VNE-AC: Virtual Network Embedding Algorithm based on Ant Colony Metaheuristic

Ilhem Fajjari[§†], Nadjib Aitsaadi[‡], Guy Pujolle[†] and Hubert Zimmermann[§]
[§]Ginkgo Networks: 2 A Rue Danton, 92120 Montrouge, France
[‡]HIPERCOM – INRIA: Domaine de Voluceau - Rocquencourt - B.P. 105, 78153 Le Chesnay Cedex - France
[†]UPMC - University of Paris 6: 4 Place Jussieu, 75005 Paris, France
ilhem.fejjari@ginkgo-networks.com, nadjib.aitsaadi@inria.fr, guy.pujolle@lip6.fr, hubert.zimmermann@ginkgo-networks.com

***Abstract*—In this paper, we address virtual network embedding problem. Indeed, our objective is to map virtual networks in the substrate network with minimum physical resources while satisfying its required QoS in terms of bandwidth, power processing and memory. In doing so, we minimize the reject rate of requests and maximize returns for the substrate network provider. Since the problem is NP-hard and to deal with its computational hardness, we propound a new scalable embedding strategy named `VNE-AC` based on the Ant Colony metaheuristic. The intensive simulations and evaluation results show that our proposal enhances the substrate provider's revenue and outperforms the related strategies found in current literature.**

Keywords: Network virtualization, Embedding problem, Ant colony metaheuristic.

I. Introduction

An ambitious vision of future Internet networks would include Network Virtualization. This new paradigm consists of leasing and sharing the Substrate Network ($\mathcal{SN}$) infrastructure between several Virtual Network ($\mathcal{VN}$) providers in the aim of enhancing the profitability of the physical resources. However, few studies have dealt with practical challenges such as virtual network embedding, resource scheduling, mobility, interoperability, privacy, etc. [1], [2].

In this paper, we investigate the issue of $\mathcal{VN}$ embedding in which network performance and the $\mathcal{SN}$ provider's economic returns depend strongly on its success. Each virtual node must be assigned to only one substrate node. As well, each virtual link must be mapped into a substrate path. Specifically, our objective is to increase the $\mathcal{VN}$ provider's revenue by reducing the reject rate of $\mathcal{VN}$ requests, while also satisfying the required physical resources in terms of bandwidth, power processing, and memory. The mapping of virtual nodes is proved to be NP-hard in [3]. Additionally, the virtual link assignment can be formulated as an unsplittable flow or a multi-way separator problem and is thus also NP-hard [4], [5].

Since the $\mathcal{VN}$ embedding problem is computationally intractable, the optimal solution could only be generated in small-sized instances of $\mathcal{VN}$ and $\mathcal{SN}$. In this respect, we propose a new scalable $\mathcal{VN}$ mapping strategy based on a Max-Min Ant System metaheuristic (Ant Colony algorithm) [6], denoted by `VNE-AC`. The main idea behind our proposal is to take inspiration from the behaviour of collective ants in finding the best path between their nest and a food source. To do so, `VNE-AC` first divides the $\mathcal{VN}$ request into a set of sub-problems, denoted by solution components which are sorted and then sequentially assigned to form the sequence of mapping transitions. Afterward, `VNE-AC` launches the artificial ant colony, where ants perform a parallel search. To this end, each ant iteratively builds a piece of the solution (i.e. transition) where each solution component is embedded according to the available resources and the artificial pheromone trail in the $\mathcal{SN}$. It is worth noting that each substrate node is associated with a pheromone trail value for each transition. Once all the candidate solutions (i.e. full $\mathcal{VN}$) are mapped, `VNE-AC` updates the pheromone trail in the $\mathcal{SN}$. The pheromone trail is evaporated in all substrate nodes. Nevertheless, it will also be reinforced in the contributing substrate nodes to build the best mapping topology. Indeed, the solution component is likely to be embedded in the substrate node with the highest pheromone trail value. Finally, the ants repeat the same process, during a predefined number of iterations, to improve the solution according to the new trail values. To gauge the effectiveness of our proposal, we compared the `VNE-AC` algorithm to some prominent existing proposals found in related literature: `VNE-Least` [7], `VNE-Cluster` [7], `VNE-Subdividing` [7], and `VNE-Greedy` [8]. The simulation results obtained show that `VNE-AC` outperforms the related algorithms in terms of $\mathcal{VN}$ reject rate, mapping cost, request revenue, and resource utilization.

The reminder of this paper is organized as follows. The next Section will summarize related work addressing the $\mathcal{VN}$ embedding problem. In Section III, we will formulate both the network model and the $\mathcal{VN}$ mapping problem mathematically. Then, we will describe our mapping algorithm, `VNE-AC`, and the simulation results obtained in Section IV and Section V respectively. Finally, Section VI concludes this paper.

II. Related Work

Few research papers have studied the $\mathcal{VN}$ embedding problem. In this section, we will outline the main proposals found in existing literature.

In [7], the authors propose three $\mathcal{VN}$ assignment algorithms denoted by `VNE-Least`, `VNE-Cluster`, and `VNE-Subdividing`. The first method, `VNE-Least`, treats virtual nodes and links mapping separately. Thus, substrate and virtual nodes are sorted according to their stress and connectivity degree respectively. Then, `VNE-Least` assigns the virtual node with highest connectivity degree to the least stressed substrate node recursively until all the virtual nodes are embedded. Thereafter, `VNE-Least` makes use of the shortest distance algorithm to connect the mapped virtual nodes. On the other hand, the `VNE-Cluster` and `VNE-Subdividing` algorithms take into account the substrate link load when selecting the substrate nodes. `VNE-Cluster` assigns the virtual nodes in a similar manner to `VNE-Least`, except that nodes' stress and path distance are replaced by the newly defined metrics. The final proposal, `VNE-Subdividing`, subdivides the $\mathcal{VN}$ request into star topologies so as not to deal with the request as one whole unit. Next, the stars are mapped sequentially using `VNE-Cluster`. However, the authors assume that

the resources are unlimited in the $\mathcal{SN}$ which is not a realistic assumption. Thus, the $\mathcal{VN}$ request reject rate is not evaluated in the paper.

In [8], the authors proposed the `VNE-Greedy` virtual network embedding algorithm. In this system, the substrate and virtual nodes are sorted according to the available and requested resources respectively. Then, the virtual node with the highest resource request is assigned to the substrate node containing the largest available resource metric value recursively until all the virtual nodes are mapped. Next, the nodes are connected using the K-shortest paths algorithm. `VNE-Greedy`'s main drawback is the substrate path building algorithm. Indeed, the shortest path algorithm does not consider the congested $\mathcal{SN}$ links, which implies an increase of hot-spots in the $\mathcal{SN}$ and an increase in request reject rate. Moreover, the virtual nodes and links mapping are not coordinated.

The authors of [9] model the $\mathcal{VN}$ as a directed graph with two types of nodes: access and core. The required $\mathcal{VN}$ resources are defined in terms of the expected traffic, which is expressed as an upper limit on allowed traffic between all access node pairs. The authors' objective is to calculate the minimum request bandwidth for the $\mathcal{VN}$ according to the access nodes' upper limit of traffic. Nonetheless, the weakness in the proposal consists in is the lack of consideration of $\mathcal{SN}$ capacities (i.e. unlimited) and the use of static routing tables in the network. In addition, the proposal requires the star $\mathcal{VN}$ topology, which is strongly binding.

In [10], the authors propound two $\mathcal{VN}$ embedding algorithms, named `Deterministic-ViNE` and `Randomized-ViNE`. Here, the substrate graph is augmented with meta-nodes and meta-edges to form a meta-graph. Each meta-node corresponds to one virtual node and meta-edge is a links between a meta-node and the substrate nodes located in its required geographic area. Note that each virtual node is associated with a specified region where it could be hosted. Nevertheless, it is not realistic to expect end-users to specify all virtual nodes' locations. In fact, only the locations of access nodes can be fixed. The main drawback here is the nodes' locations constraints. Indeed, when these are not defined, `D-ViNE` and `R-ViNE` cannot be executed since the meta-graph cannot be built.

In this paper, we propose a new $\mathcal{VN}$ embedding algorithm based on the ant colony metaheuristic, named `VNE-AC`. It is worth noting that we do not restrain the $\mathcal{VN}$ embedding problem by assuming unlimited substrate resources, or specific $\mathcal{VN}$ topologies or restricting geographic locations of the substrate core node. To the best of our knowledge, this paper is the first proposal that takes into consideration the above assumptions.

III. Problem Statement

A. Network Model

We model the $\mathcal{SN}$ as an undirected graph denoted by $\mathcal{G}^s(\mathcal{N}^s, \mathcal{E}^s)$, where $\mathcal{N}^s$ and $\mathcal{E}^s$ are the sets of physical nodes and their connected links respectively. Each physical node, $n_i^s \in \mathcal{N}^s$, is characterized by its i) residual processing power ($\mathcal{C}_{n_i^s}$), ii) residual memory ($\mathcal{M}_{n_i^s}$), iii) type: access or core ($\mathcal{X}_{n_i^s}$), and iv) geographic location ($\mathcal{L}_{n_i^s}$). Note that if $\mathcal{X}_{n_i^s} = 1$, then n_i^s is an access node. Otherwise, $\mathcal{X}_{n_i^s} = 0$. Likewise, each physical link, $e_x^s \in \mathcal{E}^s$, is typified by its available residual bandwidth, denoted by $\mathcal{B}_{e_x^s}$.

Similarly, the $\mathcal{VN}$ request is modeled as an undirected graph, denoted by $\mathcal{G}^v(\mathcal{N}^v, \mathcal{E}^v)$, where $\mathcal{N}^v$ and $\mathcal{E}^v$ are the sets of virtual nodes and their virtual links respectively. Within the $\mathcal{VN}$ request, each virtual node, $n_i^v \in \mathcal{N}^v$, is associated with the required processing power ($\mathcal{C}_{n_i^v}$) and memory ($\mathcal{M}_{n_i^v}$), its type ($\mathcal{X}_{n_i^v}$), and geographic location ($\mathcal{L}_{n_i^v}$) whether is an access node. Moreover, each virtual link, $e_x^v \in \mathcal{E}^v$, requests $\mathcal{B}_{e_x^v}$ in terms of bandwidth.

B. Virtual Network Embedding Problem

In this section, we will model the embedding problem of the $\mathcal{VN}$ ($\mathcal{G}^v$) in the $\mathcal{SN}$ ($\mathcal{G}^s$). First, we assume that all the physical resources (i.e. bandwidth, processing power, and memory) in $\mathcal{G}^s$ are limited. In fact, $\mathcal{G}^s$ is not able to host an infinity number of $\mathcal{VN}$ requests. Consequently, an intelligent and judicious mapping of the $\mathcal{VN}$ in $\mathcal{G}^s$ is necessary in order to maximize the acceptance rate and the substrate provider's profit.

Let $\triangle_{\mathcal{N}}$ denote the binary node-mapping matrix with m rows and n columns. Each row is associated with one virtual node and m is equal to $|\mathcal{N}^v|$. As well, each column is associated with one substrate node and n is equal to $|\mathcal{N}^s|$. Then, $\triangle_{\mathcal{N}}(i,j) = 1$ when the virtual node n_i^v is mapped in the substrate node n_j^s. Otherwise, $\triangle_{\mathcal{N}}(i,j) = 0$.

Node mapping is constrained so that for each $\mathcal{VN}$ request, $\mathcal{G}^v$, two virtual nodes cannot be assigned to the same substrate node. Formally,

$$\forall j \in \{1, 2, \ldots, n\}, \sum_{i}^{m} \triangle_{\mathcal{N}}(i,j) \leq 1 \tag{1}$$

In addition, each virtual node must be assigned to only one physical node. Formally,

$$\forall i \in \{1, 2, \ldots, m\}, \sum_{j}^{n} \triangle_{\mathcal{N}}(i,j) = 1 \tag{2}$$

The virtual node, n_i^v, can be mapped in the substrate node, n_j^s, if the available residual resources (i.e. $\mathcal{C}_{n_j^s}$ and $\mathcal{M}_{n_j^s}$) are at least equal to those required (i.e. $\mathcal{C}_{n_i^v}$, $\mathcal{M}_{n_i^v}$). Formally,

$$\begin{gathered} \triangle_{\mathcal{N}}(i,j)\left(\mathcal{C}_{\mathcal{N}_j^s} - \mathcal{C}_{\mathcal{N}_i^v}\right) \geq 0 \\ \triangle_{\mathcal{N}}(i,j)\left(\mathcal{M}_{\mathcal{N}_j^s} - \mathcal{M}_{\mathcal{N}_i^v}\right) \geq 0 \\ \forall j \in \{1, 2, \ldots, n\}, \forall i \in \{1, 2, \ldots, m\} \end{gathered} \tag{3}$$

Besides, if n_i^v is an access node then it can only be embedded in substrate access nodes located in the geographic area delimited by a circle with a center $\mathcal{L}_{n_i^v}$ and a radius $\mathcal{D}$. Formally,

$$\begin{gathered} \triangle_{\mathcal{N}}(i,j)\mathcal{X}_{n_i^v}||\mathcal{L}_{n_i^v}\mathcal{L}_{n_j^s}|| \leq \mathcal{D} \\ \left(\mathcal{X}_{n_i^v} - \mathcal{X}_{n_j^s}\right) 1_{(\triangle_{\mathcal{N}}(i,j)=1)} = 0 \\ \forall j \in \{1, 2, \ldots, n\}, \forall i \in \{1, 2, \ldots, m\} \end{gathered} \tag{4}$$

where 1_{cond} is equal to 1 if "cond" is true, otherwise is 0.

Each virtual link, e_x^v between n_i^v and n_j^v is assigned to a unsplittable substrate **path** denoted by $\mathcal{P}_{e_x^v}$ between n_i^s and n_j^s. Note that $\mathcal{P}_{e_x^v}$ is a set of substrate links. In fact $\mathcal{SN}$s mainly make use of shortest-path-based routing protocols such as Open Shortest Path First (OSPF). Indeed, to employ sophisticated splittable routing algorithms, already deployed IP routers must be upgraded. This would significantly increase capital expenditures. Let $\triangle_{\mathcal{E}}$ denote the binary edge assignment matrix with p rows and q columns. Each row is related to one virtual link and p is equal to $|\mathcal{E}^v|$. Furthermore, each column is related to one substrate link and q is equal to $|\mathcal{E}^s|$. $\triangle_{\mathcal{E}}(x,y)$ is equal to 1 if the substrate link $e_y^s \in \mathcal{P}_{e_x^v}$. Moreover, in the matrix $\triangle_{\mathcal{E}}$, all the allocated substrate links e_y^s in each row e_x^v must form a path between n_i^s and n_j^s. Otherwise, the configuration is not allowed. Moreover, the available residual bandwidth of all the physical links in $\mathcal{P}_{e_x^v}$ must be at least equal to $\mathcal{B}_{e_x^v}$. Formally,

$$\begin{gathered}\triangle_{\mathcal{E}}(x,y)\left(\mathcal{B}_{e_y^s}-\mathcal{B}_{e_x^v}\right)\geq 0\\ \forall x\in\{1,2,\ldots,p\},\forall y\in\{1,2,\ldots,q\}\end{gathered}\tag{5}$$

Our objective is to generate, for each $\mathcal{VN}$ request, the best possible mapping (i.e. $\triangle_{\mathcal{N}}$ and $\triangle_{\mathcal{E}}$) while also minimizing the embedding cost in terms of allocated resources in the $\mathcal{SN}$. Note that for a specific $\mathcal{VN}$ request, whatever the mapping the resources allocated by the virtual nodes are identical. However, the assigned resources for the virtual links depend on the substrate path length. For this reason, we define the cost of the mapping request by the total bandwidth used in all the substrate paths. In this respect, our objective is to minimize the mapping request cost as expressed below.

$$\sum_{x=1}^{p}\sum_{y=1}^{q}\triangle_{\mathcal{E}}(x,y)\mathcal{B}_{e_x^v}\tag{6}$$

We will now outline our $\mathcal{VN}$ embedding optimization problem.

Problem 1 $\mathcal{VN}$ embedding optimization problem

`minimize`	$\sum_{x=1}^{p}\sum_{y=1}^{q}\triangle_{\mathcal{E}}(x,y)\mathcal{B}_{e_x^v}$
`subject to:`	$\sum_{i}^{m}\triangle_{\mathcal{N}}(i,j)\leq 1$
	$\sum_{j}^{n}\triangle_{\mathcal{N}}(i,j)=1$
	$\triangle_{\mathcal{N}}(i,j)\left(\mathcal{C}_{\mathcal{N}_j^s}-\mathcal{C}_{\mathcal{N}_i^v}\right)\geq 0$
	$\triangle_{\mathcal{N}}(i,j)\left(\mathcal{M}_{\mathcal{N}_j^s}-\mathcal{M}_{\mathcal{N}_i^v}\right)\geq 0$
	$\triangle_{\mathcal{N}}(i,j)\mathcal{X}_{n_i^v}\|\mathcal{L}_{n_i^v}\mathcal{L}_{n_j^s}\|\leq\mathcal{D}$
	$\left(\mathcal{X}_{n_i^v}-\mathcal{X}_{n_j^s}\right)1_{(\triangle_{\mathcal{N}}(i,j)=1)}=0$
	$\{e_y^s\}$ `builds a path for` e_x^v
	$\triangle_{\mathcal{E}}(x,y)\left(\mathcal{B}_{e_y^s}-\mathcal{B}_{e_x^v}\right)\geq 0$
	$\triangle_{\mathcal{N}}(i,j)\in\{0,1\}$
	$\triangle_{\mathcal{E}}(x,y)\in\{0,1\}$
	$\forall j\in\{1,2,\ldots,n\},\forall i\in\{1,2,\ldots,m\}$
	$\forall x\in\{1,2,\ldots,p\},\forall y\in\{1,2,\ldots,q\}$

Our problem is a binary combinatorial optimization problem. As stated earlier, it has been proved to be NP-hard [3]–[5]. In the next section, we will propose a new scalable $\mathcal{VN}$ embedding algorithm based on the ant colony metaheuristic, denoted by `VNE-AC`.

IV. Proposal: `VNE-AC` algorithm

As stated above, the $\mathcal{VN}$ embedding problem is resolved mainly by using greedy algorithms or standard solvers (e.g., Cplex, Glpk, etc). Unfortunately, a greedy approach usually converges to a local optimum and solvers cannot tackle large-size problems. In this respect, we propose to make use of metaheuristics. These can be applied to several optimization problems, but there are no details explaining how they can be applied. Indeed, the main challenge is to adapt the metaheuristic to address the problem in hand. In existing literature, several metaheuritsics can be found such as Ant Colony, Tabu Search, Genetic Algorithm, etc.

In this paper, we propose a new $\mathcal{VN}$ embedding algorithm, denoted by `VNE-AC`. It is based on a `Max-Min Ant System` [6] metaheuristic which operates as follows. First, the problem is divided into a set of solution components that are sorted and then solved sequentially. Note that solving a solution component is only equivalent to building a small part of the overall solution. Next, A_{max} parallel artificial ants are launched and iteratively explore the search space until a predetermined number of iterations N_{max} is reached. During each iteration, each ant incrementally constructs the solution by transiting (i.e. moving) from one solution component to another. To do so, the ant localizes potential candidates in the $\mathcal{SN}$ for each solution component in the search space then selects just one by applying a stochastic local decision. It is worth noting that the decision is based on heuristic information and artificial pheromone trails, which respectively quantify the desirability of a *priori* and *posteriori* transition. Once each ant builds its full solution, the best one (i.e. the one that most enhances the objective function) among all solutions generated by all ants is selected. Furthermore, the artificial pheromone is slightly evaporated in all the environment. This helps the ants to discover new trajectories in the environment and not to be trapped in local optimums. Nevertheless, the artificial pheromone trail of each solution component is reinforced at the visited points in the environment according to the best trajectory traveled by ants to build the whole solution. This helps the ants to improve and continually refine the best solution obtained. The process is repeated during N_{max} iterations and the global best solution generated by the A_{max} ants is considered to be the output solution. The fundamental stages of our proposal are: i) *Formation of solution components*, ii) *Localization of potential candidates*, iii) *Stochastic selection of the candidate*, iv) *Selection of the best solution* and v) *Updating the pheromone trail*. These will now be described in greater detail.

A. Formation of solution components

In this stage, the $\mathcal{VN}$ request, $\mathcal{G}^v$, is split up into many sub-requests, known as solution components, denoted by $\mathcal{SC}_a$. To do this, we first map all access nodes in $\mathcal{G}^v$ since their geographic locations are fixed. Indeed, the virtual access node n_i^v (i.e. $\mathcal{X}_{n_i^v}=1$) is embedded in the substrate access node n_j^s (i.e. $\mathcal{X}_{n_j^s}=1$) offering the most residual resources (giving priority to processing power) within the circular geographic area defined by the center $\mathcal{L}_{n_i^v}$ and the radius $\mathcal{D}$. Recall that the residual resources of n_j^s (i.e. $\mathcal{C}_{n_j^s}$ and $\mathcal{M}_{n_j^s}$) must be at least equal to those requested (i.e. $\mathcal{C}_{n_i^v}$ and $\mathcal{M}_{n_i^v}$). Then, existing virtual links between the virtual access nodes will be mapped as explained in the next section, IV-C.

The remaining virtual nodes and links are denoted by $\tilde{\mathcal{G}}^v$. Note that in $\tilde{\mathcal{G}}^v$, we found many virtual links missing one of their outermost virtual nodes, which had already been mapped. Henceforth, we will refer to this type of link as "hanging link".

Thus, after mapping virtual access nodes and the existing virtual links between them, we can now sequentially generate the sequence of solution components $\{\mathcal{SC}_a\}, a=1,2,\cdots,\xi$. To do so, we select the virtual node with the highest number of hanging links in $\tilde{\mathcal{G}}^v$. Then, $\mathcal{SC}_a$ is constituted using the **selected virtual node** and all its attached **hanging links** only. Afterward, we subtract $\mathcal{SC}_a$ from $\tilde{\mathcal{G}}^v$ (i.e. $\tilde{\mathcal{G}}^v\leftarrow\tilde{\mathcal{G}}^v\backslash\mathcal{SC}_a$) and we then repeat the same process recursively, to form the rest of solution components, until $\tilde{\mathcal{G}}^v$ becomes empty (i.e. $\tilde{\mathcal{G}}^v=\emptyset$).

As a result, the $\{\mathcal{SC}_a\}$ series, $a=1,2,\cdots,\xi$, is generated. Thus the artificial ants can start mapping $\{\mathcal{SC}_a\}$ sequentially in the same order during each iteration. In this respect, to embed the solution component $\mathcal{SC}_a$, each ant first of all localizes

potential candidate substrate nodes that could host $\mathcal{SC}_a$, as detailed hereafter.

B. Localization of potential candidates

The potential substrate nodes $\{\mathcal{PN}_a^b\}$ for hosting $\mathcal{SC}_a$ are those supplied with sufficient residual resources (i.e. processing power and memory) and localized within $\mathcal{H}$ hops from the nearest substrate core node to a point $\mathcal{P}$. We define $\mathcal{P}$ as the barycentre of the set of substrate nodes, denoted by $\mathcal{N}_a$, hosting the virtual nodes of hanging links in $\mathcal{SC}_a$. By doing so, we aim to embed the virtual node of $\mathcal{SC}_a$ in the substrate node located at the same distance to its directly connected nodes, $\mathcal{N}_a$, that are already mapped. Consequently, the path lengths will be balanced and consumed resources will be minimized.

Once the potential set of substrate candidates is determined $\mathcal{PN}_a = \{\mathcal{PN}_a^b\}$, one is selected according to a stochastic decision outlined in the following section.

C. Stochastic selection of the candidate

To select the best potential substrate node $\mathcal{PN}_a^b$, we define the probability of desirability, denoted by p_{ab}, to embed $\mathcal{SC}_a$ in $\mathcal{PN}_a^b$. This probability, p_{ab}, depends on the heuristic information, η_{ab}, and the artificial pheromone trail, τ_{ab}, of $\mathcal{SC}_a$ in $\mathcal{PN}_a^b$, as recommended in `Max-Min Ant System` [6]. Formally,

$$p_{ab} = \frac{\tau_{ab}^{\alpha} \cdot \eta_{ab}^{\beta}}{\sum_{c=1}^{|\mathcal{PN}_a|} \left[\tau_{ac}^{\alpha} \cdot \eta_{ac}^{\beta}\right]} \tag{7}$$

where α and β respectively determine the relative importance of the pheromone trail and the heuristic information. Remember that $\mathcal{PN}_a$ is the set of potential candidates for hosting $\mathcal{SC}_a$ determined in the previous stage.

With this in mind, for each substrate node n_b^s we can define the heuristic information η_{ab} as:

$$\eta_{ab} = \mathcal{C}_{n_b^s} + \mathcal{M}_{n_b^s} + \sum_{n_k^s \in \mathcal{N}_a} \mathcal{B}_{\mathcal{P}_{bk}} \tag{8}$$

where $\mathcal{P}_{bk}$ is the best path between n_b^s and n_k^s in terms of our new metric (see equation (9)).

Thanks to η_{ab}, substrate nodes with high available resources are associated with a high value of p_{ab} and thus, are more likely to be selected as $\mathcal{PN}_a^b$. Furthermore, the artificial pheromone trail τ_{ab} is defined in each iteration as expressed in equations (10) and (11).

In doing so, we assign $\mathcal{SC}_a$ to one candidate in $\mathcal{PN}_a$ according to the probability values $\{p_{ab}\}$ calculated above. Next, we generate the substrate paths associated with the virtual hanging links in $\mathcal{SC}_a$ since both extremity nodes are now assigned. Our main objective is to reduce the allocated bandwidth in the $\mathcal{SN}$. So, the Dijkstra shortest path in terms of hops would be the best approach. Unfortunately, we noticed many $\mathcal{VN}$ request rejects due to the hot spots in the $\mathcal{SN}$ and the required bandwidth is not taken into consideration. For this reason, we propose to generate the shortest path according to our own new metric, which takes into account the residual bandwidth and the path hop-length. In fact, we aim to choose the shortest paths offering the best bandwidths in order to reduce resources allocated to substrate links. The distance of a single path, $\mathcal{P}$, is defined as:

$$d(\mathcal{P}) = \frac{length(\mathcal{P})}{\min_{e_x^s \in \mathcal{P}} \mathcal{B}_{e_x^s}} \tag{9}$$

Finally, once all the ants map $\{\mathcal{SC}_a\}, a = 1, 2, \cdots, \xi$, and generate the full $\mathcal{VN}$ mapping solution $\mathcal{SL}_1, \mathcal{SL}_2, \cdots, \mathcal{SL}_{A_{max}}$, we can select the best one, $\mathcal{SL}_{best}$, as detailed in the next section. It is worth noting that each ant converges to a solution that is different to the ones generated by the other ants due to the probability-based nature of the embedding decision.

D. Selection of the best solution

The selection of one solution among those generated by all ants is based on the objective function value defined by equation (6). Thus, we select the solution $\mathcal{SL}_{best}$ that minimizes $\mathcal{VN}$ request's cost which is defined as the total allocated bandwidth in the $\mathcal{SN}$.

E. Updating the pheromone trail

Before reiterating the process, moving from iteration t to $t+1$, the pheromone trail is evaporated for all $\{\mathcal{SC}_a\}, a = 1, 2, \cdots, \xi$, in the whole $\mathcal{SN}$. To achieve this, the pheromone trail is weighted by ρ with $0 < \rho < 1$. Formally,

$$\tau_{ab}(t+1) = \rho \cdot \tau_{ab}(t) \tag{10}$$

Conversely, we reinforce the pheromone trail of $\{\mathcal{SC}_a\}$, $a = 1, 2, \cdots, \xi$, in the substrate candidate nodes $\{\mathcal{PN}_1^{best}, \mathcal{PN}_2^{best}, \ldots, \mathcal{PN}_\xi^{best}\}$ participating in the construction of the best solution $\mathcal{SL}_{best}$. To do so, we increase the pheromone trail by the inverse of $\mathcal{SL}_{best}$'s mapping cost in the $\mathcal{SN}$ weighted by ϕ as suggested in [6].

$$\tau_{ab}(t+1) = \tau_{ab}(t+1) + \frac{\phi}{\sum_{x=1}^{p}\sum_{y=1}^{q} \triangle_{\mathcal{E}}(x,y)\mathcal{B}_{e_x^v}} \tag{11}$$

where p and q are equal to $|\mathcal{E}^v|$ and $|\mathcal{E}^s|$ respectively, as mentioned in equation (5) and ϕ a constant parameter.

It is worth noting that the lowest and the largest bounds of τ_{ab} are fixed in relation to the `Max-Min Ant System` as described in [6]. Moreover, at $t = 0$, $\tau_{ab}(0)$ is set to some arbitrarily high value in order to increase the exploration of solutions during the first iterations.

V. Performance Evaluation

In this section, we will study the efficiency of our proposal, `VNE-AC`. To achieve this, first we describe our discrete event $\mathcal{VN}$ embedding simulator. Then, we will define several metrics in the aim of showing the advantages of `VNE-AC` compared with the prominent existing schemes mentioned above: i) `VNE-Least` [7], ii) `VNE-Cluster` [7], iii) `VNE-Subdividing` [7], and iv) `VNE-Greedy` [8]. Finally, building on the outputs of the above simulations, we will assess our proposal and comment the results obtained.

A. Simulation environment

We implement a discrete event $\mathcal{VN}$ embedding simulator. In this respect, we use the `GT-ITM` tool to generate random $\mathcal{SN}$ and $\mathcal{VN}$ topologies. Note that we model the arrival of $\mathcal{VN}$ requests by a Poisson Process with rate λ_A. We also model $\mathcal{VN}$ lifetime by exponential distribution with mean μ_L.

As stated in [8], we set the $\mathcal{SN}$ size to 100. In this case, the ratio of access and core nodes is fixed to 20% and 80%, respectively. Furthermore, we set the $\mathcal{VN}$ size according to a discrete uniform distribution, using the values given in $[2, 10]$. Since virtual access nodes are defined by customers, we assume that each virtual node could be access or core with a probability of 0.5. It is worth noting that in both cases ($\mathcal{VN}$ and $\mathcal{SN}$), each pair of nodes is randomly connected with a probability of 0.5. The arrival rate λ_A and the average lifetime μ_L of

$\mathcal{VN}$s are fixed to 4 requests per 100 time unit and 1000 time units respectively. We calibrate the capacity of substrate nodes and links (i.e. $\mathcal{C}_{n_i^s}$, $\mathcal{M}_{n_i^s}$, and $\mathcal{B}_{e_x^s}$) according to a continuous uniform distribution taking values in $[50, 100]$. As well, we set the required virtual resources (i.e. $\mathcal{C}_{n_i^v}$, $\mathcal{M}_{n_i^v}$, and $\mathcal{B}_{e_x^v}$) according to a continuous uniform distribution, using the values given in $[10, 20]$.

In our simulator, we implement our `VNE-AC` proposal alongside the related strategies: i) `VNE-Least`, ii) `VNE-Cluster`, and iii) `VNE-Greedy`. Based on extensive simulations, we calibrate our heuristic `VNE-AC`'s parameters. We set the number of ants A_{max} and the number of iterations N_{max} to 10 and to 20, respectively. We fix the hop-range within local search area $\mathcal{H}$, the importance of heuristic information and pheromone trail α and β to 2, 1 and 2 respectively. We set the number of $\mathcal{VN}$ requests to 2000. All simulation results of pseudo-random strategies are calculated with a confidence level equal to 99.70%. Note that tiny confidence intervals are not shown in the following figures.

B. Performance metrics

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

1) $\mathbb{Q}(t)$*:* is the reject rate of $\mathcal{VN}$ requests during the time period $[0, t]$.

2) $\mathbb{C}_{tot}(t)$*:* is the cumulative provisional cost of all accepted requests in the $\mathcal{SN}$ during the time period $[0, t]$. Formally,

$$\mathbb{C}_{tot}(t) = \sum_{\mathcal{G}^v \in \mathcal{AR}_t} \mathbb{C}(\mathcal{G}^v) \tag{12}$$

where $\mathbb{C}(\mathcal{G}^v)$ is defined as in equation (6) and $\mathcal{AR}_t$ is the set of accepted requests during $[0, t]$.

3) $\overline{\mathbb{C}(t)}$*:* is the average provisional cost of requests in the $\mathcal{SN}$ during the time period $[0, t]$. Formally,

$$\overline{\mathbb{C}(t)} = \frac{\sum_{\mathcal{G}^v \in \mathcal{AR}_t} \mathbb{C}(\mathcal{G}^v)}{|\mathcal{AR}_t|} \tag{13}$$

4) $\mathbb{R}_{tot}(t)$*:* evaluates the $\mathcal{SN}$ provider's profitability by calculating the total benefit of all accepted requests during the time period $[0, t]$ (i.e. cumulative total revenue). Formally it can be expressed as:

$$\mathbb{R}_{tot}(t) = \sum_{\mathcal{G}^v \in \mathcal{AR}_t} \mathbb{R}(\mathcal{G}^v) \tag{14}$$

where $\mathbb{R}(\mathcal{G}^v)$ is the request revenue defined as in [7] [8] [10]:

$$\mathbb{R}(\mathcal{G}^v) = \sum_{n_i^v \in \mathcal{N}^v} \mathcal{C}_{n_i^v} + \sum_{n_i^v \in \mathcal{N}^v} \mathcal{M}_{n_i^v} + \sum_{\mathcal{B}_{e_x^v} \in \mathcal{E}^v} \mathcal{B}_{e_x^v} \tag{15}$$

5) $\overline{\mathbb{R}(t)}$*:* measures the average revenue of $\mathcal{VN}$ requests. It is worth noting that when $\overline{\mathbb{R}(t)}$ is high, the embedding strategy maps requests generating a larger revenue. It can be expressed as:

$$\overline{\mathbb{R}(t)} = \frac{\sum_{\mathcal{G}^v \in \mathcal{AR}_t} \mathbb{R}(\mathcal{G}^v)}{|\mathcal{AR}_t|} \tag{16}$$

6) $\mathbb{U}(t)$*:* measures average usage rate of substrate links at time t. It can be expressed as:

$$\mathbb{U}(t) = \frac{1}{|\mathcal{E}^s|} \times \sum_{e_s^x \in \mathcal{E}^s} \left(\frac{\mathcal{B}_{e_s^x}}{\mathcal{B}_{e_s^x}^{max}} \right) \tag{17}$$

In the following section, we will present the results of our simulations and summarize the key observations.

C. Evaluation Results

In Table I, we compare the reject rate and total revenue of the whole (i.e. 2000) $\mathcal{VN}$ requests. It shows that our proposal, `VNE-AC`, significantly outperforms the related strategies. We can see that `VNE-AC` denies only $4.56 \pm 0.40\%$ of requests. It is worth noting that `VNE-Cluster`, `VNE-Least`, and `VNE-Subdividing` decline the most of requests (69%, 73.90%, and 67.45 respectively). This is mainly because the latter strategies do not consider the residual resources in the $\mathcal{SN}$. In addition, our proposal improves `VNE-Greedy` by roughly three times ($4.56\% \pm 0.40 \rightarrow 12.95\%$). As well, we can see that `VNE-AC` maximizes the substrate provider's revenue. Note that request's revenue is proportional to the amount of allocated resources. This can be explained due to the fact that the larger number of accepted requests leads to the increase of allocated resources.

TABLE I
PERFORMANCE OF $\mathcal{VN}$ REQUESTS

Strategy	reject rate (%)	provider's revenue ($\times 10^4$)
`VNE-AC`	4.56 ± 0.40	47.70 ± 0.26
`VNE-Greedy`	12.95	42.51
`VNE-Cluster`	69	13.47
`VNE-Least`	73.90	10.05
`VNE-Subdividing`	67.45	10.85

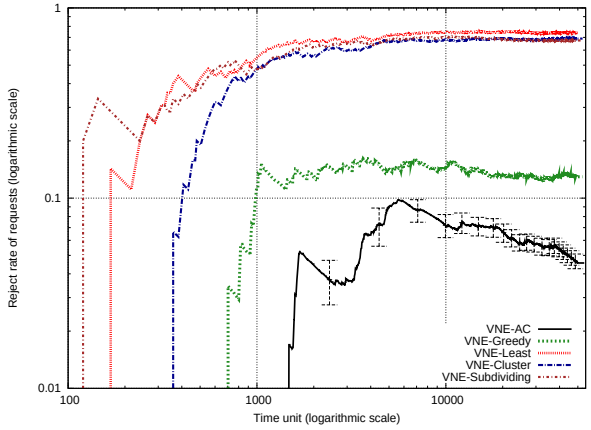


Fig. 1. Reject rate - $\mathbb{Q}$

Fig. 1 illustrates the reject rate of requests $\mathbb{Q}(t)$ during the simulation. It shows that `VNE-AC` notably minimizes the reject rate compared with the related strategies, throughout the simulations. The rationale behind this can be explained by the fact that our scheme aims to maximize the residual resources in the $\mathcal{SN}$. Thus, `VNE-AC` avoids more critical resources and consequently prevents congestion in the $\mathcal{SN}$. As illustrated in Fig. 1, it is worth noting that the requests's reject process starts late with `VNE-AC` (at 1420 time unit) compared to the related strategies. Moreover, the `VNE-AC` reject rate's peak is the lowest and then decreases during the simulation which proves that the resources are allocated efficiently. In fact, unlike other algorithms, our solution is based on pseudo-randomized node selection. This approach, as proved in algorithm design randomization, leads to effective performance in computationally intractable problems.

Fig. 3 shows the cumulative and average requests' provisional cost obtained. As depicted in Fig. 3.(a), we can see that `VNE-AC` enhances the total provisional cost (i.e. lowest) compared with `VNE-Greedy` and `VNE-Cluster`. In this case, our proposal allocates fewer resources (i.e. bandwidth) than `VNE-Greedy` and `VNE-Cluster` while also accepting more requests, as illustrated in Fig. 1. As a result, our proposal,

VNE-AC, efficiently embeds the requests. As shown in Fig. 2, thanks to our distance path metric defined in equation (9), we succeed in making a trade-off between minimizing the length of substrate paths and maximizing residual bandwidth in the $\mathcal{SN}$. We can see that VNE-AC allocates approximately the same bandwidth in the $\mathcal{SN}$ but our proposal assigns more requests. It is worth noting that VNE-Least and VNE-Subdividing consume less resources than our proposal. This can be explained by the fact that VNE-Least is based on shortest path algorithm and VNE-Subdividing favors the mapping of small $\mathcal{VN}$ requests in terms of required resources (as illustrated in Fig. 4(b)) which rationally decreases the provision cost. However, both strategies have the highest reject rate since they do not take into account the bottleneck effect in the $\mathcal{SN}$.

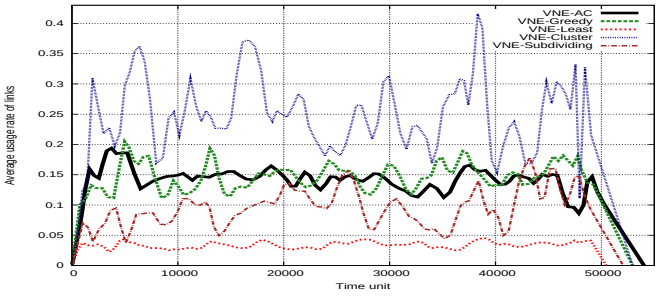
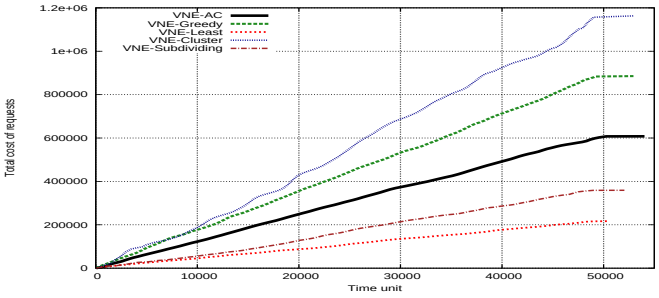
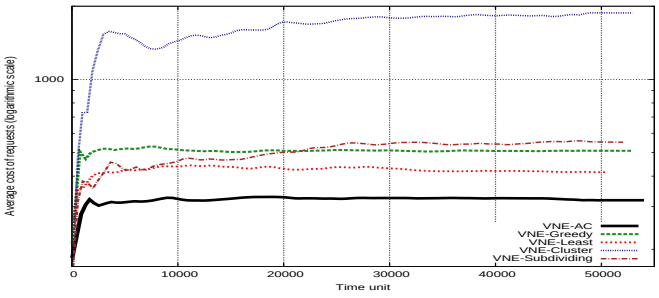


Fig. 2. Link's average usage rate - $\mathbb{U}$



(a) Cumulative - $\mathbb{C}_{tot}$

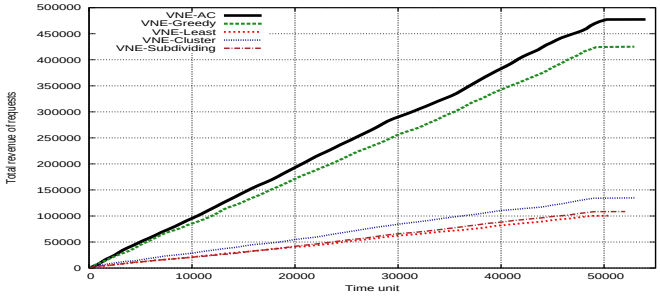


(b) Average - $\overline{\mathbb{C}}$

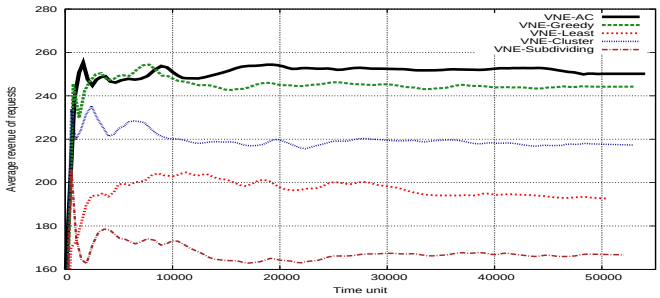
Fig. 3. Comparison of provision cost

It is clear to see in Fig. 3.(b) that VNE-AC produces the best average provisional cost thanks to our path metric and the large number of accepted requests. We can also see that VNE-AC's average provisional cost is constant throughout the simulation, even though more $\mathcal{VN}$ requests are embedded in the $\mathcal{SN}$.

Fig. 4(a) shows that our proposal, VNE-AC, performs better than the other approaches tested, since it offers higher revenue to the $\mathcal{SN}$ provider. We can see that returns generated by VNE-AC rise linearly together with incoming requests. This can be explained by the fact that VNE-AC accepts many more requests than related schemes. Additionally, coordinated nodes and links mapping stages leads to efficient mapping and thus generates a higher income. Moreover, VNM-AC not only improves the total revenue but also embeds requests with larger revenue potentials. This is clearly shown in the Fig. 4(b) which plots the average revenue of requests. We can see that the average revenue of VNE-AC is roughly constant, in spite of incoming requests.



(a) Cumulative - $\mathbb{R}_{tot}$



(b) Average - $\overline{\mathbb{R}}$

Fig. 4. Comparison of revenue

VI. Conclusion

This paper studied the virtual network embedding problem. Note that it is NP-hard and computationally intractable. In response, we proposed a new efficient and scalable strategy named VNE-AC. It is based on a Max-Min Ant System metaheuristic. Therefore, our aim is to minimize the allocated resources in the substrate network for each virtual network request in order to minimize the reject rate and maximize the provider's revenue. Based on extensive simulations, we have shown that our proposal makes embedding more effective and achieves better resource utilization. Moreover, VNE-AC outperforms related strategies proposed in existing literature: VNE-Greedy, VNE-Cluster, VNE-Subdividing and VNE-Least.

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] ——, "A survey of network virtualization," *Computer Networks*, vol. 54, 2010.

[3] D. G. Andersen, "Theoretical approaches to node assignment," 2002, http://www.cs.cmu.edu/?dga/papers/andersen-assign.

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

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

[6] T. Stützle and H. H. Hoos, "MAX-MIN ant system," *Future Gener. Comput. Syst.*, vol. 16, pp. 889–914, 2000.

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

[8] 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.

[9] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," Washington University in St. Louis, Tech. Rep., 2006. [Online]. Available: http://www.arl.wustl.edu/~jl1/research/tech_report_2006.pdf

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