Efficient Virtual Network Embedding With Backtrack Avoidance for Dynamic Wireless Networks

Sherif Abdelwahab, ~~Senior~~ *Member, IEEE*, Bechir Hamdaoui, *Senior Member, IEEE*, Mohsen Guizani, *Fellow, IEEE*, and Taieb Znati

***Abstract*—We develop an efficient virtual network embedding (VNE) algorithm, termed BIRD-VNE, for mobile wireless networks. BIRD-VNE is an approximation algorithm that ensures a close to optimal virtual embedding profit and acceptance rate while minimizing the number of virtual network migrations resulting from the mobility of wireless nodes. BIRD-VNE employs a constraint satisfaction framework by which we analyze the constraint propagation properties of the VNE problem and design constraint processing algorithms that efficiently narrow the solution space and avoid backtracking as much as possible without compromising the solution quality. Our evaluation results show that the likelihood that BIRD-VNE results in backtracking is small, thus demonstrating its effectiveness in reducing the search space. We analytically and empirically verify that BIRD-VNE outperforms existing VNE algorithms with respect to computational efficiency, closeness to optimality, and its ability to avoid potential migrations in mobile wireless networks.**

***Index Terms*—Mobile wireless networks, virtual network embedding, remote sensor networks.**

I. INTRODUCTION

VIRTUAL network embedding in wireless networks can have a pivotal role in several areas including: sensor network virtualization [1], vehicular cloud [2], mobile edge computing [3], [4], [5], network based and geographically distributed cloud environment [6], [7], and cyber foraging [8]. By means of virtualization, it is possible to embed, with low cost, large-scale virtual sensor networks onto sensor-equipped physical devices (e.g. smart-phones, autonomous vehicles) so as to perform specific sensing tasks and autonomous, agile, and timely decisions in a distributed manner. Such virtual networks can support several applications such as: urban sensing, intelligent transportation, terrain exploration, disaster recovery, and surveillance. In addition, VNE can be used to enable virtual content delivery in wireless networks near the network edge. VNE algorithms can then deploy surrogates of services (e.g. networked virtual servers) in proximity to users to improve their perceived latency, where geographical locations and mobility patterns of users are crucial parameters to maintain a target content delivery quality. In a more general context, virtual network embedding in wireless networks can enable effective distributed processing of real-time content and allow agile decision making from data at its "actual sources".

The focus of this paper is on the design of virtual network embedding (VNE) techniques that enable on-demand mapping of virtual networks onto substrate mobile wireless networks. More specifically, the VNE problem consists of mapping the virtual nodes to substrate nodes and the virtual links to substrate paths in such a way that all resource (CPU, storage, and bandwidth) requirements of the virtual network are met. Here, a virtual network consists of a set of virtual nodes, each requiring CPU processing capability and storage capacity to process data in a predefined geographical area, and a set of virtual links connecting these virtual nodes, each requiring some bandwidth capacity. The substrate network, on the other hand, consists of a large set of mobile wireless nodes, each having sensing and Internet-access capabilities.

Unlike wired networks, mobile wireless networks' dynamics (e.g. node mobility, link instability) create new challenges that require new architectural and algorithmic considerations when it comes to enabling VNE. Mobility of substrate nodes, in particular, may invalidate the operations of virtual networks as nodes move away from desired locations of some virtual nodes. Such a mobility can also change the connectivity of the substrate nodes–and so can the substrate paths–that are already used by virtual links, making them insufficient or invalid. In such cases, VNE solutions shall remap (migrate) invalid virtual networks to other substrate nodes and paths [9]. As migrations incur a significant overhead [10], we shall design architectural and algorithmic solutions that can effectively capture node mobility and topology changes, and minimize virtual network migrations due to nodes mobility while not compromising the effectiveness of VNE techniques.

The effectiveness of the VNE techniques can essentially be captured through three metrics: computation time (the time it takes to solve a VNE instance), embedding cost (the amount of overhead incurred and resources needed to solve a VNE instance), and acceptance rate (the ratio of successfully solved VNE instances to the total number of instances). Therefore, in addition to meeting the resource requirements, the aim of VNE techniques is to reduce the computation time, minimize the embedding cost, and increase the acceptance rate. The challenge, however, is that these three performance goals are often

Manuscript received February 18, 2015; revised June 29, 2015 and September 30, 2015; accepted November 23, 2015. This work was supported by the National Science Foundation (NSF) under Grant CNS-1162296. The associate editor coordinating the review of this paper and approving it for publication was W. Wang.

S. Abdelwahab and B. Hamdaoui are with Oregon State University, Corvallis, OR 97331 USA (e-mail: abdelwas@eecs.orst.edu; hamdaoui@eecs.orst.edu).

M. Guizani is with ~~Qatar University, Doha, Qatar~~ (e-mail: mguizani@ieee.org).

T. Znati is with the University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: znati@cs.pitt.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2015.2507134

conflicting with one another. For instance, backtracking algorithms can, in general, find optimal solutions, but they do so in exponential time [11]. Other heuristic approaches, on the other hand, can find solutions in polynomial time, but these solutions are sub-optimal, thus leading to low acceptance rates [12].

In this paper, we develop VNE techniques that strike a good balance between these three performance goals by finding near optimal solutions in polynomial times (short execution times) while yielding high embedding profits (minimal embedding costs) and high acceptance rate. The proposed approach takes also into account potential virtual networks migrations due to substrate nodes mobility in its objective definition to minimize the anticipated overhead associated with migrating invalid virtual networks. Our proposed approach consists of designing algorithms that are based on backtracking techniques so as to ensure good solution optimality, while reducing the computational complexity and the embedding cost by exploiting the constraint propagation properties of the VNE problem. Essentially, they reduce the embedding complexity and cost by narrowing down the search space and avoiding backtracking as much as possible without compromising the solution quality so as to maintain high acceptance rates and minimize potential virtual network migrations. To recap, our contributions in this paper are twofold.

- Developing pruning techniques that reduce the embedding time and cost significantly by reducing the search space. These techniques eliminate the need for backtracking during the embedding solution search, thereby enhancing the embedding time without compromising the optimality of the obtained VNE solutions.
- Developing techniques that account for the VNE embedding cost, expressed in terms of the amount of resources needed and the migration overhead incurred to successfully embed a virtual network, to devise VNE algorithms with minimal embedding costs and minimal potential virtual network migrations.

The rest of the paper is organized as follows. The next section surveys the existing techniques that are related to our proposed VNE approach. In Section III, we state and formulate the VNE problem. We begin by modeling the virtual and substrate networks and the substrate node mobility, and by defining the node and link mapping steps to be performed during the VNE process. We then describe the overall design goals of the VNE technique. In Section IV, we present our pruning techniques proposed to reduce the embedding search space. We then, in Section V, use these pruning techniques to develop a polynomial-time VNE algorithm, which leverages the benefits of our proposed pruning techniques to avoid backtracking while still maintaining the optimality of the obtained VNE solutions. In the same section, we also derive analytic bounds on the approximation ratio of the incurred objective value of the proposed algorithm. Finally, we present our experimental results and findings in Section VI, and conclude the paper in Section VIII.

II. Related Work

Virtual Network Embedding Algorithms: There have recently been research efforts aiming to develop VNE algorithms, and the recent survey by Fischer et al. [12] presents a detailed classification of such algorithms. Broadly speaking, these algorithms can be classified into three categories: backtracking based algorithms (e.g. branch and bound), stochastic algorithms, and heuristics.

Backtracking based algorithms generally consist of formulating and solving the VNE problem using branch and bound or exact backtracking based techniques [13], [14], [15], [16], [17]. For example, Lischka et. al. [13] show that the VNE problem can be formulated as a graph isomorphism (which is known to be *NP-hard*) and then using a backtracking based algorithm to solve it. Backtracking can, in general, find optimal slutions. However, they do so in exponential time [18].

Stochastic algorithms like simulated annealing, particle swarm optimization, tabu search, or genetic algorithms, are other common approaches that can be used to search for VNE solutions. For example, [19] uses particle swarm optimization to find near optimal solutions in relatively short execution times (as shown empirically). The major drawback of stochastic algorithms, besides their relatively long execution times, is their high likelihood of getting stuck in local minima.

Heuristic algorithms attracts the most attention of researchers given their less complexity when compared to exact backtracking algorithms. Heuristics on the other hand can only find inexact solutions and hardly provide tight approximation gaps [20], [21], [22], [23], [24], [25]. For example, Zhu and Ammar in [20] adopt one very basic greedy algorithm that greedily search for feasible nodes to serve a virtual network and then compute the shortest paths between these nodes. If the evaluated shortest paths can satisfy the demands of the virtual links, the virtual network is considered successfully embedded. This is the most simple but sub-optimal algorithm which brings no guarantee to solve the VNE problem. We refer to this algorithm throughout as baseline. The authors in [21] formulate the VNE problem as two stage, coordinated node and link mapping problems, that are both formulated as Mixed ILP (MIP), and then use a rounding relaxation to find near optimal solutions by an off-the-shelf solver. This algorithm can, however, be very slow especially when the size of the virtual network (number of nodes and links) is large, and is shown to have a worst case complexity of $O(n^{14}\, b^2\, \ln b\, \ln\ln b)$ where n is the number of substrate nodes, and b is the number of input bits to the linear program [21], [22]. Several other works adopted a similar approach to [21], formulating the VNE problem as MIP [26], [27]. Formulating the VNE problem as MIP allows a mechanical problem formulation that can address a wide range of objectives such as energy-awareness and fault-tolerance [28], [29], [6], [7]. Heuristic algorithms, though have better execution times than backtracking algorithms, do result in low acceptance rates, due to their sub-optimal embedding nature.

Our algorithm, Bird-VNE, follows a constraint processing design methodology and involves a simplified form of backtracking to bound the resulting approximation-ratio. Our algorithm is different from other backtracking based solutions in that it relies on the analysis of the constraint properties of the VNE problem. This analysis allows us to develop constraint processing algorithms specific to the VNE problem that effectively prune the search space. Unlike other heuristics, Bird-VNE allocates substrate paths directly to the requested

virtual links, rather than separating node and link mapping or at most coordinating their allocations. This approach leads to a proved approximation-ratio that tightens the Bird-VNE performance which was first proposed in our work in [30].

Virtual Network Embedding and Migration in Wireless Networks: Designing VNE algorithms that account for network dynamics (e.g. wireless link quality instability, links failure, node mobility, etc.) attracted little attention [31], [32], [33]. The authors in [33] discuss virtualization measures that can ensure network embedding feasibility in wireless networks under dynamic behaviors. Also in [32], the authors propose to use VNE over *static* wireless multihop networks. Unlike these papers, we design our VNE embedding considering wireless network dynamics due to substrate nodes that can invalidate already embedded virtual networks, hence mandating migrating these virtual networks to ensure service continuity.

Virtual network migration has also attracted the attention of some researchers to fix invalid virtual networks [9], [34], [35]. The work by Houidi et. al [9] is one example in which the authors propose to continuously monitor already embedded virtual networks and to detect possible events that may trigger migration, hence adaptively reembed these virtual networks. Unfortunately virtual network migration is accompanied with several challenges and overheads. A recent study demonstrates the potential migration challenges including: unavoidable packet loss, slow adaptability of switches to changes, and critical deadline time to switch packets to new paths. [10].

In this paper, we extend our work in [30] to take into account the potential virtual network migration overheads by minimizing the likelihood of migrating already embedded virtual networks which arises due to substrate node mobility. Our work also matches the recent recommendations in [10] where an awareness of the potential migrations during the Virtual network embedding phase is needed to avoid the migration drawbacks. Unlike existing virtual network migration algorithms, if we integrate Bird-VNE with a migration solution (e.g. as in [9]), that solution shall become activated less frequently.

III. System Model and Design Objective

We abstract and model the substrate (physical) network, consisting of a set S of n nodes, as an undirected graph $\Phi = (S, L)$ where L is the set of substrate links with each link $l \in L$ corresponding to a connected pair of nodes $s, s' \in S$. We assume that each node $s \in S$ offers a processing capacity $C(s)$, and each link $l \in L$ offers a bandwidth capacity $C(l)$.

In what follows, let $\mathbf{R}$ be the set of all possible paths between all substrate node pairs, where a path $P(s, s')$ between two substrate nodes s and s' is a sequence of connected links (or pairs of nodes) in L. Throughout the paper, $P(s, s')$ (or sometimes P) will also refer to the set of all the links constituting the path. The path length, $|P|$, and the bandwidth capacity, $C(P) = \min_{l \in P} C(l)$, characterize P.

We also consider that the substrate nodes are mobile, and adopt the modified Random Way Point (RWP) mobility model proposed in [36] to model the substrate node mobility. This model describes the mobility of any substrate node s by an infinite sequence of quadruples $\{(\mathbf{X}_{i-1}, \mathbf{X}_i, C_i, W_i)_s\}_{i \in \mathbb{N}}$, where i denotes the i-th movement sample of node s. For every movement sample i, s moves from the starting waypoint $\mathbf{X}_{i-1}$ to the target waypoint $\mathbf{X}_i$ with velocity C_i. Upon arrival to the target waypoint $\mathbf{X}_i$, s waits W_i time units.

Given the waypoint $\mathbf{X}_{i-1}$, the node chooses the target waypoint $\mathbf{X}_i$ randomly such that the included angle θ_i between the vector $\mathbf{X}_i - \mathbf{X}_{i-1}$ and the abscissa is uniformly distributed in $[0, 2\pi]$ and the transition length $Z_i = \|\mathbf{X}_i - \mathbf{X}_{i-1}\|$ is Rayleigh distributed. The angles $\{\theta_1, \theta_2, \ldots\}$ are i.i.d., and the transition lengths $\{Z_1, Z_2, \ldots\}$ of a substrate node s are also i.i.d. with parameter λ_s and a CDF $P(Z_i < z) = 1 - \exp(-\lambda_s \pi z^2)$, $z > 0$.

Velocities C_i are generally i.i.d. random variables with arbitrary distributions. Even with randomly distributed velocities, it is sufficient for the purpose of this paper that $C_i \equiv C_s$, where C_s is a positive constant, equaling the average speed of substrate node s. Waiting times $\{W_1, W_2, \ldots\}$ of a substrate node s are also assumed to be i.i.d. exponential with parameter μ_s and a CDF $P(W_i < w) = 1 - \exp(-\mu_s w)$, $w > 0$

The following are important stochastic properties of the modified RWP [36]:

1) *Transition time* T_r, defined as the time a substrate node spends between two successive waypoints. For a substrate node s moving with constant velocity C_s, the Probability Distribution Function (PDF) of T_r is $f_{T_r}(t) = 2\pi\lambda_s C_s^2 t \exp(-\lambda_s \pi C_s^2 t^2)$ and the (Cumulative Density Function) CDF is $P(T_r < t) = 1 - \exp(-\pi\lambda_s t^2 C^2)$, $\lambda_s > 0$.
2) *Target waypoint distribution.* Given $\mathbf{X}_{i-1}$, the PDF of the target waypoint $\mathbf{X}_i$ in polar coordinates is given by

$$f_{\mathbf{X}_i}(r, \theta) = \lambda_s \exp(-\lambda_s \pi r^2). \quad (1)$$

We also assume that there exists a central node that is responsible for managing the substrate network and embedding the virtual network requests. That is, the central node will be receiving multiple different VNE requests in real time, and embedding them one at time. Each VNE request i is to be embedded for τ_i time units (i.e. τ_i is VNE i's service time).

A. Virtual Network Embedding

A VNE request can be represented as an undirected graph $\Upsilon = (V, E)$ where V is the set of the virtual nodes and E is the set of the virtual links (i.e. connected pairs of virtual nodes). In what follows, let $n_v = |V|$ and $m_v = |E|$. Each node $v \in V$ has a geographical location and a requested node stress $T(v)$ (e.g. processing capacity). Similarly, each virtual link $e \in E$ has a requested link stress $T(e)$ (e.g. link bandwidth). Table I summarizes the key notations.

Suppose that, at a given point in time, the central node has already received and successfully embedded a total of $k-1$ virtual network requests, $\Upsilon^{(1)}, \Upsilon^{(2)}, \ldots, \Upsilon^{(k-1)}$, and the k^{th} request, $\Upsilon^{(k)}$, has just arrived. The problem of embedding of the k^{th} virtual network $\Upsilon^{(k)} = (V^{(k)}, E^{(k)})$ into the substrate network Φ consists of the following two mappings.

Node mapping: maps each virtual node $v \in V^{(k)}$ to a distinct substrate node $s \in S$ subject to two constraints. One, s

TABLE I
SUMMARY OF NOTATIONS

Substrate Network		Random Way Point		Virtual Network	
Symbol	Definition	Symbol	Definition	Symbol	Definition
S	Set of substrate nodes	$\mathbf{X}$	A waypoint	τ_i	Service time
n	Number of nodes	C	Traveling velocity	Υ	Virtual network
Φ	Substrate network	W	Waiting time at $\mathbf{X}$	V	Set of virtual nodes
L	Set of substrate links	Z	Transition lenth	E	Set of virtual links
$C(s)$ and $C(l)$	Substrate capacities	λ	Rayleigh parameter	n_v and m_v	Number of virtual nodes/links
$\mathbf{R}$	Set of all paths	T_r	Transition time	$T(v)$	Processing demand
$P(s,s')$	A substrate path	$f_{\mathbf{X}_i}(r,\theta)$	Waypoint distribution	$T(e)$	Bandwidth demand

must be within Δ distance from v, where Δ is a parameter associated with the VNE request. Two, the sum of the requested processing capacities of all virtual nodes mapped to s (including those mapped from previous VNE requests) must not exceed the offered processing capacity of s. Formally, letting $Dist(u,v)$ denote the Euclidean distance between u and v, node mapping consists of finding a node mapping function, $\mathcal{M}\left(V^{(k)}\right): v \in V^{(k)} \mapsto \mathcal{M}(v) \in S$, such that $\mathcal{M}(v_i) = \mathcal{M}\left(v_j\right)$ iff $v_i = v_j$, $Dist(\mathcal{M}(v), v) \leq \Delta$ for all $v \in V^{(k)}$, and $\sum_{v \in \cup_{i=1}^{k} V^{(i)}: \mathcal{M}(v)=s} T(v) \leq C(s)$ for all $s \in S$.

Link mapping: maps each virtual link $e \in E^{(k)}$ to a substrate path $P \in \mathbf{R}$ subject to two constraints. One, the end virtual nodes of e must correspond to the end substrate nodes of P. Two, for every $l \in L$, the sum of the requested bandwidth capacities of all virtual links (including those belonging to previous VNE requests) whose mapped paths go through the substrate link l must not exceed the offered bandwidth capacity of l. Formally, link mapping consists of finding a link mapping function, $\mathcal{M}\left(E^{(k)}\right): e = (v,v') \in E^{(k)} \mapsto \mathcal{M}(e) = P(s,s') \in \mathbf{R}$, such that $\mathcal{M}(v) = s$, $\mathcal{M}\left(v'\right) = s'$, and $\sum_{e \in \cup_{i=1}^{k} V^{(i)}: l \in \mathcal{M}(e)} T(e) \leq C(l)$ for all $l \in L$.

Definition 3.1: The embedding of $\Upsilon^{(k)}$ is said to be feasible when both the node mapping and link mapping tasks defined above are successful.

Upon successfully embedding the k^{th} VNE request, the central node updates the locations of the substrate nodes, as well as the amounts of the available/remaining substrate resources. These are the remaining processing capacity of substrate node s, denoted by $R^{(k)}(s) = C(s) - \sum_{v \in \cup_{i=1}^{k} V^{(i)}: \mathcal{M}(v)=s} T(v)$, the remaining bandwidth capacity of substrate link l, denoted by $R^{(k)}(l) = C(l) - \sum_{e \in \cup_{i=1}^{k} V^{(i)}: l \in \mathcal{M}(e)} T(e)$, and the remaining path capacity of substrate path P, denoted by $R^{(k)}(P) = \min_{l \in P} R^{(k)}(l)$. Also, upon receiving a new VNE request, the central node constructs the mapping domains of the virtual nodes and links, which are defined as follows.

Definition 3.2: The mapping domain D_v of a virtual node $v \in V^{(k)}$ is defined to be the set of all substrate nodes whose Euclidean distances to v are each less than Δ and whose remaining processing capacities are each greater than $T(v)$; i.e., $D_v = \{s \in S : Dist(s,v) \leq \Delta, R^{(k)}(s) \geq T(v)\}$.

Definition 3.3: The mapping domain D_e of a virtual link $e = (v,v') \in E^{(k)}$ is defined to be the set of all substrate paths whose end nodes (s,s') are in $D_v \times D_{v'}$ and whose remaining capacities are each greater than $T(e)$; i.e., $D_e = \{P(s,s') \in \mathbf{R} : (s,s') \in D_v \times D_{v'}, R^{(k)}(P(s,s')) \geq T(e)\}$.

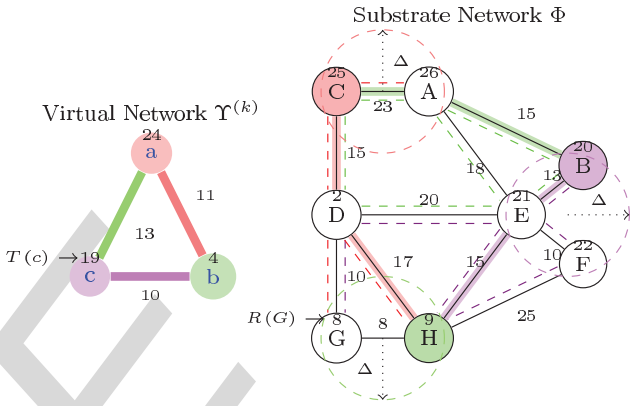

Fig. 1. Virtual Network Embedding: node mapping domains are shown in dashed circles ($radius = \Delta$) and link mapping domains are shown in dashed lines parallel to substrate paths.

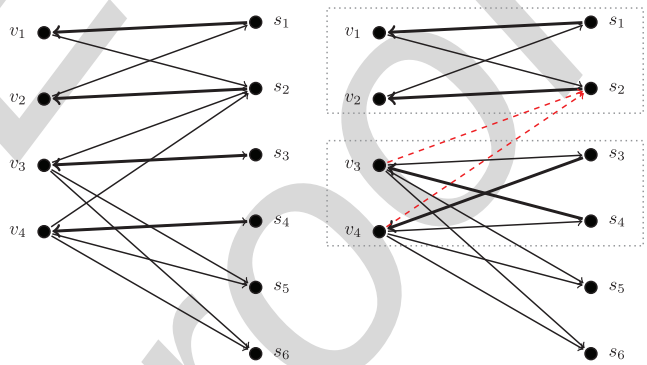

Fig. 2. Procedure 1 illustration. (a):A Maximum cardinality matching (thick edges), v is connected to s if $s \in D_v$, (b):Alternating graph with two strongly connected components. Edges crossing the strongly connected components cannot be in a maximum matching therefore Procedure 1 prunes them.

Figure 1 shows a VNE example, where the graph on the left side is the virtual network and that on the right side is the substrate network. In this example, the node mapping domains are $D_a = \{A, C\}$, $D_b = \{G, H\}$, and $D_c = \{B, E, F\}$, the link mapping domains are shown in dashed lines (e.g. $D_{(a,c)} = \{\{(A,B)\}, \{(A,E)\}, \{(A,E),(E,B)\}, \{(A,C),(C,D),(D,E)\}, \{(A,C),(C,D),(D,E),(E,B)\}, \{(A,B),(B,E)\}\}$). The VNE solution is given by (i) the node mappings, $\mathcal{M}(a) = C$, $\mathcal{M}(b) = H$, and $\mathcal{M}(c) = B$, and (ii) the link mappings, $\mathcal{M}((a,b)) = \{(C,D),(D,H)\}$, $\mathcal{M}((a,c)) = \{(C,A),(A,B)\}$, and $\mathcal{M}((b,c)) = \{(H,E),(E,B)\}$.

B. Probability of VNE Migration due to Node Mobility

If a virtual node v is mapped to a substrate node s, a migration is triggered when the distance $d = Dist(v, s)$ becomes greater than Δ. More specifically, a migration will not be triggered due to s's mobility if s stays within the circle $A(v, \Delta)$ of diameter Δ centered at v for a period longer than τ, the service time of the virtual network request incorporating node v. From (1), the probability that the target waypoint of the substrate node is within $A(v, \Delta)$ is, for $0 \leq d \leq \Delta$,

$$P(A(v,\Delta)) = \int_{\Delta-d}^{\Delta+d}\int_{0}^{2\pi} f_{\mathbf{X}_i}(r,\theta)\, r\,\mathrm{d}r\mathrm{d}\theta,$$
$$= \exp(-\pi\lambda_s(d-\Delta)^2) - \exp(-\pi\lambda_s(d+\Delta)^2).$$

Let $H(s)$ be the probability that a migration is triggered due to the mobility of substrate node s. $H(s)$ can be approximated as the probability that neither the target waypoint is within $A(v, \Delta)$ and the total time spent in $A(v, \Delta)$ is $\geq \tau$ nor the target waypoint is outside $A(v, \Delta)$ and the transition time to the boarder of $A(v, \Delta)$ is $\geq \tau$. Computing the PDF of the total time spent in $A(v, \Delta)$ $(W + T_r)$ requires convolution of the PDFs of W and T_r, and strong assumptions on relative values of λ_s, C_s, and τ, which are outside the control of the embedding algorithm. To simplify the analysis and the VNE objective design, we assume that: i) the time spent within $A(v, \Delta)$ is dominated by the waiting time at the target waypoint $\mathbf{X}_i$, ii) if the target waypoint is outside $A(v, \Delta)$, the whole transition time is spent within $A(v, \Delta)$, and iii) the waiting time of the starting waypoint has elapsed at the time of the virtual network embedding. Since we are mainly interested in evaluating the migration probability of a substrate node relative to other substrate nodes, the impact of these assumptions is minimal. With this, $H(s)$ can be expressed as

$$H(s) = 1 - P(W \geq \tau)P(A(v,\Delta)) - P(T_r \geq \tau)(1 - P(A(v,\Delta))) \quad (2)$$

To minimize the migration overhead, the VNE algorithm shall map virtual nodes to substrate nodes with the least migration probability, $H(s)$. Unlike traditional virtual network embedding and migration algorithms, this requires the estimation of the transition length and waiting time distribution parameters and the use of the estimated parameters to evaluate the migration probability associated with mapping a virtual node v to a substrate node s. The maximum likelihood estimation of the transition length parameter is $\hat{\lambda_s} = \frac{1}{4}(\overline{Z^2})^{-2}$, where the $\overline{Z^2}$ denotes the second sample moment of Z, and that of the waiting time parameter is $\hat{\mu_s} = \frac{1}{\overline{W}}$, where $\overline{W}$ denotes the sample moment of W.

C. VNE Design Objective

Our objective is to develop an algorithm that finds feasible VNEs while maximizing the embedding profit and minimizing the migration overhead. We say that a feasible embedding is optimal when its profit is maximum.[1] Given a virtual network Υ, the profit is defined as the difference between the revenue generated from embedding Υ and its embedding cost, i.e. Profit(Υ) = Revenue (Υ) − Cost (Υ).

To achieve the VNE design objective, we model the embedding cost to capture the cost of node mapping, the cost of link mapping, and the potential cost of migration that may arise as a result of mobility. It is defined as

$$\text{Cost}(\Upsilon) = \sum_{v\in V}\alpha T(v) + \sum_{e\in E}\beta T(e) \times |\mathcal{M}(e)| + \sum_{v\in V}\gamma(v)H(\mathcal{M}(v)), \quad (3)$$

where α and β denote the cost of processing and bandwidth resource units, respectively. The third term captures the cost of migration due to substrate nodes mobility, where $\gamma(v)$ is the cost of migrating the virtual node v. Intuitively, $\gamma(v)$ depends on the amount of resources allocated to v, as well as on v's connectivity to other virtual nodes.

We also define the revenue to be generated from successfully embedding Υ as

$$\text{Revenue}(\Upsilon) = \sum_{v\in V}\alpha' T(v) + \sum_{e\in E}\beta' T(e), \quad (4)$$

where α' and β' denote the price to be charged for each processing and bandwidth unit, respectively.

Observe that the embedding revenue in (4) depends only on the virtual network's requested resources and not on the VNE solution. Also recall that the function $H(\mathcal{M}(v))$ given in (3) represents the probability that a migration of v is triggered due to the mobility of the substrate node, $\mathcal{M}(v)$. It follows that maximizing the profit implies minimizing the embedding cost in (3), which implicitly minimizes the virtual network migration overhead due to mobility. Note that even though, in this paper, the function $H(\mathcal{M}(v))$ captures the likelihood of migration that is due to mobility, it can be used to represent/capture the migration due to any other network dynamics, like link failure.

IV. Enforcing Domain Consistency

The node and link mapping domains, defined in Definitions 3.2 and 3.3, involve coupled constraints. A mapping of a virtual node v to a substrate node $s \in D_v$ impacts other nodes and links mapping domains in several ways. First, no other virtual nodes can be mapped to s. Second, we can only map virtual links that have v as an end node to substrate paths that have s as an end node. Moreover, a mapping of a virtual link e to a substrate path $P \in D_e$ restricts other virtual links from being mapped to the substrate paths that share one or more substrate links with P. The shared links become capacity bottlenecks as their bandwidth capacity must be greater than the required bandwidth of not only e but also other virtual links mapped to paths sharing these links. A backtracking algorithm resolves such constraint couplings by mapping virtual nodes

[1] Modeling the objective as a maximization problem allows us to analytically bound the objective value, as shown later in Section V.

and virtual links one at a time, and backtracking to previous steps when the algorithm encounters an unfeasible mapping.

A VNE algorithm can avoid backtracking (backtrack-free search) if the mapping domains of all virtual nodes and links are consistent. Enforcing domain consistency involves pruning the node and link mapping domains to avoid mappings that lead to an unfeasible embedding. Unfortunately, the use of the standard consistency propagation algorithms are exponential in time. This is because the constraint network of the VNE problem has a maximum degree that is a function of n, while the running time of the standard consistency propagation algorithm, to ensure backtrack-free search, is exponential in the maximum degree of the constraint network (see [18] for details).

Fortunately, constraint propagation algorithms can take advantage of certain properties specific to VNE to prune the mapping domains in polynomial time through mapping domains consistency enforcement. In this section, we develop techniques that exploit these properties to avoid backtracking during the VNE search process, and use these techniques to design a polynomial time, *almost* backtrack-free VNE algorithm. There are two types of mapping domains consistency, virtual network topological consistency and substrate paths capacity consistency, which are presented next.

A. Virtual Network Topological Consistency

We first enforce domain consistency to ensure that the topology of the resulting solution (node and link mappings) matches exactly the topology of the virtual network, i.e. topological consistent. This requires enforcing the following: (i) substrate nodes mapped to the virtual nodes must be *all different*, (ii) end nodes of the substrate paths in link mapping domains must have corresponding substrate nodes in the node mapping domains and vice versa, and (iii) substrate nodes in the node mapping domains must maintain similar virtual node degrees.

Alldifferent virtual node mapping constraint: The constraint to map virtual nodes to distinct substrate nodes is known as the alldifferent constraint in the constraint programming context, and we next state a useful corollary following from Reégin's theorem [37] on the alldifferent constraint.

Corollary 4.1: A virtual node mapping $v \in V \mapsto s \in D_v$ leads to an unfeasible embedding if the edge (v, s) does not belong to a maximum matching that covers all the virtual nodes in the bipartite graph $B = (V \cup S, \{(v, s) : \mathcal{M}(v) = s\})$.

The above corollary can then be exploited to prune away nodes and links from the node and link mapping domains, and for completeness, we provide in Procedure 1 a brief description of such a pruning technique, which we term ALLDIFFERENT [37].

In Procedure 1, a residual graph, B', is defined as $B' = (V \cup S \cup \{t\}, M \cup E_2 \cup E_3 \cup E_4)$ where M is the set of edges in the matching directed from virtual nodes to substrate nodes, E_2 is the set of edges that are not in the matching M and are directed from substrate nodes to virtual nodes, E_3 is the set of all directed edges from substrate nodes in the matching M to a dummy node t, and E_4 is the set of all directed edges from t to substrate nodes that are not in the matching M.

Procedure 1. AllDifferent

Input: $V,\ D_{v \in V}$
Ensure: Distinct virtual node to substrate node mappings in $O\left(n_v^{1.5} n\right)$ [37].
1: Construct bipartite graph $B = (V \cup S, \{(v, s) : \mathcal{M}(v) = s\})$
2: Find a maximum matching M in B using Hopcroft-Karp algorithm [38]
3: **if** $|M| < n_v$ **then**
4: Return no feasible embedding for the given mapping domains
5: **end if**
6: Construct the residual graph B'
7: Compute the strongly connected components in B'
8: Prune the node mapping domains by deleting any edges connecting two different strongly connected components in B'.
9: **return** Narrowed virtual node mapping domains

Step 8 in Procedure 1 prunes substrate nodes from the node mapping domains that can never lead to distinct node mappings. Any edge connecting two different strongly connected components in B' corresponds to a mapping from a virtual node v to a substrate node s and does not belong to any maximum cardinality matching, hence it is not possible to find a feasible embedding with distinct node mapping if v was mapped to s. Thus, s must be removed from D_v. The time complexity of Procedure 1 is bounded by the time required to find the maximum matching using the Hopcroft-carp algorithm in step 2. Since a virtual node can have at most n substrate nodes in its node mapping domain, the number of edges in the bipartite graph B cannot exceed $n_v \times n$ edges. In the worst case, the Hopcroft-carp algotihm requires $O(\sqrt{n_v} n_v n)$ steps, hence the ALLDIFFERENT time complexity is $O\left(n_v^{1.5} n\right)$.

Relational consistency of node and link mapping domains: In the example of Fig. 1, although mapping the virtual node c to F is feasible, doing so prevents us from finding a mapping to the virtual link (a, c), as there is no substrate path between F and any substrate node in the node mapping domain D_a.

From the definition of mapping domains, we can easily observe that if two virtual nodes v, v' are connected by a virtual link e, then the end points of the substrate paths in the virtual link mapping domain D_e is a subset of the cross product of the virtual node mapping domains $D_v \times D_{v'}$. We can now rely on this simple observation and the definition of the virtual link mapping domains to conclude the following:

Lemma 4.2: The node mapping $v \in V \mapsto s \in D_v$ leads to an unfeasible embedding if there exists a link $e = (v, v') \in E$ whose link mapping domain D_e does not contain a path ending at s. Similarly, a virtual link mapping $e = (v, v') \mapsto P(s, s')$ leads to an unfeasible embedding *if* $s \notin D_v$ or $s' \notin D_{v'}$.

Proof: Assume $v \mapsto s$ and a subsequent mapping of $e = (v, v')$ such that there is no path $P \in D_e$ ending at s. A mapping of e to any substrate path in D_e results in mapping multiple virtual nodes to the same substrate node. Also, $e = (v, v') \mapsto$

Procedure 2. Node-Consistency

Input: $E,\ D_{e\in E},\ D_{v\in V}$
Ensure: Virtual node mapping domains are consistent with virtual link mapping domains in $O(m_v n)$
1: **for all** virtual link $e = (v, v') \in E$ **do**
2: $D_v \leftarrow D_v \cap u(D_e)$
3: $D_{v'} \leftarrow D_{v'} \cap v(D_e)$
4: **end for**
5: **return** Narrowed virtual node mapping domains

Procedure 3. Link-Consistency

Input: $E,\ D_{e\in E},\ D_{v\in V}$
Ensure: Virtual link mapping domains are consistent with virtual node mapping domains in $O(m_v n^2)$
1: **for all** virtual link $e = (v, v') \in E$ **do**
2: **for all** substrate path $P \in D_e$ **do**
3: if $u(P) \notin D_v \vee v(P) \notin D_{v'}$ **then**
4: $D_e \leftarrow D_e \setminus \{P\}$
5: **end if**
6: **end for**
7: **end for**
8: **return** Narrowed virtual link mapping domains

$P(s, s')$ violates the link mapping Definition 3.3 if either $s \notin D_v$ or $s' \notin D_{v'}$. ■

Using Lemma 4.2, we propose two procedures to narrow down the node and link mapping domains: Procedures 2 and 3. The functions $u(D_e)$ and $v(D_e)$ return the sets respectively of the first and the second end nodes of all the paths in D_e. When applied to a path P, $u(P)$ and $v(P)$ return the path's first and second end nodes. In each iteration, Procedure 2 prunes the substrate nodes from the node mapping domains of the end nodes of the virtual links, if there is not any substrate path in their link mapping domains that also ends at those substrate nodes. Since for each virtual link the intersection operator (step 2 and 3) requires at most $O(n)$ steps as $|D_v| \leq n$, then Procedure 2 has a worst case time complexity of $O(m_v n)$.

Procedure 3 complements Procedure 2 by pruning a substrate path from the link domain of a virtual link if the substrate nodes ending that path cannot be found in the node mapping domains of the virtual nodes ending the virtual link. Since there are at most $O(n^2)$ paths in the substrate network, the inner loop (step 2 to 6) of Procedure 3 requires at most $O(n^2)$ steps. Hence, the worst case time complexity of Procedure 3 is $O(m_v n^2)$.

Consistency of virtual and substrate node connectivity: The relational consistency of node and link mapping domains does not ensure connectivity of the virtual network, nor does it imply that the mapping domains can satisfy the virtual network connectivity requirements, especially when the node mapping domains overlap. To illustrate this, consider a new induced network of substrate nodes that represents the connectivity of the virtual link domains. In this induced network, substrate nodes are connected by an edge if there exists a path belonging to any link mapping domain that connects them. Induced network is defined formally next.

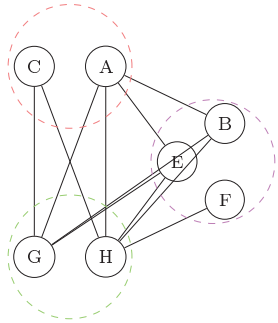

Fig. 3. Induced network I from substrate network Φ in Fig. 1. I has one connected components CC_I and three supernodes (dashed circles). $\zeta(CC_I) = 3$, $\delta(F) = 1$ and equals 2 for all other nodes.

Definition 4.1: Given a virtual network Υ, we define the induced network I of Υ as the undirected graph $I = (S_I \subset S, L_I)$ where $S_I = \cup_{v\in V} D_v$ and $L_I = \{(s, s') \in S_I^2 : \exists\ P(s, s') \in D_e$ for some $e \in E\}$.

Definition 4.2: For every connected component CC_I of I, the set $N_v(CC_I) = CC_I \cap D_v$ corresponding to the virtual node v is called the supernode of v. Let $\zeta(CC_I)$ be the number of distinct supernodes in CC_I. For every $s \in S_I$, we define $\delta(s)$ as the number of supernodes connected to s.

Fig. 3 illustrates the induced network of the example given in Fig. 1. This induced network is constructed by connecting a pair of substrate nodes in Fig. 3 when there is at least one path connecting them in any link mapping domain. In general, if the mapping $v \mapsto s$ is feasible, the function $\delta(s)$ reflects the degree of the virtual node v, and if a connected component CC_Υ of Υ is mapped to a subset of substrate nodes in Φ, the function $\zeta(CC_I)$ reflects the number of virtual nodes in the connected component CC_Υ (size of CC_Υ).

Lemma 4.3: Let $Deg_\Upsilon(v)$ denote the degree of virtual node v. A virtual node mapping $v \mapsto s$ leads to an unfeasible embedding *if* $Deg_\Upsilon(v) > \delta(s)$ or the size of the connected component of Υ (CC_Υ) that contains v is greater than the number of supernodes in CC_I that contains s.

Proof: Assume $v \mapsto s$ and $Deg_\Upsilon(v) > \delta(s)$, then there exist at least one virtual link e such that there is no substrate path P in D_e with one of its end substrate nodes equals s. Then, $v \mapsto s$ does not lead to a feasible embedding from Lemma 4.2. If $Deg_\Upsilon(v) \leq \delta(s)$ but $|CC_\Upsilon| > \zeta(CC_I)$, then there must exist an unmapped virtual node $v' \in CC_\Upsilon$, while all substrate nodes $s \in CC_I$ are already mapped to other virtual nodes in CC_Υ including v. Since v' must be mapped to one substrate node in CC_I to maintain connectivity, then mapping $v \mapsto s$ does not lead to a feasible embedding. ■

The DEGREE-CONSISTENCY procedure (Procedure 4), a direct application of Lemma 4.3, is a pruning technique that narrows down mapping domains through degree consistency enforcement. Its complexity is bounded by computing $\delta(s)$ for all the substrate nodes in the virtual node mapping domains, which is $O(n^2)$.

Running the ALLDIFFERENT, NODE-CONSISTENCY, LINK-CONSISTENCY, and DEGREE-CONSISTENCY procedures for one iteration removes some inconsistent mappings from the node and link mapping domains. To remove all

Procedure 4. Degree-Consistency

Input: $E,\ D_{v\in V}$
Ensure: Degree Consistency in $O(n^2)$
1: **for all** virtual nodes $v' \in V$ **do**
2: **for all** substrate nodes $s' \in D_{v'}$ **do**
3: **if** $Deg_{\Upsilon}(v') > \delta(s')$ **then**
4: $D_{v'} \leftarrow D_{v'} \setminus \{s'\}$
5: **end if**
6: **end for**
7: **end for**
8: **for all** connected component $CC_{\Upsilon} \in \Upsilon$ **do**
9: **for all** connected component $CC_I \in I$ **do**
10: **if** $|CC_{\Upsilon}| > \zeta(CC_I)$ **then**
11: $D_{v'} \leftarrow D_{v'} \setminus CC_I,\ \forall v' \in CC_{\Upsilon}$
12: **end if**
13: **end for**
14: **end for**
15: **return** Narrowed virtual nodes domains

Algorithm 1. Topology-Consistency

Input: $E,\ D_{e\in E},\ D_{v\in V}$
Ensure: Topology Consistency in $O(m_v n^3)$
1: **repeat**
2: NODE-CONSISTENCY($E,\ D_{e\in E},\ D_{v\in V}$)
3: ALL DIFFERENT($V,\ D_{v\in V}$)
4: LINK-CONSISTENCY($E,\ D_{e\in E},\ D_{v\in V}$)
5: DEGREE-CONSISTENCY($E,\ D_{v\in V}$)
6: **if** $\exists\ D_{v'} = \emptyset,\ \forall v' \in V\ \vee\ D_e = \emptyset,\ \forall e \in E$ **then**
7: **return** false {T}here exist a virtual node or a virtual link with an empty mapping domain.
8: **end if**
9: **until** No node or link domain is changed
10: **return true**

the inconsistencies, these procedures must repeatedly be run sequentially until no further removal is possible from either the node or the link mapping domains. The process merging all these four procedures is captured in Algorithm 1, which essentially removes inconsistency, and hence avoids backtracking, by ensuring topological consistency of the node and link mapping domains. This algorithm is referred to as TOPOLOGY-CONSISTENCY.

The complexity of TOPOLOGY-CONSISTENCY is bounded by the number of times we run the procedure LINK-CONSISTENCY in step 4. This implies a complexity of $O(m_v n^2)$ in each iteration. In the worst-case scenario, TOPOLOGY-CONSISTENCY removes one substrate node from one node mapping domain and this corresponds to at least one removal of one substrate path from link mapping domains. Hence, it requires at most n iterations to remove all the substrate nodes from one node mapping domain, thus returning false.[2] Thus, the complexity of TOPOLOGY-CONSISTENCY is $O(m_v n^3)$. But since the maximum number of virtual nodes is the number of substrate nodes; i.e., $n_v \leq n$, then the complexity of TOPOLOGY-CONSISTENCY is $O(n^5)$.

[2] A more efficient implementation checks the condition in step 6 every time any procedure removes a substrate node/link from a mapping domain.

B. Capacity Disjoint Paths Consistency

We now study the second mapping domain consistency type, substrate paths capacity consistency. Let us refer again to the example given in Fig. 1 and consider the link mapping sequence $(a, b) \mapsto P(C, H) = \{(C, D), (D, H)\}$ and $(a, c) \mapsto P(C, E) = \{(C, D), (D, E)\}$. The remaining bandwidth of the substrate link (C, D), $R((C, D)) = 15$, is less than the sum of the links' requested bandwidth capacities, which is $T((a, b)) + T((a, c)) = 24$. Hence, this mapping sequence is unfeasible. Clearly, a VNE algorithm will not backtrack if all substrate paths in the link mapping domains are disjoint (if topological consistency is enforced). However, constructing the link mapping domains from disjoint paths results in a degradation of the VNE acceptance rate (such a rate reflects the number of virtual networks that can be embedded into the substrate network), as well as in an increase in the embedding cost. Our proposed embedding algorithm does not force paths to be disjoint so as to increase the acceptance rate and decrease the embedding cost. Instead, our technique relies on the concept of *capacity disjoint* which we formally define next.

Definition 4.3: For every substrate link l, let $\bar{D}_e(l) = \{P \in D_e : P \ni l\}$ and $\bar{E}(l) = \{e \in E : \bar{D}_e(l) \neq \emptyset\}$. We say that the paths in $\mathbf{R}' = \bigcup_{e\in\bar{E}(l)} \bar{D}_e(l)$ are *capacity disjoint iff* the remaining bandwidth capacity of l is greater than the sum of the requested bandwidth capacities of all the virtual links in $\bar{E}(l)$. Formally, the paths in $\mathbf{R}'$ are said to be capacity disjoint *iff* $R^{(k)}(l) \geq \sum_{e\in\bar{E}(l)} T(e)$.

Lemma 4.4: A virtual link mapping $e \mapsto P$ leads to an unfeasible embedding *if* all the substrate paths in every unmapped virtual link's mapping domain are not capacity disjoint with P.

Proof: If a virtual link $e_i \mapsto P_i$ and in a next mapping step of virtual link e_j, all paths in D_{e_j} are not capacity disjoint with P_i, then any mapping $e_j \mapsto P_j \in D_{e_j}$ will result in at least one substrate link with negative remaining bandwidth. ■

Theorem 4.5: The proposed TOPOLOGY-CONSISTENCY algorithm ensures a backtrack-free search if all substrate paths in all link mapping domains are capacity disjoint.

Proof: It follows from Lemmas 4.2, 4.3, and 4.4 and from Corollary 4.1. ■

An algorithm that aims to ensure a backtrack-free search may remove substrate paths that are not capacity disjoint from the virtual links mapping domains. Although such an algorithm will have a complexity advantage because it is backtrack-free, it degrades the acceptance rate and the cost as it will remove substrate paths that can actually lead to feasible or minimum cost embedding. Apparently, capacity disjoint paths condition is required only for substrate paths that are actually in an incurred embedding. In order to overcome the complexity problem while still minimizing the cost and maximizing the acceptance rate, we propose Algorithm 2 (CAPACITY-DISJOINT),

Algorithm 2. Capacity-Disjoint

Input: $E,\ L,\ D_{e\in E},\ D_{v\in V}$
Ensure: Substrate paths are capacity disjoint if they are likely to coexist in an incurred embedding in $O(m\, m_v\, n^3)$.
1: **for all** $l \in L:\ \exists\ ps \in D_{e\in E},\ l \in P$ **do**
2: repeat
3: NODE-CONSISTENCY($\bar{E}(l),\ \bar{D}_e \in \bar{E}(l),\ \bar{D}_v \in \bar{V}(l)$)
4: ALL DIFFERENT($\bar{V}(l),\ \bar{D}_v \in \bar{V}(l)$)
5: LINK-CONSISTENCY($\bar{E}(l),\ \bar{D}_e \in \bar{E}(l),\ \bar{D}_v \in \bar{V}(l)$)
6: **until** No node or link sub-domain is changed
7: $R'(l) \leftarrow R(l)$
8: **for all** $e \in \bar{E}(l)$ ordered ascendingly by $|D_e|$ **do**
9: **if** $\bar{D}_e(l) \neq \emptyset$ **then**
10: $R'(l) \leftarrow R'(l) - T(e)$
11: **if** $R'(l) < 0$ **then**
12: $D_e \leftarrow D_e \setminus \bar{D}_e(l)$
13: **end if**
14: **end if**
15: **end for**
16: **end for**
17: **if** $\exists\ D_e = \emptyset,\ \forall e \in E$ **then**
18: **return false**
19: **end if**
20: **return true**

which ensures that substrate paths in link mapping domains are capacity disjoint if they are likely to coexist in an incurred embedding.

The key idea of the CAPACITY-DISJOINT algorithm is to determine the worst case scenario in which the intersecting substrate paths in $\mathbf{R}'$ can become simultaneous mappings of virtual links in $\bar{E}(l)$. These paths are found by applying topological consistency procedures, discussed earlier, on the subsets of link and node mapping domains $\bar{D}_e \in \bar{E}(l),\ \bar{D}_v \in \bar{V}(l)$ (Steps 1 to 6), where $\bar{V}(l) \subset V$ is the set of end virtual nodes of virtual edges in $\bar{E}(l)$ and $\bar{D}_v(l) \subset D_v$ is the set of substrate node mappings deduced from $\mathbf{R}'$.

The CAPACITY-DISJOINT algorithm checks if all paths that are common to every substrate link l are capacity disjoint. If not, the algorithm removes first the substrate paths $\bar{D}_e \in \bar{E}(l)$ from the domain of the virtual link(s) e that has the largest link mapping domain size $|D_e|$ (Step 7 to 16). This is to minimize the chances of ending up with an empty link mapping domain, thus maximizing the acceptance rate. Although it is clear that CAPACITY-DISJOINT algorithm does not eliminate backtracking entirely, it substantially reduces its likelihood of occurrence. We evaluate the likelihood of backtracking empirically in Section VI.

The CAPACITY-DISJOINT algorithm uses similar steps to determine possible simultaneous intersecting paths (Steps 2 to 6) for each substrate link l that intersects with some paths. Although these steps are performed on a subset of the mapping domains and it is unlikely to encounter the situation that every substrate link is a common link for all paths (as the substrate network will almost look like a path), the complexity of CAPACITY-DISJOINT is bounded by $O(m\, m_v\, n^3)$. This can be expressed as $O(n^7)$ if both the substrate and virtual networks are complete graphs and have the same number of nodes n.

V. APPROXIMATE PROFIT MAXIMIZATION

TOPOLOGY-CONSISTENCY and CAPACITY-DISJOINT algorithms, discussed in the previous section, reduce the search space and improve the running time of backtracking search. However, even in the case of backtrack-free search, an optimal optimization algorithm, like branch and bound, may still traverse the whole search space through brute-force [18]. If we assume that the VNE problem is backtrack-free, it can be viewed as the maximum weight matching problem in a bipartite graph. The bipartite graph in this case is the set of virtual links on one side of the bipartite graph connected by weighted edges to the set of substrate paths on the other side and the edge weight is the profit attained by mapping a virtual edge to a substrate path. From (3) and (4), the profit of mapping a virtual edge $e = (v, v')$ to a substrate path $P = (s, s')$ is given by

$$\begin{aligned}\text{Profit}(e, P) =&(\alpha' - \alpha)(T(v) + T(v')) \\ &+ (\beta' - \beta \times |P|)T(e) \\ &- \gamma(v)H(s) - \gamma(v')H(s').\end{aligned}$$

However, a direct application of conventional maximum weight matching algorithms (e.g. Hungarian methods or Edmond's methods) is non-trivial. Fortunately, greedy approximations to the maximum weight matching are applicable, but with some needed modifications to enforce domain consistency and to verify solution feasibility in every step. We use this observation to propose Algorithm 3, which finds a VNE such that the incurred embedding profit is at most as half as the optimal profit in an attainable special case and at least $\frac{1}{n_v}$ in general.

Our proposed VNE algorithm, termed BIRD-VNE, starts by enforcing the mapping domains consistency using TOPOLOGY-CONSISTENCY and CAPACITY-DISJOINT. It then searches for an embedding by mapping the virtual links with the smallest link mapping domain sizes and greatest demands (Line 9) first to the substrate paths in their domains with the highest profit (Line 10). After mapping each virtual link (mapping step), the algorithm ensures feasible embedding according to Definition 3.1 (Line 20). If any mapping step results in unfeasible embedding, the algorithm starts over the mapping process from the first virtual link by assigning it to an unattempted mapping in its domain until a feasible embedding is found or all mappings of the first link are tried.

This algorithm still involves a simplified form of backtracking. The algorithm always backtracks to the first virtual link mapping step. In this case, the total number of backtracks is bounded in the worst case by the size of the smallest link mapping domain. Typically, the consistency enforcing algorithms reduce the number of backtracks significantly as we will show empirically in Section VI. The following theorem bounds the worst case performance of BIRD-VNE.

Theorem 5.1: In the worst case, BIRD-VNE is an $O(\frac{1}{n_v})$-approximation to the optimal embedding profit, and

Algorithm 3. BIRD-VNE

Input: $\Upsilon = (V, E)$, $\Phi = (S, L)$ **Input:**
Require: $D_{\forall e \in E}$, $D_{\forall v \in V}$
Ensure: Embedding $\Upsilon \mapsto \Phi$ in $O(m\, m_v\, n^3)$
1: *SolutionExist* ← TOPOLOGY-CONSISTENCY
2: *SolutionExist* ← *SolutionExist* **and** CAPACITY-DISJOINT
3: *SolutionExist* ← *SolutionExist* **and** TOPOLOGY-CONSISTENCY
4: **if not** *SolutionExist* **then**
5: **return** "Reject virtual network."
6: **end if**
7: **repeat**
8: $\mathcal{M}(e) \leftarrow \emptyset,\ \forall e \in E$
9: **for all** $e = (v, v') \in E$ ordered ascendingly by $|D_e|$, and by $T(e)$ **do**
10: **for all** $P = (s, s') \in D_e$ ordered descendingly by Profit(e, P) **do**
11: **if** e is the first virtual link in the order of E **then**
12: $D_e \leftarrow D_e \setminus P$
13: **end if**
14: **if** $e \mapsto P$ result in a feasible embedding **then**
15: $\mathcal{M}(e) \leftarrow P, \mathcal{M}(v) \leftarrow u(P), \mathcal{M}(v') \leftarrow v(P)$
16: **break**
17: **end if**
18: **end for**
19: **end for**
20: **until** Feasible embedding is found or all first virtual link mapping domain are attempted.
21: **if** No feasible embedding is found **then**
22: **return** "Reject virtual network."
23: **end if**
24: **return** $\mathcal{M}(e),\ \forall e \in E$ and $\mathcal{M}(v),\ \forall v \in V$

only a $\frac{1}{2}$-approximation if there are, on average, n_v paths of the same length between any two substrate nodes.

Proof: Let x be the profit of mapping the first virtual link e to the highest profitable path P in its link mapping domain in a single iteration (step 7). The following potential mappings become invalid and will never be attempted by the algorithm until a backtracking to step 7 is decided: (i) mapping e to other substrate paths in its link mapping domain except P, (ii) mapping any other virtual link to P, (iii) mapping another virtual link e' that shares one of its end virtual nodes with e with any other substrate paths except those that also share the same end substrate node with P. Let d be the maximum degree of the virtual network, the worst case will occur if we have exactly d paths of shortest length (highest profit) and the algorithm invalidates at most d mappings of the optimal mappings (at most in the first mapping). In this case, the sum of profits of the invalidated mapping cannot exceed dx. Since the profit is non-negative, the approximation ratio is $O(\frac{1}{d})$ or more conservatively $O(\frac{1}{n_v})$. However, if there are n_v redundant substrate paths of the same length between any two substrate nodes, BIRD-VNE invalidates at most two mappings that may be optimal. This can be repeated for at most $\frac{1}{2}m_v$ of the steps (9 to 19) and the sum of the profits of the invalidated mappings cannot exceed $2x$. In this later case, the approximation ratio is $\frac{1}{2}$, which proves the theorem. ■

1) Scalability and Implementation Consideration: The complexity of BIRD-VNE is analyzed as follows. The main loop (Step 10 to 25) has m_v iterations. In the worst-case scenario, for every virtual link, it checks the feasible mappings of n^2 paths. Then, the complexity of BIRD-VNE is bounded by the CAPACITY-DISJOINT complexity $O(m\, m_v\, n^3)$ and can be written as $O(n^7)$. Although BIRD-VNE is polynomial in time and scales much better than the state-of-the art algorithms, its $O(n^7)$ complexity may prevent applying it to very large scale networks. Fortunately, this complexity bound can be improved through simple but effective implementation improvements.

The two procedures, NODE-CONSISTENCY and LINK-CONSISTENCY, can be easily implemented in parallel by implementing these algorithms on exactly m_v processing agents. In this case, the NODE-CONSISTENCY complexity is reduced to $O(n)$ while the LINK-CONSISTENCY complexity is reduced to $O(n^2)$. It then follows that the complexity of TOPOLOGY-CONSISTENCY is bounded by running ALLDIFFERENT at most n times, hence it is $O\left(n_v^{2.5} n\right)$. Similarly, the CAPACITY-DISJOINT complexity is also bounded by running ALLDIFFERENT at most m times, hence it is $O\left(n_v^{2.5} m\right)$. The complexity of CAPACITY-DISJOINT bounds the overall complexity of BIRD-VNE to $O\left(n_v^{2.5} m\right)$.

We can also improve the actual approximation ratio in practice by repeating step 7 to 19 until all virtual link mappings of the first virtual link are attempted (i.e. remove steps 14 to 17) while maintaining all the feasible solutions. We then pick the solution with the maximum total profit as our solution and the other solutions as backup solutions in case a migration is needed. This trick reduces the gap between the evaluated total profit and the optimal solution when compared to the worst case scenario, and preserves the same worst case complexity at the expense of the actual execution time.

2) Virtual Network Migration Consideration: The proposed algorithm, BIRD-VNE, can still be used, with simple modification, if virtual network migration is needed. If the previously discussed implementation in Section V-1 is adopted, we end up with multiple solutions to the same virtual network that can be quickly evaluated for feasibility, so as to choose one of these VNE solutions for migrating the virtual network instead of evaluating BIRD-VNE again from the beginning. Moreover, the following simple procedures can be carried out to perform migrations to individual nodes and links instead of migrating the whole virtual network.

Consider the case that a substrate path P is not capable of meeting the required demand of a virtual link e. This situation can happen, for example, in case of a link failure or congestion along the path, or failure of one or both end substrate nodes of P. In this case, we can immediately find another backup path (and substrate nodes if necessary), P', in $D_e \{P\}$ that has the largest profit and is also feasible with the current embedding $\mathcal{M}(e'),\ \forall e' \in E \{e\}$. This algorithm is as simple as running the steps from 9 to 19 in BIRD-VNE, while replacing D_e with $D_e \{P\}$ for only the virtual links that are impacted by the failure

of P. If this fails, the whole embedding needs to be performed again by running BIRD-VNE.

VI. NUMERICAL RESULTS

The effectiveness of the proposed algorithm, BIRD-VNE, is assessed in terms of the metrics suggested in [39]:

1) *Acceptance rate,* defined as the ratio of the total accepted virtual networks to the total requested virtual networks.
2) *Revenue to Cost ratio (R/C),* defined as $R/C = \sum_{\Upsilon} \text{Revenue}(\Upsilon) / \sum_{\Upsilon} \text{Cost}(\Upsilon)$.
3) *Average node and link utilization,* defined as $\sum_{s \in S} \frac{R(s)-C(s)}{nC(s)}$ and $\sum_{l \in L} \frac{R(l)-C(l)}{mC(l)}$, respectively.

In addition, we use the following metrics to assess the effectiveness of BIRD-VNE vis-a-vis of its ability to avoid backtracking, limit network migration, and achieve optimal VNE by comparing it to the optimal Brand and Bound technique.

1) *Average/Maximum Approximation ratio,* defined as the average/maximum ratio of the cost achieved by BIRD-VNE to that achieved by Branch and Bound.
2) *Backtrack-free ratio,* defined as the ratio of the total number of times in which BIRD-VNE finds a feasible embedding at the first attempt of the first virtual link mapping to the total number of accepted requests.
3) *Migration ratio,* defined as the ratio of the total number of virtual network migrations to the total number of accepted requests.

A. Simulation Setup

We compare the performance of BIRD-VNE with two existing algorithms, Randomized Virtual Network Embedding with shortest path link mapping (RVINE-SP) and with multicommodity flow link mapping (RVINE-MCF) [21], which are integrated to an event-driven simulator that we developed .[3] We also compare the performance achievable under BIRD-VNE to that achievable under the basic Greedy algorithm, referred to as BASELINE and proposed in [20].

The simulator generates Φ and Υ according to Erdös−Rènyi model. Similar to [21], Φ has 0.5 probability of connecting any two substrate nodes, $n = 50$, $C(s) \sim U(0, 50), \forall s \in S$ and $C(l) \sim U(0, 50), \forall l \in L$. Substrate nodes are placed randomly on a (25×25) grid. The mean inter-arrival time of virtual networks ranges from 5 to 25 networks per time unit, and the average service time is set to $\tau = 1000$ time units. Each pair of virtual nodes in Υ is connected with 0.5 probability, $n_v \sim U(1, 10)$, $\Delta = 15$, $T(v) \sim U(0, 20), \forall v \in V$ and $T(e) \sim U(1, 50), \forall e \in E$. The routing of the substrate network $\mathbf{R}$ is computed once in the prepossessing initialization step using the all shortest path algorithm. All the cost parameters α, β, γ are set to unity in the simulations.

We simulate the mobility of substrate nodes by setting $\tau = 50$, and the average waiting time at each waypoint to $\mu_s^{-1} = 100$ time units for all the substrate nodes. All substrate nodes travel with the same constant speed $C_i = 5$ speed units, and the average transition length of all the nodes is 5 length units (i.e. $\frac{\lambda_s^2}{2} = 5$). We consider a wireless network infrastructure in which the connectivity between the substrate nodes are not impacted by their mobility since fixed clones of the mobile nodes actually execute the virtual network requests in a geographically distributed cloud infrastructure as discussed in Section VII and as illustrated in Fig. 11.

[3]Implementations of RVICE-SP and RVINE-MCF are online available at http://www.mosharaf.com/ViNE-Yard.tar.gz

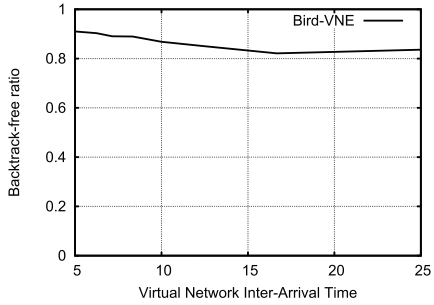

Fig. 4. Backtrack-free ratio of BIRD-VNE shows the effectiveness of search space pruning.

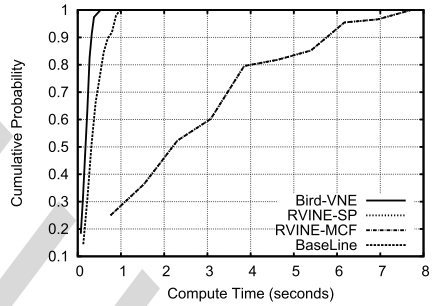

Fig. 5. Experimental computation time CDF of BIRD-VNE shows its computation effectiveness.

B. Performance Evaluation

1) BIRD-VNE *Improves the Acceptance Rate:* Fig. 8 shows that BIRD-VNE has a 15% better acceptance rate when compared to the other algorithms. The improvement in the acceptance rate is a direct result of Theorem 4.5 and is consistent for different loads. BIRD-VNE is likely to find a feasible embedding once it passes the consistency enforcement steps 1 to 6.

On the other hand, RVINE-SP and RVINE-MCF first rely on LP relaxations to solve the non-convex MIP problem, and then round the solution of the relaxation to the nearest integer. This way, RVINE-SP and RVINE-MCF may unnecessarily reject a VNE request by falsely concluding that it cannot be embedded. Moreover, when there is no solution, RVINE-SP and RVINE-MCF tend to spend a significant amount of time searching for solutions before eventually rejecting unfeasible requests as shown in Fig. 4-b.

2) BIRD-VNE *Avoids Backtracking:* Fig. 4-a shows the backtrack-free ratio of BIRD-VNE. BIRD-VNE is unlikely to encounter a backtracking, and finds a feasible solution from the first attempt. In this simulation setup, the backtrack-free ratio

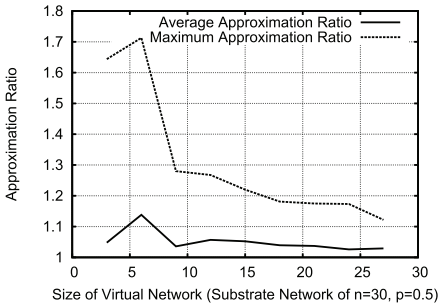

Fig. 6. BIRD-VNE Maximum and average approximation Ratio (optimal is branch and bound).

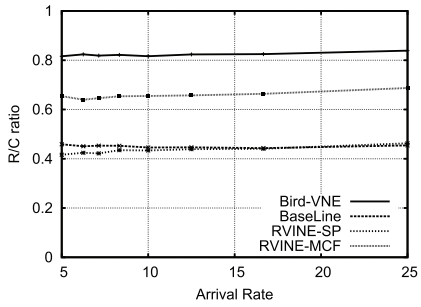

Fig. 7. Revenue to Cost ratio for $\alpha = \beta = \gamma = \alpha' = \beta' = 1$.

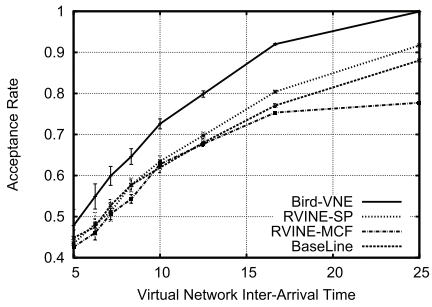

Fig. 8. Acceptance rates of BIRD-VNE under different arrival rates.

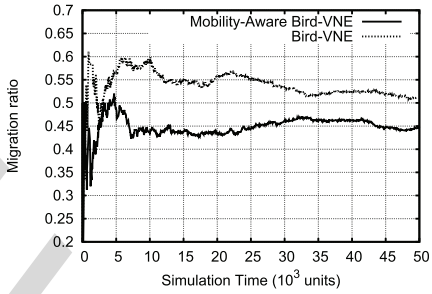

Fig. 9. Mobility-aware Bird-VNE improves the migration ratio.

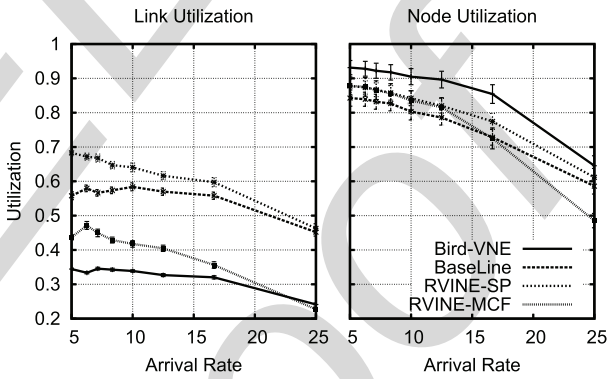

Fig. 10. Average substrate node and link utilization.

is greater than 80% regardless of the arrival rate. This demonstrates the effectiveness of TOPOLOGY-CONSISTENCY and CAPACITY-DISJOINT in pruning the search space by removing the virtual links and nodes that can cause backtracking. Moreover, in large-scale networks where link bandwidth is not a bottleneck, it is possible to ensure a 100% backtrack-free search by ensuring that all link mapping domains are capacity consistent according to Theorem 4.5.

3) BIRD-VNE *Minimizes and Bounds the Average Cost:* The approximation ratio is assessed by comparing the cost achieved by BIRD-VNE to the optimal cost achieved by branch and bound for a substrate network with 30 nodes. As shown in Fig. 4-c, the cost achieved by BIRD-VNE is, on average, only about 5% higher than the optimal cost (i.e., average ratio = about 1.05). But the maximum cost can reach up to 70% higher than the optimal cost (maximum ratio = about 1.7).

4) BIRD-VNE *Results in the Best Revenue to Cost Ratio:* The revenue to cost ratio reflects the average profit of BIRD-VNE, and is 20% better than RVINE-MCF as shown in \figure-name Fig. 7. This is expected for two reasons. First, BIRD-VNE is a $\frac{1}{2}$-approximation of the optimal cost, which contributes to the R/C ratio by minimizing the cost. Second, we have shown numerically that BIRD-VNE has the highest acceptance rate, which directly reflects on the total generated revenue by accepting as many virtual network requests as possible.

5) BIRD-VNE *Link Utilization is Better:* The average link utilization achievable under BIRD-VNE is comparable to that achievable under RVINE-MCF when considering various inter-arrival rates, as shown in Fig. 10. However, for higher loads, the average link utilization of BIRD-VNE is less that that of RVINE-MCF, which confirms our earlier argument stating that BIRD-VNE tends to allocate shorter substrate paths to the virtual links with higher demands. On the other hand, the average node utilization achieved by BIRD-VNE is generally greater than that achieved by RVINE-MCF due to the better acceptance rates.

6) BIRD-VNE *Reduces the Migration Ratio:* Fig. 9 shows the effectiveness of BIRD-VNE in minimizing the migration ratio. In this figure, Mobility-Aware Bird-VNE corresponds to $\gamma(v) = 1$ and Bird-VNE corresponds to $\gamma(v) = 0$. Even when the migration cost is low (i.e., $\gamma(v) = 1$), BIRD-VNE can reduce the migration ratio by at least 10%. This gain can be increased by increasing the migration cost (γ), which is a design trade-off. Observe that because of mobility, about 50% of the accepted virtual networks face migrations.

VII. DISCUSSION AND PRACTICAL CONSIDERATIONS

Several architectural and practical considerations pertain to the discussed virtual network network embedding solution. We discuss some possible approaches to address these challenges.

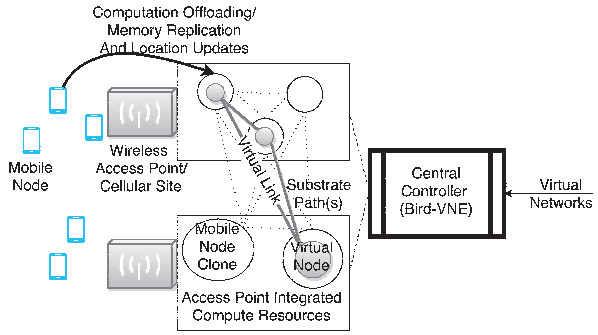

Fig. 11. Architecture: Mobile nodes are connected through wireless infrastructure with integrated compute resources that host clones of mobile nodes and actually implement the requested virtual networks.

Topology changes: Substrate nodes are generally resources limited (e.g. smart-phones) and mobile which results in network topology changes that require updating all the substrate paths computations following any topology change. Updating all paths, **R**, can be addressed architecturally or algorihmically. Fig. 11 shows a possible architecture utilizing the emerging mobile edge computing to address this challenge by augmenting a wireless network infrastructure with distributed cloud resources. Each mobile node replicates its data and states (e.g sensors measurements, locations) to a corresponding clone that is proximate to the node (i.e. at the access point or cellular site). Clones are the actual entities that shall execute the virtual network requests. Cloning the mobile nodes provides several advantages over executing the virtual networks directly on the mobile nodes including: (i) providing manageable and salable processing and link capacity according to virtual networks demands, (ii) facilitating energy conservation of the actual mobile nodes which may be power limited (e.g. sensor nodes), (iii) preventing excessive latency compared to replicating nodes' data in *distant data-centers*, and (iv) preventing substrate network topology changes due to mobility. Unfortunately, the architecture shown in Fig. 11 is not sufficient to prevent updating **R** in some cases such as back-hauling links or node failures or changes in clones deployment. Fortunately updating the set of all paths **R** is not as expensive as computing it from scratch and has remarkable long research history. The authors in [40], for example, study the combinatorial properties of graphs that can be used to update all shortest paths in dynamic networks in $O(n^2 \log^3 n)$ which is *not* a dominant factor in the complexity analysis of our proposed techniques as discussed in Section IV and Section V.

Mobility Model: the general RWP model cannot capture exact mobility patterns especially in walking scenarios. However, the recent modifications of the RWP model in [36] captures the mobility patterns almost exactly at the accuracy of cell level in 3GPP cellular networks which is suitable for several applications such as virtual sensor networks, and virtual content delivery networks. If a finer grain location resolution (e.g. locations of pedestrians at few meters error) were needed by some applications, the RWP model may fail to capture the exact mobility trajectory. In such cases, one can employ other mobility models that characterize smooth movements of mobile nodes (see for e.g. the Semi-Markov Smooth model [41]), or employ model independant trajectory tracking methods (e.g. Kalman filtering) to track nodes locations. Such methods are outside the scope of this paper.

Multipath adoption: If multipath were allowed for mapping virtual links, we conjecture improvements particularly in the acceptance rate [11]. First, multipaths shall allow online path optimization, and traffic splitting for highly demanding virtual links. Second, multipaths shall increase link utilization making the most benefits of the network. Third, multipaths shall facilitate a better sharing of mobile wireless nodes. Finally, multipaths shall allow balancing the substrate network traffic used by the virtual networks and already existing services.

VIII. Conclusion

The coupled constraints in the virtual network embedding problem make it intractable. Instead of over-provisioning the physical network and splitting virtual links across multiple paths, we propose VNE techniques that effectively prune the search space, thereby reducing the execution times by avoiding backtracking, while not compromising the quality of the obtained VNE solutions, expressed in terms of acceptance rates. Our simulations show that the likelihood of performing a backtrack-free search is greater than 80%, confirming the effectiveness of the proposed pruning techniques. These techniques are then exploited to design a polynomial-time, $\frac{1}{2}$-approximation VNE algorithm. We show analytically and empirically that the proposed algorithm outperforms MIP-based algorithms in terms of the revenue to cost ratio and the acceptance rate while minimizing the migration cost arising due to the mobility of physical nodes.

References

[1] S. Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes, "Enabling smart cloud services through remote sensing: An internet of everything enabler," *IEEE Internet Things J.*, vol. 1, no. 3, pp. 276–288, Jun. 2014.

[2] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, 2014, pp. 241–246.

[3] ETSI, "Mobile-edge computing," European Telecommunications Standards Institute (ETSI), Technical White Paper, Sep. 2014.

[4] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Socio-aware virtual network embedding," *IEEE Netw.*, vol. 26, no. 5, pp. 35–43, Sep./Oct. 2012.

[5] K. Zheng *et al.*, "Research and standards: Advanced cloud and virtualization techniques for 5G networks (guest editorial)," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 16–17, Jun. 2015.

[6] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, "Greenhead: Virtual data center embedding across distributed infrastructures," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 36–49, Jan./Jun. 2013.

[7] M. Abu Sharkh, M. Jammal, A. Shami, and A. Ouda, "Resource allocation in a network-based cloud computing environment: Design challenges," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 46–52, Nov. 2013.

[8] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 10–17, Aug. 2001.

[9] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, "Adaptive virtual network provisioning," in *Proc. 2nd ACM SIGCOMM Workshop Virtualized Infrastruct. Syst. Archit.*, 2010, pp. 41–48.

[10] S. Lo, M. Ammar, E. Zegura, and M. Fayed, "Virtual network migration on real infrastructure: A planetlab case study," in *Proc. IFIP Netw. Conf.*, 2014, pp. 1–9.

[11] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008.
[12] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, Nov. 2013.
[13] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. ACM Workshop Virtualized Infrastruct. Syst. Archit.*, 2009, pp. 81–88.
[14] I. Houidi and D. Zeghlache, "Exact adaptive virtual network embedding in cloud environments," in *Proc. IEEE 22nd Int. Workshop Enabling Technol. Infrastruct. Collaborative Enterprises (WETICE)*, 2013, pp. 319–323.
[15] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer, "Energy efficient virtual network embedding," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 756–759, May 2012.
[16] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, 2011.
[17] J. F. Botero, X. Hesselbach, A. Fischer, and H. De Meer, "Optimal mapping of virtual networks with hidden hops," *Telecommun. Syst.*, vol. 51, no. 4, pp. 273–282, 2012.
[18] R. Dechter, *Constraint Processing*. San Mateo, CA, USA: Morgan Kaufmann, 2003.
[19] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *Int. J. Commun. Syst.*, vol. 26, no. 8, pp. 1054–1073, 2013.
[20] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. INFOCOM*, 2006, pp. 1–12.
[21] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.
[22] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 16th Annu. ACM Symp. Theory Comput.*, 1984, pp. 302–311.
[23] M. Till Beck, A. Fischer, H. de Meer, J. F. Botero, and X. Hesselbach, "A distributed, parallel, and generic virtual network embedding framework," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2013, pp. 3471–3475.
[24] J. F. Botero, M. Molina, X. Hesselbach-Serra, and J. R. Amazonas, "A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems," *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1735–1752, 2013.
[25] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Quanitsubstrate," in *Proc. IEEE INFOCOM*, 2014, pp. 1–9.
[26] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 816–827, Mar. 2014.
[27] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Trans. Netw. Serv. Manage.*, vol. 10, no. 4, pp. 356–368, Dec. 2013.
[28] S. Su, Z. Zhang, A. X. Liu, X. Cheng, Y. Wang, and X. Zhao, "Energy-aware virtual network embedding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607–1620, Oct. 2014.
[29] A. Jarray and A. Karmouch, "Cost-efficient mapping for fault-tolerant virtual networks," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 668–681, Mar. 2015.
[30] S. Abdelwahab, B. Hamdaoui, and M. Guizani, "BIRD-VNE: Backtrack-avoidance virtual network embedding in polynomial time," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2014, pp. 4983–4989.
[31] D. Yun and Y. Yi, "Virtual network embedding in wireless multihop networks," in *Proc. 6th Int. Conf. Future Internet Technol.*, 2011, pp. 30–33.
[32] D. Yun, J. Ok, B. Shin, S. Park, and Y. Yi, "Embedding of virtual network requests over static wireless multihop networks," *Comput. Netw.*, vol. 57, no. 5, pp. 1139–1152, 2013.
[33] X. Wang, P. Krishnamurthy, and D. Tipper, "Wireless network virtualization," in *Proc. IEEE Int. Conf. Comput. Netw. Commun. (ICNC)*, 2013, pp. 818–822.
[34] G. Sun *et al.*, "Adaptive provisioning for evolving virtual network request in cloud-based datacenters," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2012, pp. 1617–1622.
[35] D. Arora, A. Feldmann, G. Schaffrath, and S. Schmid, "On the benefit of virtualization: Strategies for flexible server allocation," in *Proc. USENIX Workshop Hot Topics Manage. Internet Cloud Enterprise Netw. Serv. (Hot-ICE)*, 2011, pp. 2–2.
[36] X. Lin, R. Ganti, P. Fleming, and J. Andrews, "Towards understanding the fundamentals of mobility in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1686–1698, Apr. 2013.
[37] J.-C. Régin, "A filtering algorithm for constraints of difference in case csps," in *Proc. 12th Nat. Conf. Artif. Intell. AAAI*, 1994, vol. 94, pp. 362–367.
[38] J. E. Hopcroft and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput.*, vol. 2, no. 4, pp. 225–231, 1973.
[39] A. Fischer, J. F. Botero, M. Duelli, D. Schlosser, X. Hesselbach, and H. De Meer, "ALEVIN-A framework to develop, compare, and analyze virtual network embedding algorithms," *Electron. Commun. EASST*, vol. 37, pp. 1–12, 2011.
[40] C. Demetrescu and G. F. Italiano, "A new approach to dynamic all pairs shortest paths," *J. ACM*, vol. 51, no. 6, pp. 968–992, 2004.
[41] M. Zhao and W. Wang, "A unified mobility model for analysis and simulation of mobile wireless networks," *Wireless Netw.*, vol. 15, no. 3, pp. 365–389, 2009.

Sherif Abdelwahab (S'07–M'11–SM'14) received the B.S. and M.S. degrees in electronics and communications engineering from Cairo University, Giza, Egypt, in 2004 and 2010, respectively. He is currently pursuing the Ph.D. degree at the School of Electrical Engineering and Computer Science (EECS), Oregon State University, Corvallis, OR, USA. Previously, he was with the mobile networks industry with Alcatel-Lucent from 2004 to 2007 and with Etisalat from 2008 to 2013. His research interests include distributed networking, networked systems and services, mobile and wireless networks.

Bechir Hamdaoui (S'02–M'05–SM'12) received the Diploma of Graduate Engineer from the National Engineering School of Tunis, Tunis, Tunisia, the M.S. degrees both in electrical and computer engineering and computer sciences, and the Ph.D. degree in electrical and computer engineering from the University of Wisconsin at Madison, Madison, WI, USA, in 1997, 2002, 2004, and 2005, respectively. He is currently an Associate Professor with the School of EECS, Oregon State University, Corvallis, OR, USA. His research interests include distributed resource management and optimization, parallel computing, co-operative and cognitive networking, cloud computing, and Internet of Things. He is currently an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (2013–present), and the *Wireless Communications and Computing Journal* (2009–present). He also served as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (2009–2014) and *Journal of Computer Systems, Networks, and Communications* (2007–2009). He served as the Chair for the 2011 ACM MobiComs SRC Program, and as the Program Chair/Co-Chair of several IEEE symposia and workshops (including GC 2016, ICC 2014, IWCMC 2009–2015, CTS 2012, PERCOM 2009). He also served on the technical program committees of many IEEE/ACM conferences, including INFOCOM, ICC, GLOBECOM, and PIMRC. He was the recipient of the NSF CAREER Award in 2009.

Mohsen Guizani (S'85–M'89–SM'99–F'09) received the B.S. (with distinction) and M.S. degrees in electrical engineering, the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor and the Chair of the Department of Electrical and Computer Engineering with the University of Idaho, Moscow, ID, USA. Previously, he worked with Western Michigan University, Kalamazoo, MI, USA, University of West Florida, Pensacola, FL, USA, Kuwait University, Kuwait City, Kuwait, and Qatar University, Doha, Qatar. He is the author of eight books and more than 400 publications in refereed journals and conferences. His research interests include computer networks, wireless communications and mobile computing, security, and Internet of Things. He currently serves on the editorial boards of six technical journals. He is a Senior Member of ACM.

Taieb Znati's research interests include the design and analysis of evolvable, secure and resilient network architectures and protocols for wired and wireless communication networks, and the design of new fault-tolerant mechanisms for energy-aware resiliency in data-intensive computing. He is also interested in bio-inspired approaches to address complex computing and communications design issues that arise in large-scale heterogeneous wired and wireless networks. He has served as the General Chair of several main conferences, including GlobeCom 2010, the IEEE INFOCOM 2005, SECON 2004, the first IEEE conference on Sensor and Ad Hoc Communications and Networks, the Annual Simulation Symposium, and the Communication Networks and Distributed Systems Modeling and Simulation Conference. He also served or currently serves as a member of the editorial boards of a number of networking, distributed system and security journals and transactions.

Efficient Virtual Network Embedding With Backtrack Avoidance for Dynamic Wireless Networks

Sherif Abdelwahab, *Senior Member, IEEE*, Bechir Hamdaoui, *Senior Member, IEEE*, Mohsen Guizani, *Fellow, IEEE*, and Taieb Znati

***Abstract*—We develop an efficient virtual network embedding (VNE) algorithm, termed BIRD-VNE, for mobile wireless networks. BIRD-VNE is an approximation algorithm that ensures a close to optimal virtual embedding profit and acceptance rate while minimizing the number of virtual network migrations resulting from the mobility of wireless nodes. BIRD-VNE employs a constraint satisfaction framework by which we analyze the constraint propagation properties of the VNE problem and design constraint processing algorithms that efficiently narrow the solution space and avoid backtracking as much as possible without compromising the solution quality. Our evaluation results show that the likelihood that BIRD-VNE results in backtracking is small, thus demonstrating its effectiveness in reducing the search space. We analytically and empirically verify that BIRD-VNE outperforms existing VNE algorithms with respect to computational efficiency, closeness to optimality, and its ability to avoid potential migrations in mobile wireless networks.**

***Index Terms*—Mobile wireless networks, virtual network embedding, remote sensor networks.**

I. INTRODUCTION

VIRTUAL network embedding in wireless networks can have a pivotal role in several areas including: sensor network virtualization [1], vehicular cloud [2], mobile edge computing [3], [4], [5], network based and geographically distributed cloud environment [6], [7], and cyber foraging [8]. By means of virtualization, it is possible to embed, with low cost, large-scale virtual sensor networks onto sensor-equipped physical devices (e.g. smart-phones, autonomous vehicles) so as to perform specific sensing tasks and autonomous, agile, and timely decisions in a distributed manner. Such virtual networks can support several applications such as: urban sensing, intelligent transportation, terrain exploration, disaster recovery, and surveillance. In addition, VNE can be used to enable virtual content delivery in wireless networks near the network edge. VNE algorithms can then deploy surrogates of services (e.g. networked virtual servers) in proximity to users to improve their perceived latency, where geographical locations and mobility patterns of users are crucial parameters to maintain a target content delivery quality. In a more general context, virtual network embedding in wireless networks can enable effective distributed processing of real-time content and allow agile decision making from data at its "actual sources".

The focus of this paper is on the design of virtual network embedding (VNE) techniques that enable on-demand mapping of virtual networks onto substrate mobile wireless networks. More specifically, the VNE problem consists of mapping the virtual nodes to substrate nodes and the virtual links to substrate paths in such a way that all resource (CPU, storage, and bandwidth) requirements of the virtual network are met. Here, a virtual network consists of a set of virtual nodes, each requiring CPU processing capability and storage capacity to process data in a predefined geographical area, and a set of virtual links connecting these virtual nodes, each requiring some bandwidth capacity. The substrate network, on the other hand, consists of a large set of mobile wireless nodes, each having sensing and Internet-access capabilities.

Unlike wired networks, mobile wireless networks' dynamics (e.g. node mobility, link instability) create new challenges that require new architectural and algorithmic considerations when it comes to enabling VNE. Mobility of substrate nodes, in particular, may invalidate the operations of virtual networks as nodes move away from desired locations of some virtual nodes. Such a mobility can also change the connectivity of the substrate nodes–and so can the substrate paths–that are already used by virtual links, making them insufficient or invalid. In such cases, VNE solutions shall remap (migrate) invalid virtual networks to other substrate nodes and paths [9]. As migrations incur a significant overhead [10], we shall design architectural and algorithmic solutions that can effectively capture node mobility and topology changes, and minimize virtual network migrations due to nodes mobility while not compromising the effectiveness of VNE techniques.

The effectiveness of the VNE techniques can essentially be captured through three metrics: computation time (the time it takes to solve a VNE instance), embedding cost (the amount of overhead incurred and resources needed to solve a VNE instance), and acceptance rate (the ratio of successfully solved VNE instances to the total number of instances). Therefore, in addition to meeting the resource requirements, the aim of VNE techniques is to reduce the computation time, minimize the embedding cost, and increase the acceptance rate. The challenge, however, is that these three performance goals are often

Manuscript received February 18, 2015; revised June 29, 2015 and September 30, 2015; accepted November 23, 2015. This work was supported by the National Science Foundation (NSF) under Grant CNS-1162296. The associate editor coordinating the review of this paper and approving it for publication was W. Wang.

S. Abdelwahab and B. Hamdaoui are with Oregon State University, Corvallis, OR 97331 USA (e-mail: abdelwas@eecs.orst.edu; hamdaoui@eecs.orst.edu).

M. Guizani is with Qatar University, Doha, Qatar (e-mail: mguizani@ieee.org).

T. Znati is with the University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: znati@cs.pitt.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2015.2507134

conflicting with one another. For instance, backtracking algorithms can, in general, find optimal solutions, but they do so in exponential time [11]. Other heuristic approaches, on the other hand, can find solutions in polynomial time, but these solutions are sub-optimal, thus leading to low acceptance rates [12].

In this paper, we develop VNE techniques that strike a good balance between these three performance goals by finding near optimal solutions in polynomial times (short execution times) while yielding high embedding profits (minimal embedding costs) and high acceptance rate. The proposed approach takes also into account potential virtual networks migrations due to substrate nodes mobility in its objective definition to minimize the anticipated overhead associated with migrating invalid virtual networks. Our proposed approach consists of designing algorithms that are based on backtracking techniques so as to ensure good solution optimality, while reducing the computational complexity and the embedding cost by exploiting the constraint propagation properties of the VNE problem. Essentially, they reduce the embedding complexity and cost by narrowing down the search space and avoiding backtracking as much as possible without compromising the solution quality so as to maintain high acceptance rates and minimize potential virtual network migrations. To recap, our contributions in this paper are twofold.

- Developing pruning techniques that reduce the embedding time and cost significantly by reducing the search space. These techniques eliminate the need for backtracking during the embedding solution search, thereby enhancing the embedding time without compromising the optimality of the obtained VNE solutions.
- Developing techniques that account for the VNE embedding cost, expressed in terms of the amount of resources needed and the migration overhead incurred to successfully embed a virtual network, to devise VNE algorithms with minimal embedding costs and minimal potential virtual network migrations.

The rest of the paper is organized as follows. The next section surveys the existing techniques that are related to our proposed VNE approach. In Section III, we state and formulate the VNE problem. We begin by modeling the virtual and substrate networks and the substrate node mobility, and by defining the node and link mapping steps to be performed during the VNE process. We then describe the overall design goals of the VNE technique. In Section IV, we present our pruning techniques proposed to reduce the embedding search space. We then, in Section V, use these pruning techniques to develop a polynomial-time VNE algorithm, which leverages the benefits of our proposed pruning techniques to avoid backtracking while still maintaining the optimality of the obtained VNE solutions. In the same section, we also derive analytic bounds on the approximation ratio of the incurred objective value of the proposed algorithm. Finally, we present our experimental results and findings in Section VI, and conclude the paper in Section VIII.

II. Related Work

Virtual Network Embedding Algorithms: There have recently been research efforts aiming to develop VNE algorithms, and the recent survey by Fischer et al. [12] presents a detailed classification of such algorithms. Broadly speaking, these algorithms can be classified into three categories: backtracking based algorithms (e.g. branch and bound), stochastic algorithms, and heuristics.

Backtracking based algorithms generally consist of formulating and solving the VNE problem using branch and bound or exact backtracking based techniques [13], [14], [15], [16], [17]. For example, Lischka et. al. [13] show that the VNE problem can be formulated as a graph isomorphism (which is known to be *NP-hard*) and then using a backtracking based algorithm to solve it. Backtracking can, in general, find optimal slutions. However, they do so in exponential time [18].

Stochastic algorithms like simulated annealing, particle swarm optimization, tabu search, or genetic algorithms, are other common approaches that can be used to search for VNE solutions. For example, [19] uses particle swarm optimization to find near optimal solutions in relatively short execution times (as shown empirically). The major drawback of stochastic algorithms, besides their relatively long execution times, is their high likelihood of getting stuck in local minima.

Heuristic algorithms attracts the most attention of researchers given their less complexity when compared to exact backtracking algorithms. Heuristics on the other hand can only find inexact solutions and hardly provide tight approximation gaps [20], [21], [22], [23], [24], [25]. For example, Zhu and Ammar in [20] adopt one very basic greedy algorithm that greedily search for feasible nodes to serve a virtual network and then compute the shortest paths between these nodes. If the evaluated shortest paths can satisfy the demands of the virtual links, the virtual network is considered successfully embedded. This is the most simple but sub-optimal algorithm which brings no guarantee to solve the VNE problem. We refer to this algorithm throughout as baseline. The authors in [21] formulate the VNE problem as two stage, coordinated node and link mapping problems, that are both formulated as Mixed ILP (MIP), and then use a rounding relaxation to find near optimal solutions by an off-the-shelf solver. This algorithm can, however, be very slow especially when the size of the virtual network (number of nodes and links) is large, and is shown to have a worst case complexity of $O(n^{14}\, b^2\, \ln b\, \ln\ln b)$ where n is the number of substrate nodes, and b is the number of input bits to the linear program [21], [22]. Several other works adopted a similar approach to [21], formulating the VNE problem as MIP [26], [27]. Formulating the VNE problem as MIP allows a mechanical problem formulation that can address a wide range of objectives such as energy-awareness and fault-tolerance [28], [29], [6], [7]. Heuristic algorithms, though have better execution times than backtracking algorithms, do result in low acceptance rates, due to their sub-optimal embedding nature.

Our algorithm, Bird-VNE, follows a constraint processing design methodology and involves a simplified form of backtracking to bound the resulting approximation-ratio. Our algorithm is different from other backtracking based solutions in that it relies on the analysis of the constraint properties of the VNE problem. This analysis allows us to develop constraint processing algorithms specific to the VNE problem that effectively prune the search space. Unlike other heuristics, Bird-VNE allocates substrate paths directly to the requested

virtual links, rather than separating node and link mapping or at most coordinating their allocations. This approach leads to a proved approximation-ratio that tightens the Bird-VNE performance which was first proposed in our work in [30].

Virtual Network Embedding and Migration in Wireless Networks: Designing VNE algorithms that account for network dynamics (e.g. wireless link quality instability, links failure, node mobility, etc.) attracted little attention [31], [32], [33]. The authors in [33] discuss virtualization measures that can ensure network embedding feasibility in wireless networks under dynamic behaviors. Also in [32], the authors propose to use VNE over *static* wireless multihop networks. Unlike these papers, we design our VNE embedding considering wireless network dynamics due to substrate nodes that can invalidate already embedded virtual networks, hence mandating migrating these virtual networks to ensure service continuity.

Virtual network migration has also attracted the attention of some researchers to fix invalid virtual networks [9], [34], [35]. The work by Houidi et. al [9] is one example in which the authors propose to continuously monitor already embedded virtual networks and to detect possible events that may trigger migration, hence adaptively reembed these virtual networks. Unfortunately virtual network migration is accompanied with several challenges and overheads. A recent study demonstrates the potential migration challenges including: unavoidable packet loss, slow adaptability of switches to changes, and critical deadline time to switch packets to new paths. [10].

In this paper, we extend our work in [30] to take into account the potential virtual network migration overheads by minimizing the likelihood of migrating already embedded virtual networks which arises due to substrate node mobility. Our work also matches the recent recommendations in [10] where an awareness of the potential migrations during the Virtual network embedding phase is needed to avoid the migration drawbacks. Unlike existing virtual network migration algorithms, if we integrate Bird-VNE with a migration solution (e.g. as in [9]), that solution shall become activated less frequently.

III. System Model and Design Objective

We abstract and model the substrate (physical) network, consisting of a set S of n nodes, as an undirected graph $\Phi = (S, L)$ where L is the set of substrate links with each link $l \in L$ corresponding to a connected pair of nodes $s, s' \in S$. We assume that each node $s \in S$ offers a processing capacity $C(s)$, and each link $l \in L$ offers a bandwidth capacity $C(l)$.

In what follows, let $\mathbf{R}$ be the set of all possible paths between all substrate node pairs, where a path $P(s, s')$ between two substrate nodes s and s' is a sequence of connected links (or pairs of nodes) in L. Throughout the paper, $P(s, s')$ (or sometimes P) will also refer to the set of all the links constituting the path. The path length, $|P|$, and the bandwidth capacity, $C(P) = \min_{l \in P} C(l)$, characterize P.

We also consider that the substrate nodes are mobile, and adopt the modified Random Way Point (RWP) mobility model proposed in [36] to model the substrate node mobility. This model describes the mobility of any substrate node s by an infinite sequence of quadruples $\{(\mathbf{X}_{i-1}, \mathbf{X}_i, C_i, W_i)_s\}_{i \in \mathbb{N}}$, where i denotes the i-th movement sample of node s. For every movement sample i, s moves from the starting waypoint $\mathbf{X}_{i-1}$ to the target waypoint $\mathbf{X}_i$ with velocity C_i. Upon arrival to the target waypoint $\mathbf{X}_i$, s waits W_i time units.

Given the waypoint $\mathbf{X}_{i-1}$, the node chooses the target waypoint $\mathbf{X}_i$ randomly such that the included angle θ_i between the vector $\mathbf{X}_i - \mathbf{X}_{i-1}$ and the abscissa is uniformly distributed in $[0, 2\pi]$ and the transition length $Z_i = \|\mathbf{X}_i - \mathbf{X}_{i-1}\|$ is Rayleigh distributed. The angles $\{\theta_1, \theta_2, \ldots\}$ are i.i.d., and the transition lengths $\{Z_1, Z_2, \ldots\}$ of a substrate node s are also i.i.d. with parameter λ_s and a CDF $P(Z_i < z) = 1 - \exp(-\lambda_s \pi z^2)$, $z > 0$.

Velocities C_i are generally i.i.d. random variables with arbitrary distributions. Even with randomly distributed velocities, it is sufficient for the purpose of this paper that $C_i \equiv C_s$, where C_s is a positive constant, equaling the average speed of substrate node s. Waiting times $\{W_1, W_2, \ldots\}$ of a substrate node s are also assumed to be i.i.d. exponential with parameter μ_s and a CDF $P(W_i < w) = 1 - \exp(-\mu_s w)$, $w > 0$

The following are important stochastic properties of the modified RWP [36]:

1) *Transition time* T_r, defined as the time a substrate node spends between two successive waypoints. For a substrate node s moving with constant velocity C_s, the Probability Distribution Function (PDF) of T_r is $f_{T_r}(t) = 2\pi\lambda_s C_s^2 t \exp(-\lambda_s \pi C_s^2 t^2)$ and the (Cumulative Density Function) CDF is $P(T_r < t) = 1 - \exp(-\pi\lambda_s t^2 C^2)$, $\lambda_s > 0$.
2) *Target waypoint distribution.* Given $\mathbf{X}_{i-1}$, the PDF of the target waypoint $\mathbf{X}_i$ in polar coordinates is given by

$$f_{\mathbf{X}_i}(r, \theta) = \lambda_s \exp(-\lambda_s \pi r^2). \quad (1)$$

We also assume that there exists a central node that is responsible for managing the substrate network and embedding the virtual network requests. That is, the central node will be receiving multiple different VNE requests in real time, and embedding them one at time. Each VNE request i is to be embedded for τ_i time units (i.e. τ_i is VNE i's service time).

A. Virtual Network Embedding

A VNE request can be represented as an undirected graph $\Upsilon = (V, E)$ where V is the set of the virtual nodes and E is the set of the virtual links (i.e. connected pairs of virtual nodes). In what follows, let $n_v = |V|$ and $m_v = |E|$. Each node $v \in V$ has a geographical location and a requested node stress $T(v)$ (e.g. processing capacity). Similarly, each virtual link $e \in E$ has a requested link stress $T(e)$ (e.g. link bandwidth). Table I summarizes the key notations.

Suppose that, at a given point in time, the central node has already received and successfully embedded a total of $k-1$ virtual network requests, $\Upsilon^{(1)}, \Upsilon^{(2)}, \ldots, \Upsilon^{(k-1)}$, and the k^{th} request, $\Upsilon^{(k)}$, has just arrived. The problem of embedding of the k^{th} virtual network $\Upsilon^{(k)} = (V^{(k)}, E^{(k)})$ into the substrate network Φ consists of the following two mappings.

Node mapping: maps each virtual node $v \in V^{(k)}$ to a distinct substrate node $s \in S$ subject to two constraints. One, s

TABLE I
SUMMARY OF NOTATIONS

Substrate Network		Random Way Point		Virtual Network	
Symbol	Definition	Symbol	Definition	Symbol	Definition
S	Set of substrate nodes	$\mathbf{X}$	A waypoint	τ_i	Service time
n	Number of nodes	C	Traveling velocity	Υ	Virtual network
Φ	Substrate network	W	Waiting time at $\mathbf{X}$	V	Set of virtual nodes
L	Set of substrate links	Z	Transition lenth	E	Set of virtual links
$C(s)$ and $C(l)$	Substrate capacities	λ	Rayleigh parameter	n_v and m_v	Number of virtual nodes/links
$\mathbf{R}$	Set of all paths	T_r	Transition time	$T(v)$	Processing demand
$P(s,s')$	A substrate path	$f_{\mathbf{X}_i}(r,\theta)$	Waypoint distribution	$T(e)$	Bandwidth demand

must be within Δ distance from v, where Δ is a parameter associated with the VNE request. Two, the sum of the requested processing capacities of all virtual nodes mapped to s (including those mapped from previous VNE requests) must not exceed the offered processing capacity of s. Formally, letting $Dist(u,v)$ denote the Euclidean distance between u and v, node mapping consists of finding a node mapping function, $\mathcal{M}\left(V^{(k)}\right): v \in V^{(k)} \mapsto \mathcal{M}(v) \in S$, such that $\mathcal{M}(v_i) = \mathcal{M}\left(v_j\right)$ iff $v_i = v_j$, $Dist(\mathcal{M}(v), v) \leq \Delta$ for all $v \in V^{(k)}$, and $\sum_{v \in \cup_{i=1}^{k} V^{(i)}: \mathcal{M}(v)=s} T(v) \leq C(s)$ for all $s \in S$.

Link mapping: maps each virtual link $e \in E^{(k)}$ to a substrate path $P \in \mathbf{R}$ subject to two constraints. One, the end virtual nodes of e must correspond to the end substrate nodes of P. Two, for every $l \in L$, the sum of the requested bandwidth capacities of all virtual links (including those belonging to previous VNE requests) whose mapped paths go through the substrate link l must not exceed the offered bandwidth capacity of l. Formally, link mapping consists of finding a link mapping function, $\mathcal{M}\left(E^{(k)}\right): e = (v,v') \in E^{(k)} \mapsto \mathcal{M}(e) = P(s,s') \in \mathbf{R}$, such that $\mathcal{M}(v) = s$, $\mathcal{M}\left(v'\right) = s'$, and $\sum_{e \in \cup_{i=1}^{k} V^{(i)}: l \in \mathcal{M}(e)} T(e) \leq C(l)$ for all $l \in L$.

Definition 3.1: The embedding of $\Upsilon^{(k)}$ is said to be feasible when both the node mapping and link mapping tasks defined above are successful.

Upon successfully embedding the k^{th} VNE request, the central node updates the locations of the substrate nodes, as well as the amounts of the available/remaining substrate resources. These are the remaining processing capacity of substrate node s, denoted by $R^{(k)}(s) = C(s) - \sum_{v \in \cup_{i=1}^{k} V^{(i)}: \mathcal{M}(v)=s} T(v)$, the remaining bandwidth capacity of substrate link l, denoted by $R^{(k)}(l) = C(l) - \sum_{e \in \cup_{i=1}^{k} V^{(i)}: l \in \mathcal{M}(e)} T(e)$, and the remaining path capacity of substrate path P, denoted by $R^{(k)}(P) = \min_{l \in P} R^{(k)}(l)$. Also, upon receiving a new VNE request, the central node constructs the mapping domains of the virtual nodes and links, which are defined as follows.

Definition 3.2: The mapping domain D_v of a virtual node $v \in V^{(k)}$ is defined to be the set of all substrate nodes whose Euclidean distances to v are each less than Δ and whose remaining processing capacities are each greater than $T(v)$; i.e., $D_v = \{s \in S : Dist(s,v) \leq \Delta, R^{(k)}(s) \geq T(v)\}$.

Definition 3.3: The mapping domain D_e of a virtual link $e = (v,v') \in E^{(k)}$ is defined to be the set of all substrate paths whose end nodes (s,s') are in $D_v \times D_{v'}$ and whose remaining capacities are each greater than $T(e)$; i.e., $D_e = \{P(s,s') \in \mathbf{R} : (s,s') \in D_v \times D_{v'}, R^{(k)}(P(s,s')) \geq T(e)\}$.

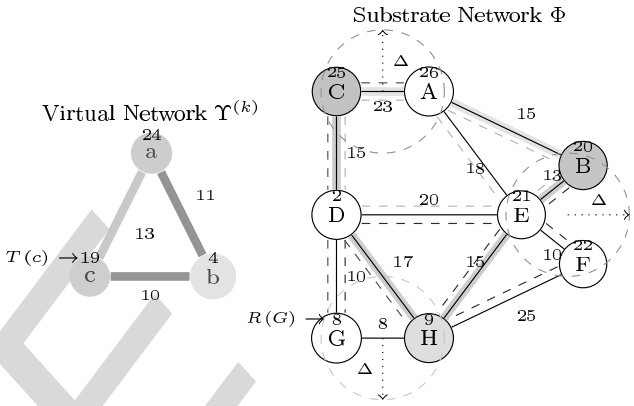

Fig. 1. Virtual Network Embedding: node mapping domains are shown in dashed circles ($radius = \Delta$) and link mapping domains are shown in dashed lines parallel to substrate paths.

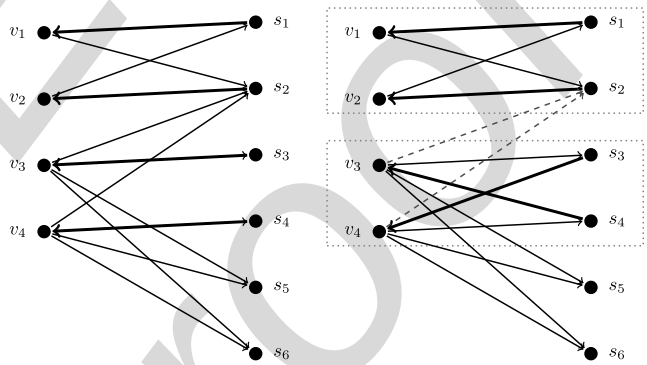

Fig. 2. Procedure 1 illustration. (a):A Maximum cardinality matching (thick edges), v is connected to s if $s \in D_v$, (b):Alternating graph with two strongly connected components. Edges crossing the strongly connected components cannot be in a maximum matching therefore Procedure 1 prunes them.

Figure 1 shows a VNE example, where the graph on the left side is the virtual network and that on the right side is the substrate network. In this example, the node mapping domains are $D_a = \{A, C\}$, $D_b = \{G, H\}$, and $D_c = \{B, E, F\}$, the link mapping domains are shown in dashed lines (e.g. $D_{(a,c)} = \{\{(A,B)\}, \{(A,E)\}, \{(A,E),(E,B)\}, \{(A,C),(C,D),(D,E)\}, \{(A,C),(C,D),(D,E),(E,B)\}, \{(A,B),(B,E)\}\}$). The VNE solution is given by (i) the node mappings, $\mathcal{M}(a) = C$, $\mathcal{M}(b) = H$, and $\mathcal{M}(c) = B$, and (ii) the link mappings, $\mathcal{M}((a,b)) = \{(C,D),(D,H)\}$, $\mathcal{M}((a,c)) = \{(C,A),(A,B)\}$, and $\mathcal{M}((b,c)) = \{(H,E),(E,B)\}$.

B. Probability of VNE Migration due to Node Mobility

If a virtual node v is mapped to a substrate node s, a migration is triggered when the distance $d = Dist(v, s)$ becomes greater than Δ. More specifically, a migration will not be triggered due to s's mobility if s stays within the circle $A(v, \Delta)$ of diameter Δ centered at v for a period longer than τ, the service time of the virtual network request incorporating node v. From (1), the probability that the target waypoint of the substrate node is within $A(v, \Delta)$ is, for $0 \leq d \leq \Delta$,

$$P(A(v,\Delta)) = \int_{\Delta-d}^{\Delta+d}\int_{0}^{2\pi} f_{\mathbf{X}_i}(r,\theta)\, r\,\mathrm{d}r\mathrm{d}\theta,$$
$$= \exp(-\pi\lambda_s(d-\Delta)^2) - \exp(-\pi\lambda_s(d+\Delta)^2).$$

Let $H(s)$ be the probability that a migration is triggered due to the mobility of substrate node s. $H(s)$ can be approximated as the probability that neither the target waypoint is within $A(v, \Delta)$ and the total time spent in $A(v, \Delta)$ is $\geq \tau$ nor the target waypoint is outside $A(v, \Delta)$ and the transition time to the boarder of $A(v, \Delta)$ is $\geq \tau$. Computing the PDF of the total time spent in $A(v, \Delta)$ $(W + T_r)$ requires convolution of the PDFs of W and T_r, and strong assumptions on relative values of λ_s, C_s, and τ, which are outside the control of the embedding algorithm. To simplify the analysis and the VNE objective design, we assume that: i) the time spent within $A(v, \Delta)$ is dominated by the waiting time at the target waypoint $\mathbf{X}_i$, ii) if the target waypoint is outside $A(v, \Delta)$, the whole transition time is spent within $A(v, \Delta)$, and iii) the waiting time of the starting waypoint has elapsed at the time of the virtual network embedding. Since we are mainly interested in evaluating the migration probability of a substrate node relative to other substrate nodes, the impact of these assumptions is minimal. With this, $H(s)$ can be expressed as

$$H(s) = 1 - P(W \geq \tau)P(A(v,\Delta)) - P(T_r \geq \tau)(1 - P(A(v,\Delta))) \quad (2)$$

To minimize the migration overhead, the VNE algorithm shall map virtual nodes to substrate nodes with the least migration probability, $H(s)$. Unlike traditional virtual network embedding and migration algorithms, this requires the estimation of the transition length and waiting time distribution parameters and the use of the estimated parameters to evaluate the migration probability associated with mapping a virtual node v to a substrate node s. The maximum likelihood estimation of the transition length parameter is $\hat{\lambda_s} = \frac{1}{4}(\overline{Z^2})^{-2}$, where the $\overline{Z^2}$ denotes the second sample moment of Z, and that of the waiting time parameter is $\hat{\mu_s} = \frac{1}{\overline{W}}$, where $\overline{W}$ denotes the sample moment of W.

C. VNE Design Objective

Our objective is to develop an algorithm that finds feasible VNEs while maximizing the embedding profit and minimizing the migration overhead. We say that a feasible embedding is optimal when its profit is maximum.[1] Given a virtual network Υ, the profit is defined as the difference between the revenue generated from embedding Υ and its embedding cost, i.e. Profit(Υ) = Revenue (Υ) − Cost (Υ).

To achieve the VNE design objective, we model the embedding cost to capture the cost of node mapping, the cost of link mapping, and the potential cost of migration that may arise as a result of mobility. It is defined as

$$\text{Cost}(\Upsilon) = \sum_{v\in V}\alpha T(v) + \sum_{e\in E}\beta T(e) \times |\mathcal{M}(e)| + \sum_{v\in V}\gamma(v)H(\mathcal{M}(v)), \quad (3)$$

where α and β denote the cost of processing and bandwidth resource units, respectively. The third term captures the cost of migration due to substrate nodes mobility, where $\gamma(v)$ is the cost of migrating the virtual node v. Intuitively, $\gamma(v)$ depends on the amount of resources allocated to v, as well as on v's connectivity to other virtual nodes.

We also define the revenue to be generated from successfully embedding Υ as

$$\text{Revenue}(\Upsilon) = \sum_{v\in V}\alpha' T(v) + \sum_{e\in E}\beta' T(e), \quad (4)$$

where α' and β' denote the price to be charged for each processing and bandwidth unit, respectively.

Observe that the embedding revenue in (4) depends only on the virtual network's requested resources and not on the VNE solution. Also recall that the function $H(\mathcal{M}(v))$ given in (3) represents the probability that a migration of v is triggered due to the mobility of the substrate node, $\mathcal{M}(v)$. It follows that maximizing the profit implies minimizing the embedding cost in (3), which implicitly minimizes the virtual network migration overhead due to mobility. Note that even though, in this paper, the function $H(\mathcal{M}(v))$ captures the likelihood of migration that is due to mobility, it can be used to represent/capture the migration due to any other network dynamics, like link failure.

IV. Enforcing Domain Consistency

The node and link mapping domains, defined in Definitions 3.2 and 3.3, involve coupled constraints. A mapping of a virtual node v to a substrate node $s \in D_v$ impacts other nodes and links mapping domains in several ways. First, no other virtual nodes can be mapped to s. Second, we can only map virtual links that have v as an end node to substrate paths that have s as an end node. Moreover, a mapping of a virtual link e to a substrate path $P \in D_e$ restricts other virtual links from being mapped to the substrate paths that share one or more substrate links with P. The shared links become capacity bottlenecks as their bandwidth capacity must be greater than the required bandwidth of not only e but also other virtual links mapped to paths sharing these links. A backtracking algorithm resolves such constraint couplings by mapping virtual nodes

[1] Modeling the objective as a maximization problem allows us to analytically bound the objective value, as shown later in Section V.

and virtual links one at a time, and backtracking to previous steps when the algorithm encounters an unfeasible mapping.

A VNE algorithm can avoid backtracking (backtrack-free search) if the mapping domains of all virtual nodes and links are consistent. Enforcing domain consistency involves pruning the node and link mapping domains to avoid mappings that lead to an unfeasible embedding. Unfortunately, the use of the standard consistency propagation algorithms are exponential in time. This is because the constraint network of the VNE problem has a maximum degree that is a function of n, while the running time of the standard consistency propagation algorithm, to ensure backtrack-free search, is exponential in the maximum degree of the constraint network (see [18] for details).

Fortunately, constraint propagation algorithms can take advantage of certain properties specific to VNE to prune the mapping domains in polynomial time through mapping domains consistency enforcement. In this section, we develop techniques that exploit these properties to avoid backtracking during the VNE search process, and use these techniques to design a polynomial time, *almost* backtrack-free VNE algorithm. There are two types of mapping domains consistency, virtual network topological consistency and substrate paths capacity consistency, which are presented next.

A. Virtual Network Topological Consistency

We first enforce domain consistency to ensure that the topology of the resulting solution (node and link mappings) matches exactly the topology of the virtual network, i.e. topological consistent. This requires enforcing the following: (i) substrate nodes mapped to the virtual nodes must be *all different*, (ii) end nodes of the substrate paths in link mapping domains must have corresponding substrate nodes in the node mapping domains and vice versa, and (iii) substrate nodes in the node mapping domains must maintain similar virtual node degrees.

Alldifferent virtual node mapping constraint: The constraint to map virtual nodes to distinct substrate nodes is known as the alldifferent constraint in the constraint programming context, and we next state a useful corollary following from Reégin's theorem [37] on the alldifferent constraint.

Corollary 4.1: A virtual node mapping $v \in V \mapsto s \in D_v$ leads to an unfeasible embedding if the edge (v, s) does not belong to a maximum matching that covers all the virtual nodes in the bipartite graph $B = (V \cup S, \{(v, s) : \mathcal{M}(v) = s\})$.

The above corollary can then be exploited to prune away nodes and links from the node and link mapping domains, and for completeness, we provide in Procedure 1 a brief description of such a pruning technique, which we term ALLDIFFERENT [37].

In Procedure 1, a residual graph, B', is defined as $B' = (V \cup S \cup \{t\}, M \cup E_2 \cup E_3 \cup E_4)$ where M is the set of edges in the matching directed from virtual nodes to substrate nodes, E_2 is the set of edges that are not in the matching M and are directed from substrate nodes to virtual nodes, E_3 is the set of all directed edges from substrate nodes in the matching M to a dummy node t, and E_4 is the set of all directed edges from t to substrate nodes that are not in the matching M.

Procedure 1. AllDifferent

Input: $V,\ D_{v \in V}$
Ensure: Distinct virtual node to substrate node mappings in $O\left(n_v^{1.5} n\right)$ [37].
1: Construct bipartite graph $B = (V \cup S, \{(v, s) : \mathcal{M}(v) = s\})$
2: Find a maximum matching M in B using Hopcroft-Karp algorithm [38]
3: **if** $|M| < n_v$ **then**
4: Return no feasible embedding for the given mapping domains
5: **end if**
6: Construct the residual graph B'
7: Compute the strongly connected components in B'
8: Prune the node mapping domains by deleting any edges connecting two different strongly connected components in B'.
9: **return** Narrowed virtual node mapping domains

Step 8 in Procedure 1 prunes substrate nodes from the node mapping domains that can never lead to distinct node mappings. Any edge connecting two different strongly connected components in B' corresponds to a mapping from a virtual node v to a substrate node s and does not belong to any maximum cardinality matching, hence it is not possible to find a feasible embedding with distinct node mapping if v was mapped to s. Thus, s must be removed from D_v. The time complexity of Procedure 1 is bounded by the time required to find the maximum matching using the Hopcroft-carp algorithm in step 2. Since a virtual node can have at most n substrate nodes in its node mapping domain, the number of edges in the bipartite graph B cannot exceed $n_v \times n$ edges. In the worst case, the Hopcroft-carp algotihm requires $O(\sqrt{n_v} n_v n)$ steps, hence the ALLDIFFERENT time complexity is $O\left(n_v^{1.5} n\right)$.

Relational consistency of node and link mapping domains: In the example of Fig. 1, although mapping the virtual node c to F is feasible, doing so prevents us from finding a mapping to the virtual link (a, c), as there is no substrate path between F and any substrate node in the node mapping domain D_a.

From the definition of mapping domains, we can easily observe that if two virtual nodes v, v' are connected by a virtual link e, then the end points of the substrate paths in the virtual link mapping domain D_e is a subset of the cross product of the virtual node mapping domains $D_v \times D_{v'}$. We can now rely on this simple observation and the definition of the virtual link mapping domains to conclude the following:

Lemma 4.2: The node mapping $v \in V \mapsto s \in D_v$ leads to an unfeasible embedding if there exists a link $e = (v, v') \in E$ whose link mapping domain D_e does not contain a path ending at s. Similarly, a virtual link mapping $e = (v, v') \mapsto P(s, s')$ leads to an unfeasible embedding *if* $s \notin D_v$ or $s' \notin D_{v'}$.

Proof: Assume $v \mapsto s$ and a subsequent mapping of $e = (v, v')$ such that there is no path $P \in D_e$ ending at s. A mapping of e to any substrate path in D_e results in mapping multiple virtual nodes to the same substrate node. Also, $e = (v, v') \mapsto$

Procedure 2. Node-Consistency

Input: $E,\ D_{e\in E},\ D_{v\in V}$
Ensure: Virtual node mapping domains are consistent with virtual link mapping domains in $O(m_v n)$
1: **for all** virtual link $e = (v, v') \in E$ **do**
2: $D_v \leftarrow D_v \cap u(D_e)$
3: $D_{v'} \leftarrow D_{v'} \cap v(D_e)$
4: **end for**
5: **return** Narrowed virtual node mapping domains

Procedure 3. Link-Consistency

Input: $E,\ D_{e\in E},\ D_{v\in V}$
Ensure: Virtual link mapping domains are consistent with virtual node mapping domains in $O(m_v n^2)$
1: **for all** virtual link $e = (v, v') \in E$ **do**
2: **for all** substrate path $P \in D_e$ **do**
3: if $u(P) \notin D_v \vee v(P) \notin D_{v'}$ **then**
4: $D_e \leftarrow D_e \setminus \{P\}$
5: **end if**
6: **end for**
7: **end for**
8: **return** Narrowed virtual link mapping domains

$P(s, s')$ violates the link mapping Definition 3.3 if either $s \notin D_v$ or $s' \notin D_{v'}$. ■

Using Lemma 4.2, we propose two procedures to narrow down the node and link mapping domains: Procedures 2 and 3. The functions $u(D_e)$ and $v(D_e)$ return the sets respectively of the first and the second end nodes of all the paths in D_e. When applied to a path P, $u(P)$ and $v(P)$ return the path's first and second end nodes. In each iteration, Procedure 2 prunes the substrate nodes from the node mapping domains of the end nodes of the virtual links, if there is not any substrate path in their link mapping domains that also ends at those substrate nodes. Since for each virtual link the intersection operator (step 2 and 3) requires at most $O(n)$ steps as $|D_v| \leq n$, then Procedure 2 has a worst case time complexity of $O(m_v n)$.

Procedure 3 complements Procedure 2 by pruning a substrate path from the link domain of a virtual link if the substrate nodes ending that path cannot be found in the node mapping domains of the virtual nodes ending the virtual link. Since there are at most $O(n^2)$ paths in the substrate network, the inner loop (step 2 to 6) of Procedure 3 requires at most $O(n^2)$ steps. Hence, the worst case time complexity of Procedure 3 is $O(m_v n^2)$.

Consistency of virtual and substrate node connectivity: The relational consistency of node and link mapping domains does not ensure connectivity of the virtual network, nor does it imply that the mapping domains can satisfy the virtual network connectivity requirements, especially when the node mapping domains overlap. To illustrate this, consider a new induced network of substrate nodes that represents the connectivity of the virtual link domains. In this induced network, substrate nodes are connected by an edge if there exists a path belonging to any link mapping domain that connects them. Induced network is defined formally next.

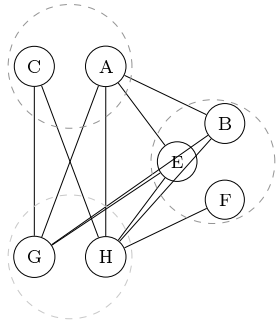

Fig. 3. Induced network I from substrate network Φ in Fig. 1. I has one connected components CC_I and three supernodes (dashed circles). $\zeta(CC_I) = 3$, $\delta(F) = 1$ and equals 2 for all other nodes.

Definition 4.1: Given a virtual network Υ, we define the induced network I of Υ as the undirected graph $I = (S_I \subset S, L_I)$ where $S_I = \cup_{v\in V} D_v$ and $L_I = \{(s, s') \in S_I^2 : \exists\ P(s, s') \in D_e$ for some $e \in E\}$.

Definition 4.2: For every connected component CC_I of I, the set $N_v(CC_I) = CC_I \cap D_v$ corresponding to the virtual node v is called the supernode of v. Let $\zeta(CC_I)$ be the number of distinct supernodes in CC_I. For every $s \in S_I$, we define $\delta(s)$ as the number of supernodes connected to s.

Fig. 3 illustrates the induced network of the example given in Fig. 1. This induced network is constructed by connecting a pair of substrate nodes in Fig. 3 when there is at least one path connecting them in any link mapping domain. In general, if the mapping $v \mapsto s$ is feasible, the function $\delta(s)$ reflects the degree of the virtual node v, and if a connected component CC_Υ of Υ is mapped to a subset of substrate nodes in Φ, the function $\zeta(CC_I)$ reflects the number of virtual nodes in the connected component CC_Υ (size of CC_Υ).

Lemma 4.3: Let $Deg_\Upsilon(v)$ denote the degree of virtual node v. A virtual node mapping $v \mapsto s$ leads to an unfeasible embedding *if* $Deg_\Upsilon(v) > \delta(s)$ or the size of the connected component of Υ (CC_Υ) that contains v is greater than the number of supernodes in CC_I that contains s.

Proof: Assume $v \mapsto s$ and $Deg_\Upsilon(v) > \delta(s)$, then there exist at least one virtual link e such that there is no substrate path P in D_e with one of its end substrate nodes equals s. Then, $v \mapsto s$ does not lead to a feasible embedding from Lemma 4.2. If $Deg_\Upsilon(v) \leq \delta(s)$ but $|CC_\Upsilon| > \zeta(CC_I)$, then there must exist an unmapped virtual node $v' \in CC_\Upsilon$, while all substrate nodes $s \in CC_I$ are already mapped to other virtual nodes in CC_Υ including v. Since v' must be mapped to one substrate node in CC_I to maintain connectivity, then mapping $v \mapsto s$ does not lead to a feasible embedding. ■

The DEGREE-CONSISTENCY procedure (Procedure 4), a direct application of Lemma 4.3, is a pruning technique that narrows down mapping domains through degree consistency enforcement. Its complexity is bounded by computing $\delta(s)$ for all the substrate nodes in the virtual node mapping domains, which is $O(n^2)$.

Running the ALLDIFFERENT, NODE-CONSISTENCY, LINK-CONSISTENCY, and DEGREE-CONSISTENCY procedures for one iteration removes some inconsistent mappings from the node and link mapping domains. To remove all

Procedure 4. Degree-Consistency

Input: $E,\ D_{v\in V}$
Ensure: Degree Consistency in $O(n^2)$
1: **for all** virtual nodes $v' \in V$ **do**
2: **for all** substrate nodes $s' \in D_{v'}$ **do**
3: **if** $Deg_{\Upsilon}(v') > \delta(s')$ **then**
4: $D_{v'} \leftarrow D_{v'} \setminus \{s'\}$
5: **end if**
6: **end for**
7: **end for**
8: **for all** connected component $CC_{\Upsilon} \in \Upsilon$ **do**
9: **for all** connected component $CC_I \in I$ **do**
10: **if** $|CC_{\Upsilon}| > \zeta(CC_I)$ **then**
11: $D_{v'} \leftarrow D_{v'} \setminus CC_I,\ \forall v' \in CC_{\Upsilon}$
12: **end if**
13: **end for**
14: **end for**
15: **return** Narrowed virtual nodes domains

Algorithm 1. Topology-Consistency

Input: $E,\ D_{e\in E},\ D_{v\in V}$
Ensure: Topology Consistency in $O(m_v n^3)$
1: **repeat**
2: NODE-CONSISTENCY($E,\ D_{e\in E},\ D_{v\in V}$)
3: ALL DIFFERENT($V,\ D_{v\in V}$)
4: LINK-CONSISTENCY($E,\ D_{e\in E},\ D_{v\in V}$)
5: DEGREE-CONSISTENCY($E,\ D_{v\in V}$)
6: **if** $\exists\ D_{v'} = \emptyset,\ \forall v' \in V\ \vee\ D_e = \emptyset,\ \forall e \in E$ **then**
7: **return** false {T}here exist a virtual node or a virtual link with an empty mapping domain.
8: **end if**
9: **until** No node or link domain is changed
10: **return true**

the inconsistencies, these procedures must repeatedly be run sequentially until no further removal is possible from either the node or the link mapping domains. The process merging all these four procedures is captured in Algorithm 1, which essentially removes inconsistency, and hence avoids backtracking, by ensuring topological consistency of the node and link mapping domains. This algorithm is referred to as TOPOLOGY-CONSISTENCY.

The complexity of TOPOLOGY-CONSISTENCY is bounded by the number of times we run the procedure LINK-CONSISTENCY in step 4. This implies a complexity of $O(m_v n^2)$ in each iteration. In the worst-case scenario, TOPOLOGY-CONSISTENCY removes one substrate node from one node mapping domain and this corresponds to at least one removal of one substrate path from link mapping domains. Hence, it requires at most n iterations to remove all the substrate nodes from one node mapping domain, thus returning false.[2] Thus, the complexity of TOPOLOGY-CONSISTENCY is $O(m_v n^3)$. But since the maximum number of virtual nodes is the number of substrate nodes; i.e., $n_v \leq n$, then the complexity of TOPOLOGY-CONSISTENCY is $O(n^5)$.

[2] A more efficient implementation checks the condition in step 6 every time any procedure removes a substrate node/link from a mapping domain.

B. Capacity Disjoint Paths Consistency

We now study the second mapping domain consistency type, substrate paths capacity consistency. Let us refer again to the example given in Fig. 1 and consider the link mapping sequence $(a, b) \mapsto P(C, H) = \{(C, D), (D, H)\}$ and $(a, c) \mapsto P(C, E) = \{(C, D), (D, E)\}$. The remaining bandwidth of the substrate link (C, D), $R((C, D)) = 15$, is less than the sum of the links' requested bandwidth capacities, which is $T((a, b)) + T((a, c)) = 24$. Hence, this mapping sequence is unfeasible. Clearly, a VNE algorithm will not backtrack if all substrate paths in the link mapping domains are disjoint (if topological consistency is enforced). However, constructing the link mapping domains from disjoint paths results in a degradation of the VNE acceptance rate (such a rate reflects the number of virtual networks that can be embedded into the substrate network), as well as in an increase in the embedding cost. Our proposed embedding algorithm does not force paths to be disjoint so as to increase the acceptance rate and decrease the embedding cost. Instead, our technique relies on the concept of *capacity disjoint* which we formally define next.

Definition 4.3: For every substrate link l, let $\bar{D}_e(l) = \{P \in D_e : P \ni l\}$ and $\bar{E}(l) = \{e \in E : \bar{D}_e(l) \neq \emptyset\}$. We say that the paths in $\mathbf{R}' = \bigcup_{e\in\bar{E}(l)} \bar{D}_e(l)$ are *capacity disjoint iff* the remaining bandwidth capacity of l is greater than the sum of the requested bandwidth capacities of all the virtual links in $\bar{E}(l)$. Formally, the paths in $\mathbf{R}'$ are said to be capacity disjoint *iff* $R^{(k)}(l) \geq \sum_{e\in\bar{E}(l)} T(e)$.

Lemma 4.4: A virtual link mapping $e \mapsto P$ leads to an unfeasible embedding *if* all the substrate paths in every unmapped virtual link's mapping domain are not capacity disjoint with P.

Proof: If a virtual link $e_i \mapsto P_i$ and in a next mapping step of virtual link e_j, all paths in D_{e_j} are not capacity disjoint with P_i, then any mapping $e_j \mapsto P_j \in D_{e_j}$ will result in at least one substrate link with negative remaining bandwidth. ■

Theorem 4.5: The proposed TOPOLOGY-CONSISTENCY algorithm ensures a backtrack-free search if all substrate paths in all link mapping domains are capacity disjoint.

Proof: It follows from Lemmas 4.2, 4.3, and 4.4 and from Corollary 4.1. ■

An algorithm that aims to ensure a backtrack-free search may remove substrate paths that are not capacity disjoint from the virtual links mapping domains. Although such an algorithm will have a complexity advantage because it is backtrack-free, it degrades the acceptance rate and the cost as it will remove substrate paths that can actually lead to feasible or minimum cost embedding. Apparently, capacity disjoint paths condition is required only for substrate paths that are actually in an incurred embedding. In order to overcome the complexity problem while still minimizing the cost and maximizing the acceptance rate, we propose Algorithm 2 (CAPACITY-DISJOINT),

Algorithm 2. Capacity-Disjoint

Input: $E,\ L,\ D_{e\in E},\ D_{v\in V}$
Ensure: Substrate paths are capacity disjoint if they are likely to coexist in an incurred embedding in $O(m\, m_v\, n^3)$.
1: **for all** $l \in L:\ \exists\ ps \in D_{e\in E},\ l \in P$ **do**
2: repeat
3: NODE-CONSISTENCY($\bar{E}(l),\ \bar{D}_e \in \bar{E}(l),\ \bar{D}_v \in \bar{V}(l)$)
4: ALL DIFFERENT($\bar{V}(l),\ \bar{D}_v \in \bar{V}(l)$)
5: LINK-CONSISTENCY($\bar{E}(l),\ \bar{D}_e \in \bar{E}(l),\ \bar{D}_v \in \bar{V}(l)$)
6: **until** No node or link sub-domain is changed
7: $R'(l) \leftarrow R(l)$
8: **for all** $e \in \bar{E}(l)$ ordered ascendingly by $|D_e|$ **do**
9: **if** $\bar{D}_e(l) \neq \emptyset$ **then**
10: $R'(l) \leftarrow R'(l) - T(e)$
11: **if** $R'(l) < 0$ **then**
12: $D_e \leftarrow D_e \setminus \bar{D}_e(l)$
13: **end if**
14: **end if**
15: **end for**
16: **end for**
17: **if** $\exists\ D_e = \emptyset,\ \forall e \in E$ **then**
18: **return false**
19: **end if**
20: **return true**

which ensures that substrate paths in link mapping domains are capacity disjoint if they are likely to coexist in an incurred embedding.

The key idea of the CAPACITY-DISJOINT algorithm is to determine the worst case scenario in which the intersecting substrate paths in $\mathbf{R}'$ can become simultaneous mappings of virtual links in $\bar{E}(l)$. These paths are found by applying topological consistency procedures, discussed earlier, on the subsets of link and node mapping domains $\bar{D}_e \in \bar{E}(l),\ \bar{D}_v \in \bar{V}(l)$ (Steps 1 to 6), where $\bar{V}(l) \subset V$ is the set of end virtual nodes of virtual edges in $\bar{E}(l)$ and $\bar{D}_v(l) \subset D_v$ is the set of substrate node mappings deduced from $\mathbf{R}'$.

The CAPACITY-DISJOINT algorithm checks if all paths that are common to every substrate link l are capacity disjoint. If not, the algorithm removes first the substrate paths $\bar{D}_e \in \bar{E}(l)$ from the domain of the virtual link(s) e that has the largest link mapping domain size $|D_e|$ (Step 7 to 16). This is to minimize the chances of ending up with an empty link mapping domain, thus maximizing the acceptance rate. Although it is clear that CAPACITY-DISJOINT algorithm does not eliminate backtracking entirely, it substantially reduces its likelihood of occurrence. We evaluate the likelihood of backtracking empirically in Section VI.

The CAPACITY-DISJOINT algorithm uses similar steps to determine possible simultaneous intersecting paths (Steps 2 to 6) for each substrate link l that intersects with some paths. Although these steps are performed on a subset of the mapping domains and it is unlikely to encounter the situation that every substrate link is a common link for all paths (as the substrate network will almost look like a path), the complexity of CAPACITY-DISJOINT is bounded by $O(m\, m_v\, n^3)$. This can be expressed as $O(n^7)$ if both the substrate and virtual networks are complete graphs and have the same number of nodes n.

V. APPROXIMATE PROFIT MAXIMIZATION

TOPOLOGY-CONSISTENCY and CAPACITY-DISJOINT algorithms, discussed in the previous section, reduce the search space and improve the running time of backtracking search. However, even in the case of backtrack-free search, an optimal optimization algorithm, like branch and bound, may still traverse the whole search space through brute-force [18]. If we assume that the VNE problem is backtrack-free, it can be viewed as the maximum weight matching problem in a bipartite graph. The bipartite graph in this case is the set of virtual links on one side of the bipartite graph connected by weighted edges to the set of substrate paths on the other side and the edge weight is the profit attained by mapping a virtual edge to a substrate path. From (3) and (4), the profit of mapping a virtual edge $e = (v, v')$ to a substrate path $P = (s, s')$ is given by

$$\begin{aligned}\text{Profit}(e, P) =&(\alpha' - \alpha)(T(v) + T(v')) \\ &+ (\beta' - \beta \times |P|)T(e) \\ &- \gamma(v)H(s) - \gamma(v')H(s').\end{aligned}$$

However, a direct application of conventional maximum weight matching algorithms (e.g. Hungarian methods or Edmond's methods) is non-trivial. Fortunately, greedy approximations to the maximum weight matching are applicable, but with some needed modifications to enforce domain consistency and to verify solution feasibility in every step. We use this observation to propose Algorithm 3, which finds a VNE such that the incurred embedding profit is at most as half as the optimal profit in an attainable special case and at least $\frac{1}{n_v}$ in general.

Our proposed VNE algorithm, termed BIRD-VNE, starts by enforcing the mapping domains consistency using TOPOLOGY-CONSISTENCY and CAPACITY-DISJOINT. It then searches for an embedding by mapping the virtual links with the smallest link mapping domain sizes and greatest demands (Line 9) first to the substrate paths in their domains with the highest profit (Line 10). After mapping each virtual link (mapping step), the algorithm ensures feasible embedding according to Definition 3.1 (Line 20). If any mapping step results in unfeasible embedding, the algorithm starts over the mapping process from the first virtual link by assigning it to an unattempted mapping in its domain until a feasible embedding is found or all mappings of the first link are tried.

This algorithm still involves a simplified form of backtracking. The algorithm always backtracks to the first virtual link mapping step. In this case, the total number of backtracks is bounded in the worst case by the size of the smallest link mapping domain. Typically, the consistency enforcing algorithms reduce the number of backtracks significantly as we will show empirically in Section VI. The following theorem bounds the worst case performance of BIRD-VNE.

Theorem 5.1: In the worst case, BIRD-VNE is an $O(\frac{1}{n_v})$-approximation to the optimal embedding profit, and

Algorithm 3. BIRD-VNE

Input: $\Upsilon = (V, E)$, $\Phi = (S, L)$ **Input:**
Require: $D_{\forall e \in E}$, $D_{\forall v \in V}$
Ensure: Embedding $\Upsilon \mapsto \Phi$ in $O(m\, m_v\, n^3)$
1: *SolutionExist* ← TOPOLOGY-CONSISTENCY
2: *SolutionExist* ← *SolutionExist* **and** CAPACITY-DISJOINT
3: *SolutionExist* ← *SolutionExist* **and** TOPOLOGY-CONSISTENCY
4: **if not** *SolutionExist* **then**
5: **return** "Reject virtual network."
6: **end if**
7: **repeat**
8: $\mathcal{M}(e) \leftarrow \emptyset,\ \forall e \in E$
9: **for all** $e = (v, v') \in E$ ordered ascendingly by $|D_e|$, and by $T(e)$ **do**
10: **for all** $P = (s, s') \in D_e$ ordered descendingly by Profit(e, P) **do**
11: **if** e is the first virtual link in the order of E **then**
12: $D_e \leftarrow D_e \setminus P$
13: **end if**
14: **if** $e \mapsto P$ result in a feasible embedding **then**
15: $\mathcal{M}(e) \leftarrow P, \mathcal{M}(v) \leftarrow u(P), \mathcal{M}(v') \leftarrow v(P)$
16: **break**
17: **end if**
18: **end for**
19: **end for**
20: **until** Feasible embedding is found or all first virtual link mapping domain are attempted.
21: **if** No feasible embedding is found **then**
22: **return** "Reject virtual network."
23: **end if**
24: **return** $\mathcal{M}(e),\ \forall e \in E$ and $\mathcal{M}(v),\ \forall v \in V$

only a $\frac{1}{2}$-approximation if there are, on average, n_v paths of the same length between any two substrate nodes.

Proof: Let x be the profit of mapping the first virtual link e to the highest profitable path P in its link mapping domain in a single iteration (step 7). The following potential mappings become invalid and will never be attempted by the algorithm until a backtracking to step 7 is decided: (i) mapping e to other substrate paths in its link mapping domain except P, (ii) mapping any other virtual link to P, (iii) mapping another virtual link e' that shares one of its end virtual nodes with e with any other substrate paths except those that also share the same end substrate node with P. Let d be the maximum degree of the virtual network, the worst case will occur if we have exactly d paths of shortest length (highest profit) and the algorithm invalidates at most d mappings of the optimal mappings (at most in the first mapping). In this case, the sum of profits of the invalidated mapping cannot exceed dx. Since the profit is non-negative, the approximation ratio is $O(\frac{1}{d})$ or more conservatively $O(\frac{1}{n_v})$. However, if there are n_v redundant substrate paths of the same length between any two substrate nodes, BIRD-VNE invalidates at most two mappings that may be optimal. This can be repeated for at most $\frac{1}{2}m_v$ of the steps (9 to 19) and the sum of the profits of the invalidated mappings cannot exceed $2x$. In this later case, the approximation ratio is $\frac{1}{2}$, which proves the theorem. ■

1) Scalability and Implementation Consideration: The complexity of BIRD-VNE is analyzed as follows. The main loop (Step 10 to 25) has m_v iterations. In the worst-case scenario, for every virtual link, it checks the feasible mappings of n^2 paths. Then, the complexity of BIRD-VNE is bounded by the CAPACITY-DISJOINT complexity $O(m\, m_v\, n^3)$ and can be written as $O(n^7)$. Although BIRD-VNE is polynomial in time and scales much better than the state-of-the art algorithms, its $O(n^7)$ complexity may prevent applying it to very large scale networks. Fortunately, this complexity bound can be improved through simple but effective implementation improvements.

The two procedures, NODE-CONSISTENCY and LINK-CONSISTENCY, can be easily implemented in parallel by implementing these algorithms on exactly m_v processing agents. In this case, the NODE-CONSISTENCY complexity is reduced to $O(n)$ while the LINK-CONSISTENCY complexity is reduced to $O(n^2)$. It then follows that the complexity of TOPOLOGY-CONSISTENCY is bounded by running ALLDIFFERENT at most n times, hence it is $O\left(n_v^{2.5} n\right)$. Similarly, the CAPACITY-DISJOINT complexity is also bounded by running ALLDIFFERENT at most m times, hence it is $O\left(n_v^{2.5} m\right)$. The complexity of CAPACITY-DISJOINT bounds the overall complexity of BIRD-VNE to $O\left(n_v^{2.5} m\right)$.

We can also improve the actual approximation ratio in practice by repeating step 7 to 19 until all virtual link mappings of the first virtual link are attempted (i.e. remove steps 14 to 17) while maintaining all the feasible solutions. We then pick the solution with the maximum total profit as our solution and the other solutions as backup solutions in case a migration is needed. This trick reduces the gap between the evaluated total profit and the optimal solution when compared to the worst case scenario, and preserves the same worst case complexity at the expense of the actual execution time.

2) Virtual Network Migration Consideration: The proposed algorithm, BIRD-VNE, can still be used, with simple modification, if virtual network migration is needed. If the previously discussed implementation in Section V-1 is adopted, we end up with multiple solutions to the same virtual network that can be quickly evaluated for feasibility, so as to choose one of these VNE solutions for migrating the virtual network instead of evaluating BIRD-VNE again from the beginning. Moreover, the following simple procedures can be carried out to perform migrations to individual nodes and links instead of migrating the whole virtual network.

Consider the case that a substrate path P is not capable of meeting the required demand of a virtual link e. This situation can happen, for example, in case of a link failure or congestion along the path, or failure of one or both end substrate nodes of P. In this case, we can immediately find another backup path (and substrate nodes if necessary), P', in $D_e \{P\}$ that has the largest profit and is also feasible with the current embedding $\mathcal{M}(e'),\ \forall e' \in E \{e\}$. This algorithm is as simple as running the steps from 9 to 19 in BIRD-VNE, while replacing D_e with $D_e \{P\}$ for only the virtual links that are impacted by the failure

of P. If this fails, the whole embedding needs to be performed again by running BIRD-VNE.

VI. NUMERICAL RESULTS

The effectiveness of the proposed algorithm, BIRD-VNE, is assessed in terms of the metrics suggested in [39]:

1) *Acceptance rate,* defined as the ratio of the total accepted virtual networks to the total requested virtual networks.
2) *Revenue to Cost ratio (R/C),* defined as $R/C = \sum_{\Upsilon} \text{Revenue}(\Upsilon) / \sum_{\Upsilon} \text{Cost}(\Upsilon)$.
3) *Average node and link utilization,* defined as $\sum_{s \in S} \frac{R(s)-C(s)}{nC(s)}$ and $\sum_{l \in L} \frac{R(l)-C(l)}{mC(l)}$, respectively.

In addition, we use the following metrics to assess the effectiveness of BIRD-VNE vis-a-vis of its ability to avoid backtracking, limit network migration, and achieve optimal VNE by comparing it to the optimal Brand and Bound technique.

1) *Average/Maximum Approximation ratio,* defined as the average/maximum ratio of the cost achieved by BIRD-VNE to that achieved by Branch and Bound.
2) *Backtrack-free ratio,* defined as the ratio of the total number of times in which BIRD-VNE finds a feasible embedding at the first attempt of the first virtual link mapping to the total number of accepted requests.
3) *Migration ratio,* defined as the ratio of the total number of virtual network migrations to the total number of accepted requests.

A. Simulation Setup

We compare the performance of BIRD-VNE with two existing algorithms, Randomized Virtual Network Embedding with shortest path link mapping (RVINE-SP) and with multicommodity flow link mapping (RVINE-MCF) [21], which are integrated to an event-driven simulator that we developed.[3] We also compare the performance achievable under BIRD-VNE to that achievable under the basic Greedy algorithm, referred to as BASELINE and proposed in [20].

The simulator generates Φ and Υ according to Erdös−Rènyi model. Similar to [21], Φ has 0.5 probability of connecting any two substrate nodes, $n = 50$, $C(s) \sim U(0, 50), \forall s \in S$ and $C(l) \sim U(0, 50), \forall l \in L$. Substrate nodes are placed randomly on a (25×25) grid. The mean inter-arrival time of virtual networks ranges from 5 to 25 networks per time unit, and the average service time is set to $\tau = 1000$ time units. Each pair of virtual nodes in Υ is connected with 0.5 probability, $n_v \sim U(1, 10)$, $\Delta = 15$, $T(v) \sim U(0, 20), \forall v \in V$ and $T(e) \sim U(1, 50), \forall e \in E$. The routing of the substrate network $\mathbf{R}$ is computed once in the prepossessing initialization step using the all shortest path algorithm. All the cost parameters α, β, γ are set to unity in the simulations.

We simulate the mobility of substrate nodes by setting $\tau = 50$, and the average waiting time at each waypoint to $\mu_s^{-1} = 100$ time units for all the substrate nodes. All substrate nodes travel with the same constant speed $C_i = 5$ speed units, and the average transition length of all the nodes is 5 length units (i.e. $\frac{\lambda_s^2}{2} = 5$). We consider a wireless network infrastructure in which the connectivity between the substrate nodes are not impacted by their mobility since fixed clones of the mobile nodes actually execute the virtual network requests in a geographically distributed cloud infrastructure as discussed in Section VII and as illustrated in Fig. 11.

[3]Implementations of RVICE-SP and RVINE-MCF are online available at http://www.mosharaf.com/ViNE-Yard.tar.gz

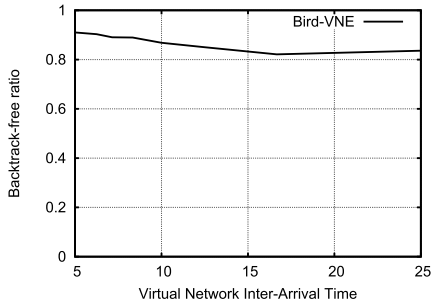

Fig. 4. Backtrack-free ratio of BIRD-VNE shows the effectiveness of search space pruning.

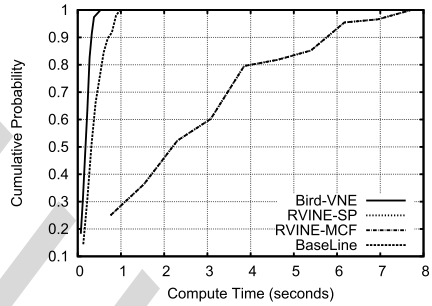

Fig. 5. Experimental computation time CDF of BIRD-VNE shows its computation effectiveness.

B. Performance Evaluation

1) BIRD-VNE *Improves the Acceptance Rate:* Fig. 8 shows that BIRD-VNE has a 15% better acceptance rate when compared to the other algorithms. The improvement in the acceptance rate is a direct result of Theorem 4.5 and is consistent for different loads. BIRD-VNE is likely to find a feasible embedding once it passes the consistency enforcement steps 1 to 6.

On the other hand, RVINE-SP and RVINE-MCF first rely on LP relaxations to solve the non-convex MIP problem, and then round the solution of the relaxation to the nearest integer. This way, RVINE-SP and RVINE-MCF may unnecessarily reject a VNE request by falsely concluding that it cannot be embedded. Moreover, when there is no solution, RVINE-SP and RVINE-MCF tend to spend a significant amount of time searching for solutions before eventually rejecting unfeasible requests as shown in Fig. 4-b.

2) BIRD-VNE *Avoids Backtracking:* Fig. 4-a shows the backtrack-free ratio of BIRD-VNE. BIRD-VNE is unlikely to encounter a backtracking, and finds a feasible solution from the first attempt. In this simulation setup, the backtrack-free ratio

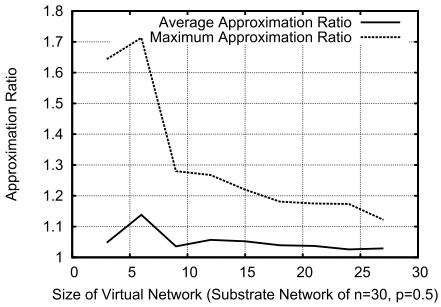

Fig. 6. BIRD-VNE Maximum and average approximation Ratio (optimal is branch and bound).

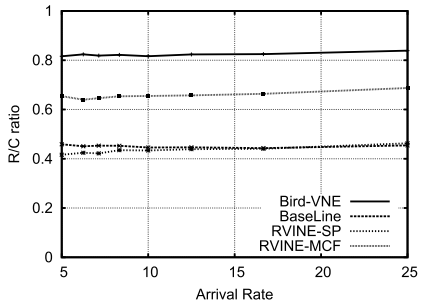

Fig. 7. Revenue to Cost ratio for $\alpha = \beta = \gamma = \alpha' = \beta' = 1$.

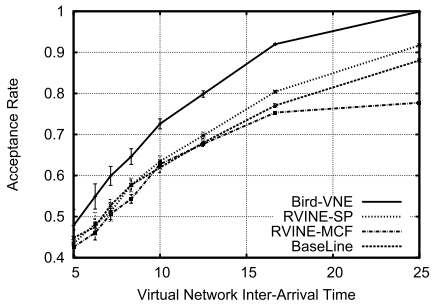

Fig. 8. Acceptance rates of BIRD-VNE under different arrival rates.

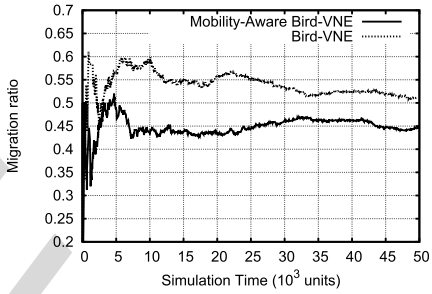

Fig. 9. Mobility-aware Bird-VNE improves the migration ratio.

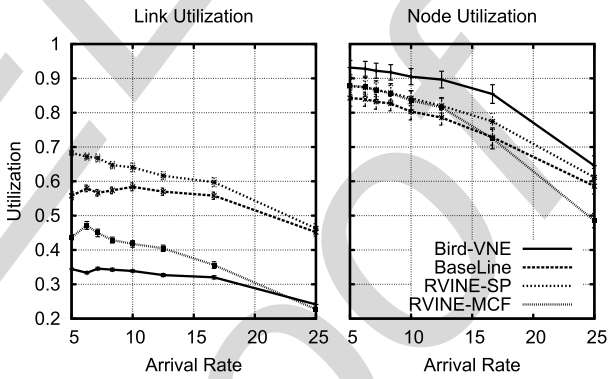

Fig. 10. Average substrate node and link utilization.

is greater than 80% regardless of the arrival rate. This demonstrates the effectiveness of TOPOLOGY-CONSISTENCY and CAPACITY-DISJOINT in pruning the search space by removing the virtual links and nodes that can cause backtracking. Moreover, in large-scale networks where link bandwidth is not a bottleneck, it is possible to ensure a 100% backtrack-free search by ensuring that all link mapping domains are capacity consistent according to Theorem 4.5.

3) BIRD-VNE *Minimizes and Bounds the Average Cost:* The approximation ratio is assessed by comparing the cost achieved by BIRD-VNE to the optimal cost achieved by branch and bound for a substrate network with 30 nodes. As shown in Fig. 4-c, the cost achieved by BIRD-VNE is, on average, only about 5% higher than the optimal cost (i.e., average ratio = about 1.05). But the maximum cost can reach up to 70% higher than the optimal cost (maximum ratio = about 1.7).

4) BIRD-VNE *Results in the Best Revenue to Cost Ratio:* The revenue to cost ratio reflects the average profit of BIRD-VNE, and is 20% better than RVINE-MCF as shown in \figure-name Fig. 7. This is expected for two reasons. First, BIRD-VNE is a $\frac{1}{2}$-approximation of the optimal cost, which contributes to the R/C ratio by minimizing the cost. Second, we have shown numerically that BIRD-VNE has the highest acceptance rate, which directly reflects on the total generated revenue by accepting as many virtual network requests as possible.

5) BIRD-VNE *Link Utilization is Better:* The average link utilization achievable under BIRD-VNE is comparable to that achievable under RVINE-MCF when considering various inter-arrival rates, as shown in Fig. 10. However, for higher loads, the average link utilization of BIRD-VNE is less that that of RVINE-MCF, which confirms our earlier argument stating that BIRD-VNE tends to allocate shorter substrate paths to the virtual links with higher demands. On the other hand, the average node utilization achieved by BIRD-VNE is generally greater than that achieved by RVINE-MCF due to the better acceptance rates.

6) BIRD-VNE *Reduces the Migration Ratio:* Fig. 9 shows the effectiveness of BIRD-VNE in minimizing the migration ratio. In this figure, Mobility-Aware Bird-VNE corresponds to $\gamma(v) = 1$ and Bird-VNE corresponds to $\gamma(v) = 0$. Even when the migration cost is low (i.e., $\gamma(v) = 1$), BIRD-VNE can reduce the migration ratio by at least 10%. This gain can be increased by increasing the migration cost (γ), which is a design trade-off. Observe that because of mobility, about 50% of the accepted virtual networks face migrations.

VII. DISCUSSION AND PRACTICAL CONSIDERATIONS

Several architectural and practical considerations pertain to the discussed virtual network network embedding solution. We discuss some possible approaches to address these challenges.

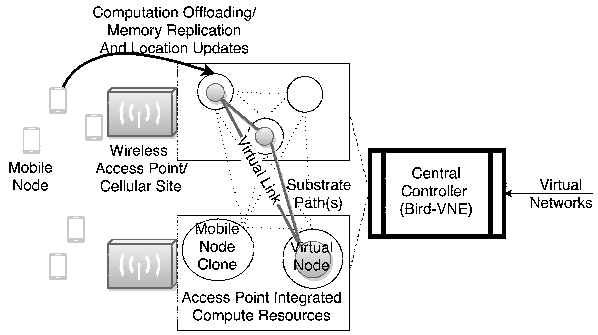

Fig. 11. Architecture: Mobile nodes are connected through wireless infrastructure with integrated compute resources that host clones of mobile nodes and actually implement the requested virtual networks.

Topology changes: Substrate nodes are generally resources limited (e.g. smart-phones) and mobile which results in network topology changes that require updating all the substrate paths computations following any topology change. Updating all paths, **R**, can be addressed architecturally or algorihmically. Fig. 11 shows a possible architecture utilizing the emerging mobile edge computing to address this challenge by augmenting a wireless network infrastructure with distributed cloud resources. Each mobile node replicates its data and states (e.g sensors measurements, locations) to a corresponding clone that is proximate to the node (i.e. at the access point or cellular site). Clones are the actual entities that shall execute the virtual network requests. Cloning the mobile nodes provides several advantages over executing the virtual networks directly on the mobile nodes including: (i) providing manageable and salable processing and link capacity according to virtual networks demands, (ii) facilitating energy conservation of the actual mobile nodes which may be power limited (e.g. sensor nodes), (iii) preventing excessive latency compared to replicating nodes' data in *distant data-centers*, and (iv) preventing substrate network topology changes due to mobility. Unfortunately, the architecture shown in Fig. 11 is not sufficient to prevent updating **R** in some cases such as back-hauling links or node failures or changes in clones deployment. Fortunately updating the set of all paths **R** is not as expensive as computing it from scratch and has remarkable long research history. The authors in [40], for example, study the combinatorial properties of graphs that can be used to update all shortest paths in dynamic networks in $O(n^2 \log^3 n)$ which is *not* a dominant factor in the complexity analysis of our proposed techniques as discussed in Section IV and Section V.

Mobility Model: the general RWP model cannot capture exact mobility patterns especially in walking scenarios. However, the recent modifications of the RWP model in [36] captures the mobility patterns almost exactly at the accuracy of cell level in 3GPP cellular networks which is suitable for several applications such as virtual sensor networks, and virtual content delivery networks. If a finer grain location resolution (e.g. locations of pedestrians at few meters error) were needed by some applications, the RWP model may fail to capture the exact mobility trajectory. In such cases, one can employ other mobility models that characterize smooth movements of mobile nodes (see for e.g. the Semi-Markov Smooth model [41]), or employ model independant trajectory tracking methods (e.g. Kalman filtering) to track nodes locations. Such methods are outside the scope of this paper.

Multipath adoption: If multipath were allowed for mapping virtual links, we conjecture improvements particularly in the acceptance rate [11]. First, multipaths shall allow online path optimization, and traffic splitting for highly demanding virtual links. Second, multipaths shall increase link utilization making the most benefits of the network. Third, multipaths shall facilitate a better sharing of mobile wireless nodes. Finally, multipaths shall allow balancing the substrate network traffic used by the virtual networks and already existing services.

VIII. Conclusion

The coupled constraints in the virtual network embedding problem make it intractable. Instead of over-provisioning the physical network and splitting virtual links across multiple paths, we propose VNE techniques that effectively prune the search space, thereby reducing the execution times by avoiding backtracking, while not compromising the quality of the obtained VNE solutions, expressed in terms of acceptance rates. Our simulations show that the likelihood of performing a backtrack-free search is greater than 80%, confirming the effectiveness of the proposed pruning techniques. These techniques are then exploited to design a polynomial-time, $\frac{1}{2}$-approximation VNE algorithm. We show analytically and empirically that the proposed algorithm outperforms MIP-based algorithms in terms of the revenue to cost ratio and the acceptance rate while minimizing the migration cost arising due to the mobility of physical nodes.

References

[1] S. Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes, "Enabling smart cloud services through remote sensing: An internet of everything enabler," *IEEE Internet Things J.*, vol. 1, no. 3, pp. 276–288, Jun. 2014.

[2] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, 2014, pp. 241–246.

[3] ETSI, "Mobile-edge computing," European Telecommunications Standards Institute (ETSI), Technical White Paper, Sep. 2014.

[4] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Socio-aware virtual network embedding," *IEEE Netw.*, vol. 26, no. 5, pp. 35–43, Sep./Oct. 2012.

[5] K. Zheng *et al.*, "Research and standards: Advanced cloud and virtualization techniques for 5G networks (guest editorial)," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 16–17, Jun. 2015.

[6] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, "Greenhead: Virtual data center embedding across distributed infrastructures," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 36–49, Jan./Jun. 2013.

[7] M. Abu Sharkh, M. Jammal, A. Shami, and A. Ouda, "Resource allocation in a network-based cloud computing environment: Design challenges," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 46–52, Nov. 2013.

[8] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 10–17, Aug. 2001.

[9] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, "Adaptive virtual network provisioning," in *Proc. 2nd ACM SIGCOMM Workshop Virtualized Infrastruct. Syst. Archit.*, 2010, pp. 41–48.

[10] S. Lo, M. Ammar, E. Zegura, and M. Fayed, "Virtual network migration on real infrastructure: A planetlab case study," in *Proc. IFIP Netw. Conf.*, 2014, pp. 1–9.

[11] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008.
[12] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, Nov. 2013.
[13] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. ACM Workshop Virtualized Infrastruct. Syst. Archit.*, 2009, pp. 81–88.
[14] I. Houidi and D. Zeghlache, "Exact adaptive virtual network embedding in cloud environments," in *Proc. IEEE 22nd Int. Workshop Enabling Technol. Infrastruct. Collaborative Enterprises (WETICE)*, 2013, pp. 319–323.
[15] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer, "Energy efficient virtual network embedding," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 756–759, May 2012.
[16] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, 2011.
[17] J. F. Botero, X. Hesselbach, A. Fischer, and H. De Meer, "Optimal mapping of virtual networks with hidden hops," *Telecommun. Syst.*, vol. 51, no. 4, pp. 273–282, 2012.
[18] R. Dechter, *Constraint Processing*. San Mateo, CA, USA: Morgan Kaufmann, 2003.
[19] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *Int. J. Commun. Syst.*, vol. 26, no. 8, pp. 1054–1073, 2013.
[20] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. INFOCOM*, 2006, pp. 1–12.
[21] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.
[22] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 16th Annu. ACM Symp. Theory Comput.*, 1984, pp. 302–311.
[23] M. Till Beck, A. Fischer, H. de Meer, J. F. Botero, and X. Hesselbach, "A distributed, parallel, and generic virtual network embedding framework," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2013, pp. 3471–3475.
[24] J. F. Botero, M. Molina, X. Hesselbach-Serra, and J. R. Amazonas, "A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems," *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1735–1752, 2013.
[25] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Quanitsubstrate," in *Proc. IEEE INFOCOM*, 2014, pp. 1–9.
[26] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 816–827, Mar. 2014.
[27] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Trans. Netw. Serv. Manage.*, vol. 10, no. 4, pp. 356–368, Dec. 2013.
[28] S. Su, Z. Zhang, A. X. Liu, X. Cheng, Y. Wang, and X. Zhao, "Energy-aware virtual network embedding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607–1620, Oct. 2014.
[29] A. Jarray and A. Karmouch, "Cost-efficient mapping for fault-tolerant virtual networks," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 668–681, Mar. 2015.
[30] S. Abdelwahab, B. Hamdaoui, and M. Guizani, "BIRD-VNE: Backtrack-avoidance virtual network embedding in polynomial time," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2014, pp. 4983–4989.
[31] D. Yun and Y. Yi, "Virtual network embedding in wireless multihop networks," in *Proc. 6th Int. Conf. Future Internet Technol.*, 2011, pp. 30–33.
[32] D. Yun, J. Ok, B. Shin, S. Park, and Y. Yi, "Embedding of virtual network requests over static wireless multihop networks," *Comput. Netw.*, vol. 57, no. 5, pp. 1139–1152, 2013.
[33] X. Wang, P. Krishnamurthy, and D. Tipper, "Wireless network virtualization," in *Proc. IEEE Int. Conf. Comput. Netw. Commun. (ICNC)*, 2013, pp. 818–822.
[34] G. Sun *et al.*, "Adaptive provisioning for evolving virtual network request in cloud-based datacenters," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2012, pp. 1617–1622.
[35] D. Arora, A. Feldmann, G. Schaffrath, and S. Schmid, "On the benefit of virtualization: Strategies for flexible server allocation," in *Proc. USENIX Workshop Hot Topics Manage. Internet Cloud Enterprise Netw. Serv. (Hot-ICE)*, 2011, pp. 2–2.
[36] X. Lin, R. Ganti, P. Fleming, and J. Andrews, "Towards understanding the fundamentals of mobility in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1686–1698, Apr. 2013.
[37] J.-C. Régin, "A filtering algorithm for constraints of difference in case csps," in *Proc. 12th Nat. Conf. Artif. Intell. AAAI*, 1994, vol. 94, pp. 362–367.
[38] J. E. Hopcroft and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput.*, vol. 2, no. 4, pp. 225–231, 1973.
[39] A. Fischer, J. F. Botero, M. Duelli, D. Schlosser, X. Hesselbach, and H. De Meer, "ALEVIN-A framework to develop, compare, and analyze virtual network embedding algorithms," *Electron. Commun. EASST*, vol. 37, pp. 1–12, 2011.
[40] C. Demetrescu and G. F. Italiano, "A new approach to dynamic all pairs shortest paths," *J. ACM*, vol. 51, no. 6, pp. 968–992, 2004.
[41] M. Zhao and W. Wang, "A unified mobility model for analysis and simulation of mobile wireless networks," *Wireless Netw.*, vol. 15, no. 3, pp. 365–389, 2009.

Sherif Abdelwahab (S'07–M'11–SM'14) received the B.S. and M.S. degrees in electronics and communications engineering from Cairo University, Giza, Egypt, in 2004 and 2010, respectively. He is currently pursuing the Ph.D. degree at the School of Electrical Engineering and Computer Science (EECS), Oregon State University, Corvallis, OR, USA. Previously, he was with the mobile networks industry with Alcatel-Lucent from 2004 to 2007 and with Etisalat from 2008 to 2013. His research interests include distributed networking, networked systems and services, mobile and wireless networks.

Bechir Hamdaoui (S'02–M'05–SM'12) received the Diploma of Graduate Engineer from the National Engineering School of Tunis, Tunis, Tunisia, the M.S. degrees both in electrical and computer engineering and computer sciences, and the Ph.D. degree in electrical and computer engineering from the University of Wisconsin at Madison, Madison, WI, USA, in 1997, 2002, 2004, and 2005, respectively. He is currently an Associate Professor with the School of EECS, Oregon State University, Corvallis, OR, USA. His research interests include distributed resource management and optimization, parallel computing, co-operative and cognitive networking, cloud computing, and Internet of Things. He is currently an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (2013–present), and the *Wireless Communications and Computing Journal* (2009–present). He also served as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (2009–2014) and *Journal of Computer Systems, Networks, and Communications* (2007–2009). He served as the Chair for the 2011 ACM MobiComs SRC Program, and as the Program Chair/Co-Chair of several IEEE symposia and workshops (including GC 2016, ICC 2014, IWCMC 2009–2015, CTS 2012, PERCOM 2009). He also served on the technical program committees of many IEEE/ACM conferences, including INFOCOM, ICC, GLOBECOM, and PIMRC. He was the recipient of the NSF CAREER Award in 2009.

Mohsen Guizani (S'85–M'89–SM'99–F'09) received the B.S. (with distinction) and M.S. degrees in electrical engineering, the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a Professor and the Chair of the Department of Electrical and Computer Engineering with the University of Idaho, Moscow, ID, USA. Previously, he worked with Western Michigan University, Kalamazoo, MI, USA, University of West Florida, Pensacola, FL, USA, Kuwait University, Kuwait City, Kuwait, and Qatar University, Doha, Qatar. He is the author of eight books and more than 400 publications in refereed journals and conferences. His research interests include computer networks, wireless communications and mobile computing, security, and Internet of Things. He currently serves on the editorial boards of six technical journals. He is a Senior Member of ACM.

Taieb Znati's research interests include the design and analysis of evolvable, secure and resilient network architectures and protocols for wired and wireless communication networks, and the design of new fault-tolerant mechanisms for energy-aware resiliency in data-intensive computing. He is also interested in bio-inspired approaches to address complex computing and communications design issues that arise in large-scale heterogeneous wired and wireless networks. He has served as the General Chair of several main conferences, including GlobeCom 2010, the IEEE INFOCOM 2005, SECON 2004, the first IEEE conference on Sensor and Ad Hoc Communications and Networks, the Annual Simulation Symposium, and the Communication Networks and Distributed Systems Modeling and Simulation Conference. He also served or currently serves as a member of the editorial boards of a number of networking, distributed system and security journals and transactions.