuss the effectiveness of our proposal by comparing it with batch and online embedding virtual network strategies.

A. Simulation Environment

We implemented a discrete event reliable $\mathcal{VN}$ mapping simulator. To this aim, we make use of GT−ITM [25] tool to generate random $\mathcal{SN}$ and $\mathcal{VN}$ topologies. We model $\mathcal{VN}$ lifetime by exponential distribution with mean μ_L. As in [6], we set the $\mathcal{SN}$ size to 100. In this case, the ratio of access and of core routers are respectively fixed to 20% and 80%. Moreover, we set $\mathcal{VN}$ size based on discrete uniform distribution taking values between $[3, 10]$. We keep the same proportion of access virtual nodes like $\mathcal{SN}$ 20% for each $\mathcal{VN}$. It is worth mentioning that in both cases ($\mathcal{VN}$ and $\mathcal{SN}$), each pair of routers is randomly connected with a probability of 0.5.

We set the number of $\mathcal{VN}$ requests to 2000. The arrival rate λ of virtual network requests and the average lifetime μ_L of $\mathcal{VN}$s are respectively set to 4 requests per 100 and 1000 time units. We calibrate the capacity of substrate nodes and links (i.e., $B(w)$, $M(w)$ and $\hat{C}(e)$) according to a continuous uniform distribution taking values in $[50, 100]$. Furthermore, we set the requested virtual resources (i.e., $B(v)$, $M(v)$ and $\hat{C}(d)$) based on a continuous uniform distribution taking values between $[10, 20]$.

Regarding $\mathcal{SN}$ reliability, the parameters a and b of the Weibull distribution modeling the MTBF are respectively set to 25×10^4 and 1.5. It is noteworthy that a and b are calibrated in order to obtain a lifetime for each physical equipment ($\approx 6 \times 10^5$) roughly equal to 10 times the simulation duration ($\approx 6 \times 10^4$ time units). Moreover, the proportional of young equipments in $\mathcal{SN}$ is set to 30%. We study the performance with respect to the adult proportion (hence old proportion) in the $\mathcal{SN}$. Besides, physical equipment's initial age follows a uniform distribution within the bounds of its group (i.e., young, adult and old).

It is worth pointing out that each performance value of the implemented strategies is equal to the average of 15 simulations. Furthermore, simulation results are always presented with confidence intervals corresponding to a confidence level of 95%. Note that tiny confidence intervals are not shown in the following figures.

B. Performance Metrics

Hereafter, we define performance metrics used to evaluate BR−VNE.

1) Q*:* is the reject rate of $\mathcal{VN}$ requests during the simulation. In other terms, the rate of $\mathcal{VN}$s that have not been mapped in the $\mathcal{SN}$.

2) F*:* is the rate of deployed $\mathcal{VN}$s that has been impacted by physical failures during the simulation. In fact, a $\mathcal{VN}$ is impacted if at least one of its components (i.e., virtual router and/or link) is impacted by the physical failure.

3) G*:* evaluates the $\mathcal{SN}$ provider's profitability by calculating the total benefit G_1 of all accepted $\mathcal{VN}$ requests minus the paid penalties G_2 to the clients induced from the physical failures during all the simulation lifetime ($\mathrm{G} = \mathrm{G}_1 - \mathrm{G}_2$). G_1 is defined as in [26]. It depends on the amount of requested resources and the lifetime of the virtual network in the backbone network $\mathcal{SN}$. We define the penalty G_2 as the reimbursement to the clients. The cost is proportional to the residual time of a $\mathcal{VN}$ heightened by a penalty ρ. Formally, for each $\mathcal{VN}$ denoted by $\mathcal{D}$:

$$\mathrm{G}_2(\mathcal{D}) = \frac{(T - \Upsilon_T)}{T} \times \mathrm{G}_1(\mathcal{D}) \times \rho \qquad \text{(V.11)}$$

where T is the requested network virtual lifetime, Υ_T is the hosting duration in the $\mathcal{SN}$ before the physical failure, and ρ is the penalty rate. In our simulations, we set the penalty to 50%.

C. Simulation results

The current subsection is organized as follow. First, we focus on setting the suitable **Computational Budget** parameter δ of BR−VNE. In fact, it is very important to fix the fastness level of the algorithm while the quality of final solution is not deteriorated. Next, BR−VNE will be compared with the virtual network batch-embedding B−Base [6] strategy. Finally, we compare our proposal BR−VNE to the most prominent online survivable $\mathcal{VN}$ algorithm CG−VNE [11].

1) BR−VNE setting parameters: Computational Budget δ is a decisive parameter since it impacts directly the BR−VNE performance. In order to get solution with better quality, δ should be fixed to high value. In other words, BR−VNE would have enough time to search the best solution by discovering new branches in the decision tree. However, dedicating a long period to computing process may delay the client service as well as consume further $\mathcal{CP}$ resources. On the other hand, setting δ to a short period for the mapping process may converge to a local optimum. Thus, $\mathcal{CP}$'s revenue would not be maximized. In the following, we will evaluate the impact of computational budget δ on the solution quality. δ is fixed respectively to: 15, 30 and 45 seconds.

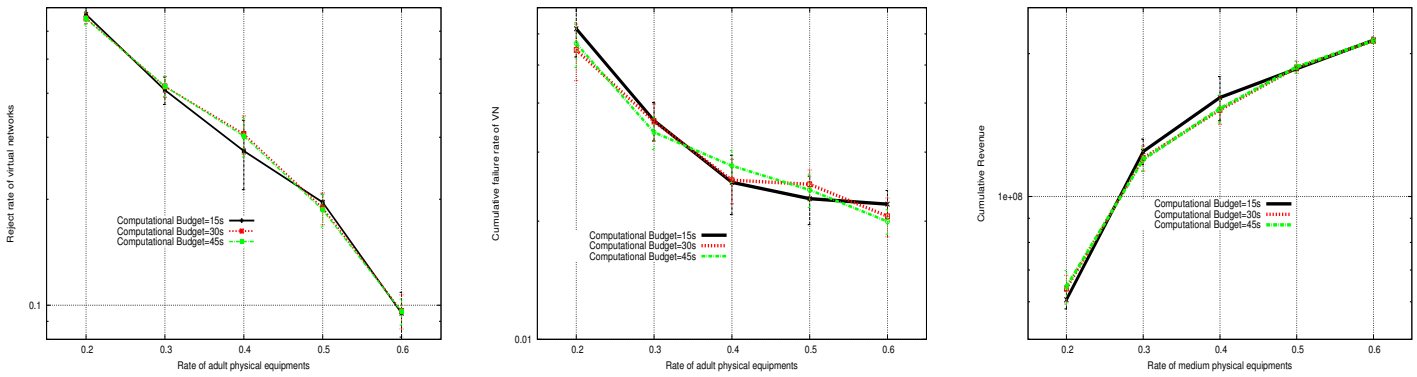
Fig. 2.(a) illustrates the reject rate Q of virtual networks. We remark that approximately the same performances are obtained even if δ is equal to 45 seconds. We can conclude that BR−VNE quickly converges to the best solution.

Fig. 2.(b) illustrates the rate F of virtual networks impacted by physical failures. It is straightforward to see that BR−VNE approximately impacts the same proportions even if the computation budget is increased. Note that the rate of adult equipment varies between 20% and 60% (i.e., rate of old equipment varies between 50% and 10%). We recall the proportion of young equipment is fixed to 30%.

Consequently, in Fig. 2.(c), the cumulative revenue of Cloud provider is roughly the same for all values of δ. In fact, the figure clearly shows that the three curves are overlapping. So we can conclude that the three values of BR−VNE approximately guarantee the same revenue to the Cloud provider. Accordingly, in the remainder simulations, we make use of the fastest value (i.e., $\delta = 15$ seconds) to compare our proposal to most prominent related strategies.

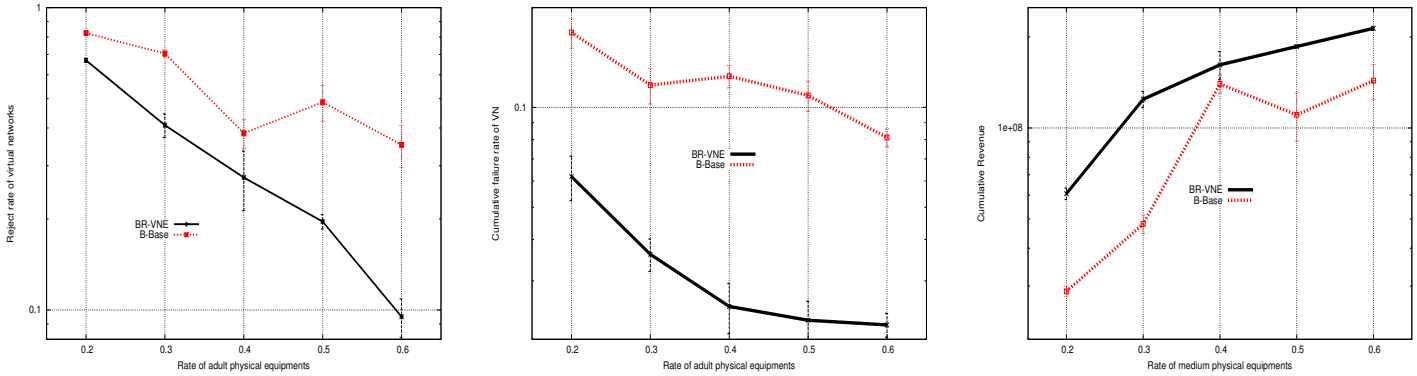
2) Comparison with batch-embedding approach: We compare our proposal BR−VNE to the best related work strategy dealing with batch-embedding $\mathcal{VN}$ B−Base [6].

Fig. 3.(a) shows that BR−VNE outperforms B−Base in term of reject rate Q of $\mathcal{VN}$s. In fact, B−Base is mapping the set of incoming $\mathcal{VN}$ requests during Δ_T following the order based on the revenue gained from each $\mathcal{VN}$ request. Unfortunately this greedy approach cripples the acceptance process of $\mathcal{VN}$ compared to our proposal. In fact, we remark that in Fig. 3.(a), BR−VNE minimizes Q with at least 18.72%



(a) Reject rate: Q (b) Blackout rate: F (c) Revenue: G

Fig. 2. BR−VNE: Calibration of computational budget δ



(a) Reject rate: Q (b) Blackout rate: F (c) Revenue: G

Fig. 3. BR−VNE versus B−Base

compared with B−Base. We can conclude that our proposal is a successful strategy which fairly manages the $\mathcal{CP}$'s backbone resources.

Fig. 3.(b) illustrates the comparison of BR−VNE and B−Base in term of blackout rate F. It is worth pointing out that B−Base does not consider physical failures. This explains the best performances of BR−VNE compared to B−Base. In fact, thanks to the reactive mechanism adopted by BR−VNE, it re-embeds impacted virtual resources by physical outages as soon as a failure occurs in the $\mathcal{VN}$. On the other hand, all impacted virtual resources will be released for B−Base algorithm. It is worth noting that even if B−Base releases more $\mathcal{VN}$ from the $\mathcal{SN}$ which means the latter is less over-loaded, but the reject rate is more important than our proposal as illustrated in Fig. 3.(a). We notice that our proposal outperforms B−Base by at least 63.25%.

Obviously, the revenue earned by BR−VNE is better than B−Base. Fig. 3.(c) illustrates the cumulative gain G. In fact, it is a direct consequence since BR−VNE outperforms B−Base in term of i) reject rate Q and ii) blackout rate F metrics. Note that if the rate of adult equipment is equal to 20% and hence old equipment is 50%, BR−VNE improves the revenue of B−Base by 110.35%.

3) Comparison with online-embedding approach: We compare our proposal BR−VNE to the most prominent online-embedding virtual network CG−VNE. In fact, we have shown in [11] that CG−VNE outperforms the related online-embedding $\mathcal{VN}$ algorithms addressing survivability.

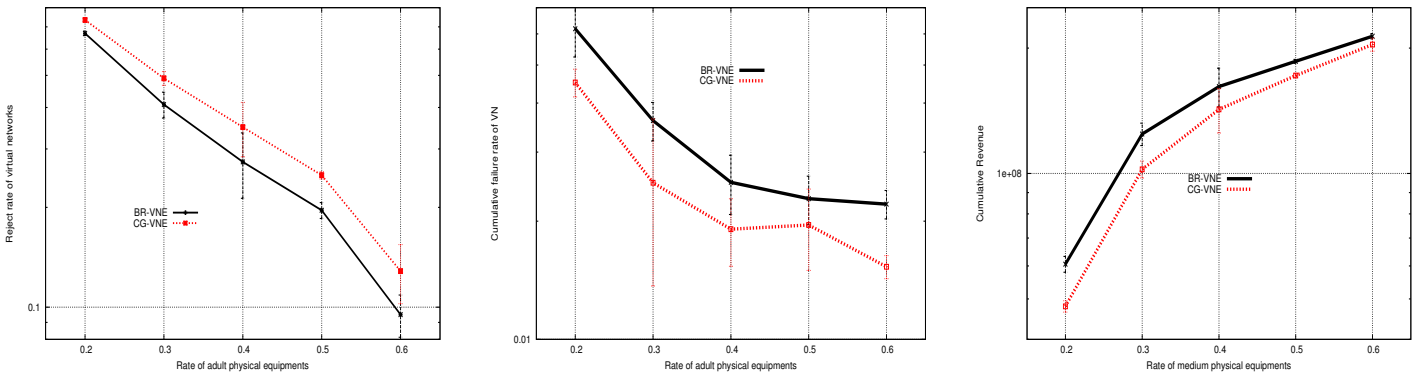
Fig. 4.(a) shows that BR−VNE outperforms CG−VNE. In fact, basing on the macroscopic view of incoming $\mathcal{VN}$s requests during the time window Δ_T, BR−VNE optimizes the choice of the accepted $\mathcal{VN}$s basing on Monte Carlo Tree search optimization algorithm. This global vision of incoming requests is missing for CG−VNE which is an online strategy. Hence, it may accept some requests crippling the embedding in the $\mathcal{SN}$. Consequently, it is straightforward to see that BR−VNE outperforms CG−VNE in term of reject rate Q. At least, the improvement is equal to 8.87%.

Fig 4.(b) depicts the comparison of blackout rate metric F between BR−VNE and CG−VNE. The latter figure shows that CG−VNE outperforms our proposal. This can be explained by two reasons. First, BR−VNE accepts more $\mathcal{VN}$ requests than CG−VNE. Hence, when a physical failure occurs in an equipment (i.e., router and/or link) within $\mathcal{SN}$, the probability to deteriorate $\mathcal{VN}$s is higher. Second, since CG−VNE and BR−VNE adopt the same reactive mechanism in order to re-embed the impacted $\mathcal{VN}$s, our proposal has less chance to find free physical resources because the residual resources of $\mathcal{SN}$ are less in BR−VNE since more $\mathcal{VN}$s are accepted .

Fortunately, the global revenue of Cloud provider is optimized with our proposal BR−VNE as illustrated in Fig. 4.(c). Thanks to batch approach, $\mathcal{CP}$ improves the turnover. In fact the balance between the revenue of accepted $\mathcal{VN}$ requests and the penalties generated by BR−VNE is positive and better than the online strategy CG−VNE. Note that if the rate of adult equipment is equal to 20% and hence old equipment is 50%, BR−VNE outperforms the revenue of CG−VNE by 26.25%.

VI. Conclusion

In this paper, we studied the problem of survivable batch-embedding virtual network. It is formulated as decision tree



(a) Reject rate: Q (b) Blackout rate: F (c) Revenue: G

Fig. 4. BR−VNE versus CG−VNE

building and searching problems which are NP-hard. We proposed a novel batch-embedding strategy named BR−VNE based on Monte Carlo Tree Search optimization algorithm. The latter can reach the upper confidence bounds in polynomial time. The proposal aims to maximize the Cloud provider's revenue and minimize the blackout rate of virtual networks caused by $\mathcal{SN}$ failures. Based on extensive simulations, our proposal outperforms the most prominent related work algorithms. In our ongoing work, we are focusing on the improvement of the randomness approach to generate the sub-branches in the decision tree.

REFERENCES

[1] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, 2010.

[2] "Calculating the cost of data center outages," Ponemon Institute, Tech. Rep., 2011. [Online]. Available: http://goo.gl/SC6Ve

[3] J. Kleinberg, "Approximation algorithms for disjoint paths problems," *PhD thesis - MIT*, 1996.

[4] S. G. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," *In Proc. IEEE Symposium on Foundations of Computer Science*, 1997.

[5] 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.

[6] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 17–29, 2008.

[7] A. Jarray and A. Karmouch, "Decomposition Approaches for Virtual Network Embedding With One-Shot Node and Link Mapping," *IEEE/ACM Transactions on Networking*, 2014.

[8] X.CHANG, B. WANG, and J. LIU, "Network State Aware Virtual Network Parallel Embedding," *IEEE Performance Computing and Communications Conference (IPCCC)*, 2012.

[9] 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.

[10] X. Liu, C. Qiao, and T. Wang, "Robust Application Specific and Agile Private (ASAP) networks withstanding multi-layer failures," *IEEE Optical Fiber Communication*, 2009.

[11] 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.

[12] ——, "PR-VNE: Preventive Reliable Virtual Network Embedding Algorithm in Clouds Network," *IEEE Global Communications Conference (Globecom)*, 2013.

[13] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," *IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.

[14] M. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," *IEEE Transactions on Network and Service Management (TNSM)*, 2012.

[15] W.-L. Yeow and C. W. U. C. Kozat, "Designing and embedding reliable virtual infrastructures," *ACM SIGCOMM workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, 2010.

[16] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," *IEEE International Conference on Communications (ICC)*, 2011.

[17] 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.

[18] 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.

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

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

[21] J. Clausen, "Branch and Bound Algorithms - Principles and Examples," *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999.

[22] 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.

[23] 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.

[24] L. Kocsis and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *Proceedings of the 17th European Conference on Machine Learning*, ser. ECML'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 282–293.

[25] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," *Proceedings of IEEE INFOCOM*, pp. 594–602, 1996.

[26] 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.