VNR-GA: Elastic Virtual Network Reconfiguration Algorithm Based on Genetic Metaheuristic

Boutheina Dab, Ilhem Fajjari‡, Nadjib Aitsaadi§ and Guy Pujolle
UPMC - University of Paris 6: 4 Place Jussieu, 75005 Paris, France
‡Virtuor: 4 residence de Galande, 92320 Chatillon, France
§LiSSi - University of Paris EST Creteil (UPEC): 122, rue Paul Armangot, 94400 Vitry Sur Seine, France
boutheina.dab@lip6.fr, ilhem.fajjari@virtuor.fr, nadjib.aitsaadi@u-pec.fr, guy.pujolle@lip6.fr

***Abstract*—Cloud Computing offers elasticity and enhances resource utilisation. This is why its success strongly depends on the efficiency of the physical resource management. This paper deals with dynamic resource reconfiguration to achieve high resource utilisation and to increase Cloud providers income. We propose a new adaptive virtual network resource reconfiguration strategy named `VNR-GA` to handle dynamic users' needs and to adapt virtual resource allocation according to the applications' requirements. The proposed algorithm `VNR-GA` is based on Genetic metaheuristic and takes advantage of resources migration techniques to recompute the resource allocation of instantiated virtual networks. In order to optimally adapt the resource allocation according to customers' needs growth, the main idea behind the proposal is to sequentially generate populations of reconfiguration solutions that minimise both the migration and mapping cost and then select the best reconfiguration solution. `VNR-GA` is validated by extensive simulations and compared to the most prominent related strategy found in literature (i.e., `SecondNet`). The results obtained show that `VNR-GA` reduces the rejection rate of i) virtual networks and ii) resource upgrade requests and thus enhances Cloud Provider revenue and customer satisfaction. Moreover, reconfiguration cost is minimised since our proposal reduces both the amount of migrated resources and their new mapping cost.**

Keywords: Network virtualization, reconfiguration, embedding problems, optimisation, Genetic metaheuristic.

I. Introduction

Cloud Computing is a promising architecture and technology enabling a flexible deployment of scaling applications. It offers elasticity and enhances resource utilisation. Indeed, Cloud Providers ($\mathcal{CP}$s) lease their substrate infrastructures on-demand to Cloud users who only pay for what they actually use. Consequently, end-users are freed from software and hardware constraints such as maintenance, updates, software license, etc. Thanks to virtualisation technology [1], elasticity and high resource utilisation can be easily achieved. Hence, many independent applications can be hosted in data centres (i.e., Software as a Service). Moreover, an application can be deployed in many geographical sites and makes use of a Virtual Network ($\mathcal{VN}$), embedded in the Cloud's backbone (i.e., Substrate Network $\mathcal{SN}$), to link all the geographical sites (i.e., Infrastructure as a Service). In this respect, an end-user can install any routing protocol within the allocated $\mathcal{VN}$ and be responsible for network administration. In other words, since virtualisation technology offers isolation, an end-user can only manage his $\mathcal{VN}$ (i.e., instance) and cannot deteriorate the rest of $\mathcal{VN}$s hosted in the $\mathcal{SN}$.

To achieve high elasticity and improve resource utilisation within the Cloud infrastructure, efficient reconfiguration techniques are mandatory and need to be set up. Indeed, in such a dynamic environment, resource reconfiguration offers flexibility that allows to re-organise scarce resources and accommodate changing customers needs. To achieve both objectives, physical resource reconfiguration can be classified into two main groups. The first, *preventive* mechanisms that aim to "tidy up" the embedded virtual resources within the $\mathcal{SN}$ in order to balance the load of physical resources and thus optimise resource usage. The second, *curative* approaches that handle dynamic user requirements in terms of hardware resources (e.g., processing power, memory and bandwidth, etc.) and adjust allocated resources in the $\mathcal{SN}$ according to the new users' needs. Indeed, depending on the running applications requirements, a client may demand whether to expand or to shrink his allocated resources within the backbone network. As a consequence, $\mathcal{CP}$ must be able to immediately adjust the capacity of each virtual network by reconfiguring the resource allocation in the $\mathcal{SN}$. Hence, the QoS required by the costumer is guaranteed.

In our previous work [2], a *preventive* greedy reconfiguration algorithm (i.e., `VNR`) was proposed. Indeed, the main goal of the proposal consists in overcoming the problem of resource fragmentation caused by the dynamic arrival and departure of $\mathcal{VN}$s. In this paper, we deal with dynamic customers' requirements. More precisely, we tackle the $\mathcal{VN}$'s bandwidth expansion problem in the $\mathcal{SN}$. To do so, we propose a *curative* reconfiguration approach that aims to offer high flexibility of resource utilisation.

The proposed algorithm is named **Virtual Network Reconfiguration based on Genetic Algorithm** `VNR-GA`. It proceeds as follows. First, the embedded $\mathcal{VN}$ concerned by the bandwidth expansion is divided into a set of star topologies, denoted by $\mathcal{SC} = \{\mathcal{SC}_i\}$. Each $\mathcal{SC}_i$ is formed by a central virtual node and virtual links that are directly attached and in which at least one requires more bandwidth. Then, `VNR-GA` randomly generates a large population of chromosomes. Based on the amount of required bandwidth and the residual substrate bandwidth (i.e., reconfiguration cost), `VNR-GA` selects the best set of feasible chromosomes and discards the rest. Note that a

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

chromosome describes a feasible sequence $\mathcal{SC}_i$ that will be sequentially migrated and this in the defined order to satisfy the new requirements. It is worth pointing out that a chromosome represents a potential solution of reconfiguration. Moreover, this chromosome is characterised by its fitness metric quantifying the reconfiguration cost. Note that a configuration cost takes into account both the number of triggered migrations and the embedding cost of the the updated $\mathcal{VN}$. Afterwards, to improve the quality of the reconfiguration solution, new populations of chromosomes are iteratively generated during a predefined number of iterations. Indeed, a population is generated by novel crossover and mutation mechanisms. In doing so, the population generated in the previous iteration is improved in terms of cost reconfiguration. At the end, the best chromosome (i.e., reconfiguration solution) obtained during all iterations is selected and then executed to satisfy the customer's demands.

To gauge the effectiveness of our proposal, we compared VNR−GA with the most prominent method found in literature denoted by SecondNet [3]. The simulation results show that VNR−GA reduces the rejection rate of i) $\mathcal{VN}$s and ii) bandwidth upgrade requests. Besides, our proposal enhances the $\mathcal{CP}$ long-term revenue compared with SecondNet strategy.

The rest of this paper is organised as follows. The next section will summarise related work focusing on virtual network reconfiguration algorithms. In Section III, we will formulate both the network model and the $\mathcal{VN}$ optimisation reconfiguration problem. Then, respectively in Section IV and Section V we will describe our reconfiguration algorithm VNR−GA and the simulation results obtained. Finally, Section VI will conclude the paper.

II. Related work

Few curative reconfiguration algorithms for virtual networks have been proposed in literature. We will hereafter summarise the prominent strategies.

In [3], the authors put forward an allocation algorithm handling incremental expansion and release of resource usage to support elasticity denoted by SecondNet. In the case of a bandwidth reservation increase between two virtual nodes, the algorithm proceeds as follows. First, they increase bandwidth reservation along the already allocated substrate path if the latter can support the new bandwidth requirement, else, they search for a new substrate path meeting the new bandwidth requirement. Then, if such a path is not found, a virtual nodes reallocation is performed using a bipartite matching algorithm proposed [4] for virtual network embedding. However, whatever the frequency of bandwidth updates, the algorithm sequentially reallocates the virtual links which can lead to useless migrations. Moreover, both node and link expansion are resolved by migrating all the concerned nodes. Hence, SecondNet strategy may lead to a poor level of performances in terms of i) reconfiguration cost and reject rates of resource upgrade requests.

In [5], the authors deal with the problem of determining dynamic topology reconfiguration for service overlay networks with dynamic communication requirement. In this context, the authors define policies to reconfigure overlay topologies. A policy is defined as a sequence of basic overlay topologies used by an overlay network in response to a communication requirements. They are characterised by a cost function combining both the reconfiguration and occupancy cost. In order to find the optimal reconfiguration policy, the proposal proceeds as follows: Assuming that the range of all communication requirements values is within a predetermined set of communication patterns: i) small and ii) large size systems. First, for small size systems, the sequence of topology transitions (i.e., policy) is formulated as a continuous time Markov decision process. Each state in the Markov model corresponds to a couple of a communication pattern with a feasible overlay. A transition in the Markov model corresponds to the cost of the reconfiguration. Then, the optimal reconfiguration policy is obtained by solving the Markov decision process. Secondly, for large size systems, the authors propose heuristic methods depending on the reconfiguration cost values: i) for low reconfiguration cost, the algorithm favours the reconfiguration (i.e., always-change strategy, ii) for high reconfiguration cost, the algorithm prohibits the reconfiguration (i.e., never change strategy) and iii) otherwise, a Cluster-Based policy is proposed where communication patterns, representing the aggregated communication requirements of all customers, are grouped into clusters. In fact, the main idea behind the aforementioned heuristics is to assign each communication pattern to the cluster with the least policy cost. Unfortunately, such an algorithm cannot be applied with a generic communication requirement. Indeed, the authors assume that all communication requirements must belong to a predefined set of communication patterns, which is not realistic and does not match necessary with customer's requirements.

In [6], the authors propose a Virtual Network Topology reconfiguration algorithm based on an enhanced traffic estimation and denoted by VNT. It is worth noting that the proposal refers to a network topology constructed over an optical layer path. VNT proceeds reconfiguration into multiple stages and its objective consists in minimising the number of reconfigured optical layer paths. Indeed, VNT heuristic first checks the usage rate of all optical layer paths. Then, for each detected congested link, a new optical link is added between two end-nodes extremities of the path containing the congested link. Each link with low utilisation rate is deleted from the topology. Unfortunately, the authors do not take advantage of node migration technique to reconfigure the topology. Moreover, reconfiguration decision is taken based on traffic estimation which even if improved, stills is prone to errors.

In this paper, we propose an elastic reconfiguration algorithm named VNR−GA. Unlike [5], our proposal is able to handle all kinds of communication requirements. Indeed, VNR−GA can be applied whatever the link requirements formulated by the owner of a virtual network (i.e., client). Moreover, unlike [3], our proposal simultaneously migrates all bandwidth upgraded virtual links in order to avoid the useless migrations

and thus minimises the $\mathcal{VN}$ reconfiguration cost. Finally, unlike [6], VNR−GA makes use of node migration in order to enhance reconfiguration performances.

III. Formulation of the Flexible $\mathcal{VN}$ Reconfiguration Problem

In this section, we will formulate the reconfiguration problem of the $\mathcal{VN}$ denoted by $\mathcal{D}$ in the substrate network ($\mathcal{SN}$) denoted by $\mathcal{G}$. To simulate real network conditions, we consider that all the physical resources (i.e., bandwidth, processing power and memory) are limited. Hence, $\mathcal{G}$ is able to host simultaneously only a finite number of $\mathcal{VN}$ requests. Consequently, an efficient reconfiguration strategy is required to enable the adjustment of substrate resources according to dynamic user's requirements.

As in our previous work [7], we model the $\mathcal{SN}$ as an undirected graph denoted by $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$ where $V(\mathcal{G})$ and $E(\mathcal{G})$ respectively represent the sets of physical routers and their connected links. Each substrate equipment, $w \in V(\mathcal{G})$, is characterised by its i) residual memory $M(w)$ and its ii) residual processing power $B(w)$. Likewise, each physical link, $e \in E(\mathcal{G})$, is typified by its i) capacity $C(e)$ and residual bandwidth $y(e)$.

As well, a $\mathcal{VN}$ request is presented 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 equipments and their virtual links. Each virtual router, $v \in V(\mathcal{D})$, is associated with the required memory $M(v)$ and processing power $B(v)$. Moreover, each virtual link $d \in E(\mathcal{D})$ is characterised by its bandwidth capacity $C(d)$.

Our problem consists in reconfiguring the virtual graph topology $\mathcal{D}$ in order to satisfy the new user's requirements in terms of bandwidth. It is worth pointing out that the reconfiguration of $\mathcal{D}$ consists in migrating some of its allocated resources while respecting the following constraints.

Let $v \in V(\mathcal{D})$ be the migrating virtual node. The substrate node $w \in V(\mathcal{G})$ can host v only if it has sufficient available resources. Formally:

$$\begin{aligned} M(w) &\geq M(v) \\ B(w) &\geq B(v) \end{aligned} \tag{III.1}$$

In this paper, our objective is to i) minimise the cost of reconfiguration and ii) maximise the $\mathcal{CP}$'s revenue. In fact, to reach the first objective we propose to minimise both the i) migration cost and the ii) embedding cost of the reconfigured elements in $\mathcal{D}$.

Let $\phi(\mathcal{D})$ denote the migration cost of the mapped $\mathcal{D}$. Similarly to [2], $\phi(\mathcal{D})$ is defined as the weighted sum of migrated virtual nodes ϕ_n and links ϕ_e. Formally,

$$\phi(\mathcal{D}) = a \cdot \phi_n(\mathcal{D}) + b \cdot \phi_e(\mathcal{D}) \tag{III.2}$$

where $0 \leq a \leq 1$ and $0 \leq b \leq 1$ are respectively the weights of ϕ_n and ϕ_e . Note that $a + b$ must be equal to 1. We recall that i) ϕ_n is equal to the number of migrated virtual nodes and ii) ϕ_e is equal to the total allocated bandwidth of the virtual link e (i.e., the sum of allocated bandwidth over the physical path).

Let $\zeta(\mathcal{D})$ denote the embedding cost of $\mathcal{D}$. As in [7], $\zeta(\mathcal{D})$ evaluates the amount of substrate bandwidth allocated for all virtual links belonging to $\mathcal{D}$. It is worth noting that $\zeta(\mathcal{D})$ depends on the length of hosting substrate paths. Formally, $\zeta(\mathcal{D})$ is expressed as follows:

$$\zeta(\mathcal{D}) = \sum_{d \in \mathcal{D}} \sum_{e \in P(d)} C(d) \tag{III.3}$$

where $P(d)$ denotes the substrate path embedding the virtual link d and e is a substrate link belonging to the path $P(d)$.

As detailed above, the optimisation problem consists in minimising reconfiguration cost. Formally:

$$minimise\,[C_{tot}(\mathcal{D}) = \alpha \cdot \phi(\mathcal{D}) + \beta \cdot \zeta(\mathcal{D})] \tag{III.4}$$

where $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ are the weights of ϕ and ζ respectively. Note that $\alpha + \beta$ must be equal to 1. It is straightforward to see that to optimise the revenue, we minimise the embedding cost of reconfigured virtual topologies. Indeed, by minimising the amount of allocated resources, the residual resources will be maximised. Hence more $\mathcal{VN}$s can be accepted and more revenue can be generated.

Given a set of virtual link bandwidth pairs R_D=$\{(d_i, C(d_i))\}_{1 \leq i \leq k}$, where $d_i \in E(\mathcal{D})$. Our objective is to find the best reconfiguration of the corresponding embedded virtual graph $\mathcal{D}$ while satisfying the new links' requirement and minimising the reconfiguration cost.

The problem outlined above is a combinatorial optimisation problem and is NP-Hard. In the next section we will propose a flexible $\mathcal{VN}$ reconfiguration algorithm based on Genetic metaheuristic called VNR−GA.

IV. Proposal: VNR−GA Algorithm

In this section, we will describe our elastic upgrade $\mathcal{VN}$ reconfiguration algorithm, VNR−GA that aims to optimally adjust the allocation of virtual resources according the dynamic users' requirements. Indeed, during the lifetime of $\mathcal{D}$, resources' shrinking or expansion may be required by the virtual topology's owner in order to respond to its new needs. These resource requirements may concern node performances (i.e., processing power, memory) or link performances (i.e. bandwidth). However, as part of this work, we assume that resource requirements are exclusively in terms of bandwidth which is the most difficult task since many substrate links are involved for each virtual link.

VNR−GA proceeds as follows. As soon as the upgrade virtual network bandwidth demands $R_D = \{(d_i, C(d_i))\}_{1 \leq i \leq k}$ arrive, VNR−GA builds the set of solution components denoted by G_D=$\{\mathcal{SC}_i\}$. In fact, $\mathcal{SC}_i$ is a star topology composed of a center virtual node with all its directly connected links in which at least one of them is in R_D (i.e., requires more bandwidth). Afterwards, with respect to R_D and G_D, VNR−GA yields the best reconfiguration solution that satisfies R_D and minimises the reconfiguration cost. To do so, an

initial population of feasible solutions (i.e., chromosomes) is put forward, then, after N_{max} iterations, during which new populations of chromosomes are created thanks to crossover and mutation operators, an optimal solution of reconfiguration, $\mathcal{S}_b$ is selected. Finally, VNR−GA performs the reconfigurations of $\mathcal{S}_b$ to satisfy customer's demands (i.e., R_D). Hereafter, we will describe the main VNR−GA components: i) *Encoding and initial population*, ii) *Fitness evaluation*, iii) *Crossover and mutation of chromosomes* and iv)*Framework of VNR−GA*.

A. Encoding and initial population

Based on aforementioned problem's formalisation, we opt for the natural encoding to represent the population's individuals. Indeed, each gene, i, of the chromosome denoted by $\mathcal{S}$ defines the identifier of the i^{th} migrated solution component. However, if the gene value corresponds to 0, it means that no solution component is migrated at this level. It is worth noting that a chromosome is expressed as follows: $\mathcal{S}$=$[i_1, i_2, ..i_k, ..., i_l]$ where $k \in [|0, |G_D|]$. As a consequence, a chromosome contains a sequence of $\{\mathcal{SC}_i\}$s migrations triggered to reconfigure the $\mathcal{VN}$ topology. Moreover, the size of a chromosome corresponds to the maximum number of sequential migrations $\{\mathcal{SC}_i\}$s. Note that the migrations of solution components are triggered sequentially and in the same order as defined in the chromosome. In other words, performing $\mathcal{S}$ is equivalent to move in order $\mathcal{SC}_{i_1}$, $\mathcal{SC}_{i_2}$, ..., $\mathcal{SC}_{i_l}$. Indeed, within a chromosome, the reconfiguration of solution component strongly affects those of next ones.

VNR−GA randomly generates a large initial population of chromosomes. Then, based the fitness of each chromosome, a proportionate selection is used. Indeed, each feasible chromosome, belonging to the initial population, is selected according to its fitness. Note that a feasible chromosome represents a solution that guarantees the reconfiguration of all updated virtual links(i.e., virtual links belonging to R_D). The fitness metric is defined below in eq. IV.5.

B. Fitness evaluation

The fitness of a chromosome is an indicator of the quality of the reconfiguration solution to satisfy the new demands of bandwidth. Since the migration is costly in terms of service interruption period. For example, in our previous work [8], we evaluated the average migration delay in testbed platform to $2ms$. Hence, we aim to minimise the migration cost of the reconfiguration. Moreover, it is worth pointing out that the mapping cost of the $\mathcal{VN}$ affects the acceptance rate of $\mathcal{VN}$s since it depends on the amount of allocated resources. Hence, we propose the following fitness metric:

$$Fitness = \frac{1}{C_{tot}(\mathcal{D}) + \epsilon} \tag{IV.5}$$

where $0 < \epsilon \ll 1$.

C. Crossover and mutation of chromosomes

Given that VNR−GA is based on Genetic metaheuristic, its effectiveness strongly depends on some parameters such as size of population, crossover and mutation operators. Hereafter, we will detail crossover and mutation operators proposed for VNR−GA algorithm.

1) Crossover operator: Most crossover operators that are proposed in literature [9] (e.g., single point crossover, two points crossover and uniform crossover) are much more adapted to binary-encoded populations than natural-encoded populations.

Based on [10], we adopt an improved crossover operator that is adapted to our reconfiguration problem. It is defined as follows. Given a pair of father chromosomes $\mathcal{S}_1 = [i_1, i_2, \cdots, i_l]$ and $\mathcal{S}_2 = [j_1, j_2, \cdots, j_l]$, we define the notion of similitude model as $Sim(\mathcal{S}_1, \mathcal{S}_2) = [k_1, \cdots, k_l] = \mathcal{S}_1 \times \mathcal{S}_2$. Note that $\mathcal{S}_1 \times S_2 = Sim$ is a chromosome. Namely, if the i^{th} gene of $\mathcal{S}_1$ (i.e., $\mathcal{SC}_i$) is equal to the i^{th} gene of $\mathcal{S}_2$ then Sim's i^{th} gene takes 1. Otherwise, it is set to 0.

The *Hamming* distance of father individuals $\mathcal{S}_1$ and $\mathcal{S}_1$ is defined as the number of genes corresponding to 1 in Sim, denoted by $H(\mathcal{S}_1, \mathcal{S}_2)$. The improved crossover operator depends on the Hamming distance H. It is defined as follows: If $H(\mathcal{S}_1, \mathcal{S}_2) \geq 2$, then the crossover operator performs a simple point crossover from the point $\alpha(\mathcal{S}_1, \mathcal{S}_2)$. Otherwise, the crossover is not performed. Note that $\alpha(\mathcal{S}_1, \mathcal{S}_2)$ is the effective distance between $\mathcal{S}_1$ and $\mathcal{S}_2$ and it is defined as the distance from the first 1 to the last 1 of the chromosome Sim. It has been proved in [10] that the above method generates new chromosomes which are distinct (i.e., no twins).

2) Mutation: An improved mutation operator is proposed to enhance the quality of generated chromosomes. The main idea behind this proposal is to replace worst existing chromosomes by new ones offering better fitness. To do so, genes (i.e., $\mathcal{SC}_i$) causing redundant links migration are discarded from the chromosome and replaced by the value 0. In other words, in one chromosome, we always favour $\mathcal{SC}$s with disjoint links. Consequently, by removing useless link migrations, the cost of migration will be reduced and so will the reconfiguration cost. Note that the mutation process is launched every f iterations in order to prevent VNR−GA from falling into local extreme.

D. Framework of VNR−GA

In this section, we will describe the framework of VNR−GA.

VNR−GA generates N-sized initial populations using *Roulette Wheel selection technique* [9]. Indeed, the quality of each chromosome is evaluated thanks to the fitness function defined in IV.5. We propose to migrate the centre v of the star moving candidate $\mathcal{SC}_i$ to a less loaded area of the substrate network while respecting the bandwidth constraint. We recall that migrating a star component requires the embedding of all attached links while migrating the centre virtual node. To do so, we evaluate the quality of each substrate node, $w \in \mathcal{SN}$ while considering its residual resources (i.e., $M(w)$ and $B(w)$ and those of its direct paths to the hosting nodes of v's neighbours. Further details about SC's migration can be found in our previous work [2].

Afterwards, during the following iterations, new chromosomes, $\mathcal{S}_k$, are generated using the above crossover operator.

Algorithm 1: VNR−GA

Input: $\mathcal{D}$, $\mathcal{G}$
Output: Reconfiguration of $\mathcal{D}$ in $\mathcal{G}$
Arrival of R_D=$\{(d_i, C_i)\}$
$G_D \leftarrow$ Generation-Solution-Components($\mathcal{D}$)
$\mathcal{S}_b \leftarrow$ Find Best reconfiguration(G_D,R_D)
for $i \in \mathcal{S}_b$ **do**
    Migrate SC_i

Algorithm 2: Best reconfiguration

Input: R_D, G_D
Output: $\mathcal{S}_b$=$[i_1, i_2, ..., i_l]$ the best reconfiguration solution (i.e., chromosome)
$\mathcal{P}_0 \leftarrow$ InitialPopulationSelection(G_D,R_D,N) // N= the size of the population
$\mathcal{P} \leftarrow \mathcal{P}_0$
for $i = 1$, $i \leq N_{max}$, $i++$ **do**
    for $j = 1$, $j \leq N$, $j++$ **do**
        for $k = 1$, $k \leq N \wedge j \neq k$, $k++$ **do**
            $\mathcal{P} \leftarrow \mathcal{P} \cup$ Crossover($\mathcal{S}_j$,$\mathcal{S}_k$)
    $\mathcal{P} \leftarrow$ PopulationSelection($\mathcal{P}$, N)
    if *(i % f)* **then**
        for $j = 1$, $j \leq N$, $j++$ **do**
            $\mathcal{P} \leftarrow \mathcal{P} \cup$ Mutate($\mathcal{S}_j$)
        $\mathcal{P} \leftarrow$ PoulationSelection($\mathcal{P}$, N)
$\mathcal{S}_b \leftarrow$ Select-Best-Solution($\mathcal{P}$)

Indeed, such chromosomes transformation aims to obtain new promising individuals that offer better fitness evaluation. We recall that every f iterations, a mutation transformation is performed to create a different generation of chromosome.

Finally, at the end of N_{max} iterations, the best chromosome is selected and the reconfiguration of $\mathcal{D}$ is performed. The VNR−GA strategy is summarised in Algorithms 1 and 2.

V. Performance Evaluation

In this section, we evaluate the level of performance of our proposed $\mathcal{VN}$ reconfiguration strategy VNR−GA and compare it with respect to the most prominent strategy found in literature, SecondNet. Note that our proposal cannot be compared with Deterministic−VNR [5] since the latter requires specific communication patterns to be applied.

To study the performance of our strategy VNR−GA, we designed and developed a discrete-event simulator. For comparison purposes, VNR−GA and SecondNet are incorporated into our previous VNE−AC embedding algorithm [7].

A. Simulation Environment

$\mathcal{SN}$ topologies are generated with size of 100 nodes using GT−ITM tool. Each pair of nodes is randomly connected with a probability of 0.5. In accordance with our previous work in [7] [11], i) the arrival of $\mathcal{VN}$ requests follows a Poisson Process distribution with rate λ and ii) $\mathcal{VN}$ lifetime is modelled by exponential distribution with mean μ_a. As stated in [11] [7] [12] [13], we make use of the following benchmark scenario. Similar to $\mathcal{SN}$s, $\mathcal{VN}$s are randomly generated using GT−ITM tool. The size of $\mathcal{VN}$s is a uniform variable taking values in $[2, 10]$. Note that the arrival rate λ and the average lifetime μ_a of $\mathcal{VN}$s are fixed respectively to 4 requests per 100 time unit and 1000 time units. The capacity of substrate nodes and links (i.e., $M(w)$, $B(w)$ and $C(e)$) are uniformly distributed in $[50, 100]$. Similarly, the required virtual resources (i.e., $M(v)$, $B(v)$and $C(d)$) are set according to a continuous uniform distribution, varying in $[10, 20]$. The rate of $\mathcal{VN}$s requiring upgrade of bandwidth, θ, follows a Bernoulli distribution varying between $[25\%, 75\%]$ of the incoming virtual network requests. It is worth noting that this rate expresses the percentage of updated $\mathcal{VN}$s among the accepted ones. For updated $\mathcal{VN}$s (i.e., $\mathcal{D}$), the number of upgraded virtual links is modelled by a discrete uniform distribution taking integer values in $[1, |E(\mathcal{D})|\ ]$. The rate of bandwidth upgrade for a virtual links is set according a uniform distribution taking real values in $[10\%, 100\%]$ of the initial required bandwidth. The number of incoming $\mathcal{VN}$ requests is set to 2000. Additional setting parameters in our experiments are listed in Table I, where i) N_{max} is the number of iterations set for our genetic based algorithm, ii) N is the size of population and iii) μ is the mutation rate.

TABLE I
VNR−GA PARAMETERS SETTING

Parameter	Signification	Value
N_{max}	The number population generations	20
N	The size of population	20
μ	The mutation frequency	25%

Moreover, all numerical results are calculated with confidence intervals corresponding to a confidence level of 95%. Tiny confidence intervals are not shown in the following figures.

B. Performance Metrics

In this section, we will define the performance metrics used to evaluate our proposal.

1) $\mathbb{Q}_v$*:* is the reject rate of $\mathcal{VN}$ requests.

2) $\mathbb{R}(\mathcal{D})$*:* is the revenue produced by a $\mathcal{VN}$ request $\mathcal{D}$. In accordance with [14] [11], $\mathbb{R}(\mathcal{D})$ is expressed as follows:

$$\mathbb{R}(\mathcal{D}) = \sum_{v \in V(\mathcal{D})} B(v) + \sum_{v \in V(\mathcal{D})} M(v) + \sum_{d \in E(\mathcal{D})} C(d) \quad \text{(V.6)}$$

3) $\mathbb{Q}_u$*:* is the reject rate of resource upgrade requests.

4) $\zeta(\mathcal{D})$*:* is the mapping cost of a $\mathcal{VN}$ request $\mathcal{D}$. It is formally defined in III.3.

5) C_{tot}*:* is the cost of reconfiguration as defined in equation III.4. We set $\alpha = \beta = \frac{1}{2}$.

TABLE II
VNR−GA PERFORMANCES

Reconfiguration strategy	VNR−GA	SecondNet
Reject rate (%)	2.73	7.25
$\mathcal{CP}$'s revenue ($\times 10^4$)	52.56	48.12
Reconfiguration cost ($\times 10^4$)	27.12	37.02

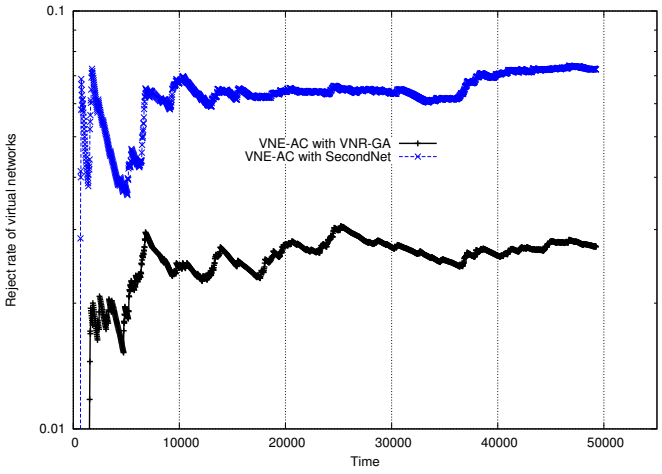

Fig. 1. Rejection rate of $\mathcal{VN}$s - $\mathbb{Q}_v$

C. Evaluation Results

In this section, we will gauge the effectiveness of our proposal VNR−GA while comparing it to the prominent related strategy SecondNet. In Table II, we compare VNR−GA and SecondNet with respect to the main performance metrics (i.e., reject rate, revenue and reconfiguration cost). It is clear to see that our reconfiguration strategy, VNR−GA, significantly outperforms SecondNet. Indeed, with VNR−GA, the embedding algorithm VNE−AC rejects only 2.73% of $\mathcal{VN}$s which considerably enhances the $\mathcal{CP}$'s revenue. However, with SecondNet, VNE−AC roughly rejects three times more $\mathcal{VN}$s, which unfavorably impacts the $\mathcal{CP}$'s income (7% less of revenue). Besides, our proposal minimises the reconfiguration cost which is crucial to guarantee the required QoS.

Fig. 1 depicts the reject rate of $\mathcal{VN}$s, $\mathbb{Q}_v(t)$, during the simulation. It is clear to see that, with VNR−GA, the embedding strategy VNE−AC maintains a low level of rejection rate ($\approx 2.7\%$). Besides, compared to SecondNet, VNR−GA considerably reduces the reject rate of $\mathcal{VN}$s.

An important metric to consider when evaluating the performance of a reconfiguration-strategy is the generated reconfiguration cost of $\mathcal{VN}$s. We recall that migrating virtual nodes and links is costly since it causes the disruption of the service. The latter may unfavourably impact the running applications (e.g., VoIP, video, etc). As depicted in Fig. 2, VNR−GA achieves a low reconfiguration cost. Such a result is predictable, since one main objective of our reconfiguration strategy consists in reducing the reconfiguration cost. By reducing the number of migrated virtual resources (i.e., nodes and links) and minimising their corresponding mapping cost, VNR−GA optimises the reconfiguration cost. Indeed, as illustrated in Fig. 2, our reconfiguration algorithm notably reduces the reconfiguration cost compared with SecondNet. In fact, the results show that VNR−GA reduces the reconfiguration cost roughly two times more ($\approx 1.7\%$) than SecondNet.

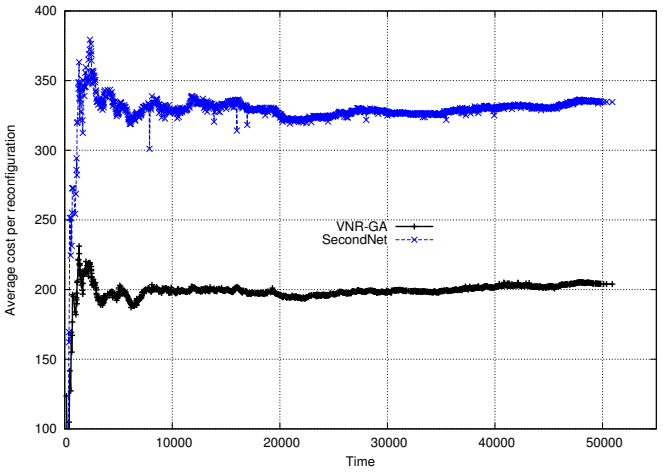

Fig. 2. Average reconfiguration cost - C_{tot}

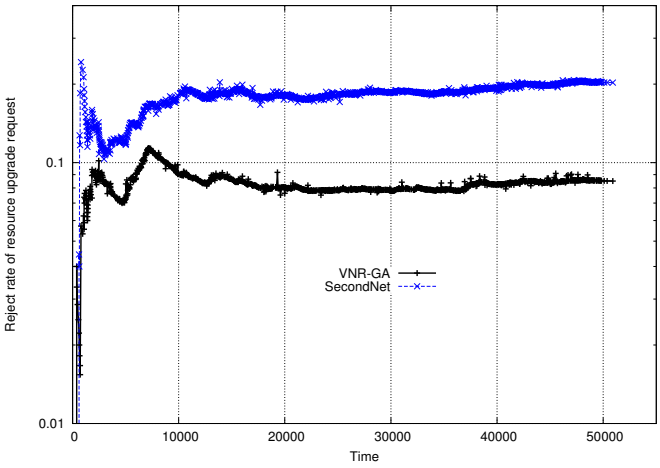

Fig. 3. Reject rate of resource upgrade requests - $\mathbb{Q}_u$

A crucial and important performance metric to evaluate the output of our proposal is the reject rate of resource upgrade requests, $\mathbb{Q}_u$. This metric evaluates the effectiveness of VNR−GA and its ability to satisfy customer's changing needs. Fig. 3 shows that VNR−GA notably minimises $\mathbb{Q}_u$ compared with SecondNet. The rationale behind this can be explained by the fact that our strategy aims to optimise the $\mathcal{VN}$ reconfiguration. Indeed, VNR−GA migrates the star moving candidates with congested links to a less loaded area of the substrate network. Consequently, our proposal balances the load and favours the acceptance of new incoming upgrade requests.

In Fig. 4, we vary the rate of bandwidth upgrade required

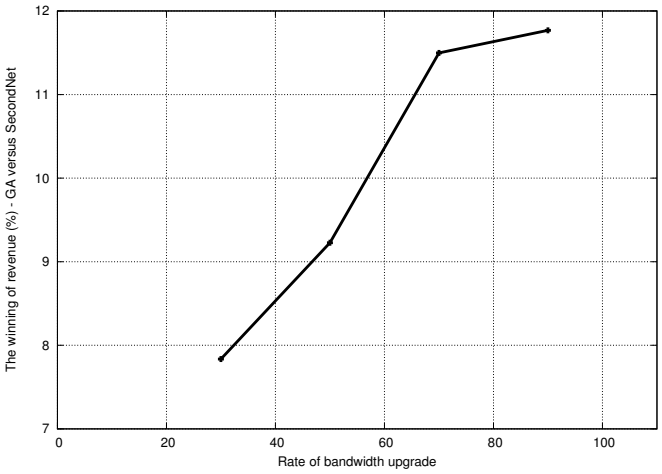

Fig. 4. Impact of bandwidth upgrade rate on $\mathbb{R}$

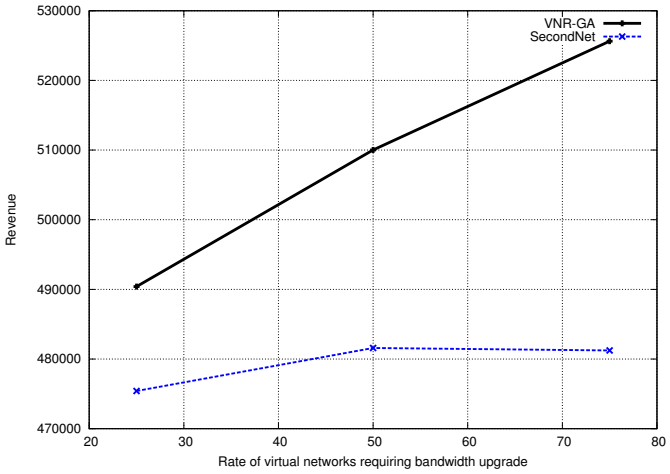

Fig. 5. Impact of θ on $\mathbb{R}$

by $\mathcal{VN}$s and we illustrate the gain of our proposal VNR−GA's revenue compared to SecondNet. It is straightforward to see that our proposal increases the revenue at least by 7% even the rate of bandwidth amount increases. We can conclude that our proposal outperforms SecondNet.

In Fig. 5, we evaluate the robustness of VNR−GA compared to SecondNet with respect to the rate of $\mathcal{VN}$s requiring upgrade of bandwidth (i.e., θ). It is clear to see that our proposal VNR−GA offers higher $\mathcal{CP}$'s revenue than SecondNet and this whatever the value of θ. Moreover, it is worth noting that VNR−GA notably enhances the revenue while the rate of clients asking bandwidth upgrade increases. Indeed, with $\theta = 75\%$ upgrade requests, the revenue is increased by 7% (i.e., $49.09 \times 10^4 \rightarrow 52.56 \times 10^4$) compared to the revenue generated with θ equals to 25%. However, SecondNet increases the revenue only by 1% (i.e., $47.54 \times 10^4 \rightarrow 48.13 \times 10^4$).

VI. Conclusion

In this paper, we studied the problem of virtual resource reconfiguration within Cloud infrastructure. We put forward a novel elastic $\mathcal{VN}$ reconfiguration algorithm called VNR−GA that aims to accommodate changing customers needs. It is based on genetic metaheuristic. Our objective is to maximise the $\mathcal{CP}$'s revenue while minimising the reconfiguration cost. The results obtained show that VNR−GA i) optimises the resource migration ii) guarantees a low mapping cost and iii) enhances the $\mathcal{CP}$'s income.

Acknowledgment

This research was supported by the World Class University program funded by the Ministry of Education, Science and Technology through the National Research Foundation of Korea (R31-10100).

References

[1] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Communications Magazine*, vol. 47, 2009.

[2] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNR Algorithm: A Greedy Approach For Virtual Networks Reconfigurations," in *GLOBECOM*. IEEE, 2011, pp. 1–6.

[3] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: a data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 15:1–15:12.

[4] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Bipartite graph partitioning and data clustering," in *Proceedings of the tenth international conference on Information and knowledge management*, ser. CIKM '01. New York, NY, USA: ACM, 2001, pp. 25–32.

[5] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 2006, pp. 1–12.

[6] Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, S. Kohei, and M. Murata, "Gradually reconfiguring virtual network topologies based on estimated traffic matrices," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 1, pp. 177–189, 2010.

[7] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "Adaptive-VNE: A flexible resource allocation for virtual network embedding algorithm," *IEEE GLOBECOM*, 2012.

[8] I. Fajjari, M. Ayari, O. Braham, G. Pujolle, and H. Zimmermann, "Towards an autonomic piloting virtual network architecture," *IFIP International Conference on New Technologies, Mobility and Security - NTMS*, pp. 1–5, 2011.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[10] Z. Qi-yi and C. Shu-chun, "An improved crossover operator of genetic algorithm," in *Computational Intelligence and Design, 2009. ISCID '09. Second International Symposium on*, vol. 2, 2009, pp. 82–86.

[11] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual Network Embedding Algorithm based on Ant Colony Metaheuristic," *IEEE ICC*, 2011.

[12] M. Chowdhury, F. Samuel, and R. Boutaba, "PolyViNE: policy-based virtual network embedding across multiple domains," in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, ser. VISA '10. New York, NY, USA: ACM, 2010, pp. 49–56.

[13] V. V. Vazirani, *Approximation algorithms*. Springer, 2001.

[14] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," *IEEE INFOCOM*, pp. 783–791, 2009.