IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 2, April 2026, pp. 1181~1193
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i2.pp1181-1193

Knowledge graph-based enhanced virtual network embedding for 6G cloud datacenter deployment

Shourok Abdelrahim[1], Samy Ghoniemy[2], Mohamed Aborizka[1]
[1]College of Computing and Information Technology, Arab Academy for Science and Technology, Cairo, Egypt
[2]Department of Computer Science and Engineering, Faculty of Computer Science and Engineering, Galala University, Cairo, Egypt

Article Info

Article history:

Received May 24, 2025
Revised Jan 19, 2026
Accepted Feb 6, 2026

Keywords:

6G cloud datacenter
Knowledge graph
Network mapping
Virtual network embedding
Virtual network inference predicate

ABSTRACT

Virtual network embedding (VNE) is the effective mapping of virtual networks onto shared physical substrate networks while boosting resource utilization and ensuring quality of service (QoS). VNE is a real challenge in network virtualization, especially in the perspective of 6G-enabled datacenters, where the demand for ultra-low latency, heavy connectivity, and dynamic resource allocation is vital. The proposed solution enables the ability to infer indirect paths for the resources prediction task on the knowledge graph (KG) by making implicit meaningful relations among the entities that compose the resource network. The simulation results indicated the inference mechanism significantly improves efficiency and adaptability. This leads to overall performance gains in terms of runtime stability, resource utilization, and energy savings in dynamic 6G scenarios. The experimental results showed that the proposed solution provided a 24.9% reduction in energy consumption for small-sized virtual network requests (VNRs), while maintaining 24.8% and 23.9% for medium and large VNRs, respectively, while it significantly decreased the delay time compared to the resulted delay using the baseline models such as asynchronous advantage actor-critic (A3C) + graph convolutional network (GCN). The results also confirmed that the integration of the inference engine algorithm with the embedding process results in remarkable reduction in the execution time while preserving embedding accuracy.

Corresponding Author:

Shourok Abdelrahim
College of Computing and Information Technology, Arab Academy for Science and Technology
Cairo, Egypt
Email: shorouk_abdelrahim@cic-cairo.com

1. INTRODUCTION

Many research outcomes, such as in [1]–[4] confirmed that mapping virtual network requests (VNRs) onto physical infrastructures while guaranteeing optimal resource utilization and scalability is a real challenge in network virtualization. This challenge arises from the complexity of resource allocation and the constraints of network topology [5]. From an artificial intelligence (AI) perspective, virtual network embedding (VNE) can be regarded as a decision-making and combinatorial optimization task that requires reasoning about multiple objectives and hidden relationships among network elements. Recent studies have demonstrated a notable advancement in the deployment of VNE and resource allocation techniques using AI methods such as machine learning (ML), deep reinforcement learning (DRL), and graph neural networks (GNN) [6]–[10], which enhance network prediction and mapping accuracy to cope with the large volume and variable rate of requests in 6G-enabled datacenters. However, these solutions often require extensive training data and high computational resources, which limit their practicality for real-time multi-access services in

dynamic 6G networks. Other solutions such as dynamic programming-based algorithm [11], reinforcement learning (RL)-based dynamic collaborative multi-layer VNE algorithm [12], and Markov decision process (MDP) model [13], aimed to improve the load balancing and optimized resource allocation by coordinating node and link mapping. They failed to reduce computational complexity due to the lack of adaptive learning limits which limit their suitability in 6G dynamic network environments. Zhang *et al.* [14], [15] proposed a combining horizontal federated and RL techniques to improve mapping accuracy and optimize resource utilization. However, they introduced significant computational overhead due to the use of breadth-first search (BFS) in link mapping. More recently, knowledge graphs (KGs) have emerged as a promising solution for intelligent VNE optimization. One of these solutions is the proposed framework that combines AI-based decision making and cognitive resource optimization algorithms for 6G networks [16], but highly complex and dynamic networks limit their scalability.

Also, Mitropoulou *et al.* [17], [18] proposed the use of static KG and ML to detect and mitigate network congestion and anomaly detection. However, limiting the full exploitation of latent relational inferences. Similarly, other researchers proposed graph convolutional network (GCN) using KG for VNE. Their framework showed lower efficiency in real-time implementations [19]. At the end of this review, an interesting work proposed in [20] presented a combination of PageRank-based heuristic VNE and Bellman-Ford algorithm to rank nodes. Although this framework improves the performance, its static feature set and lack of mapping coordination impact its adaptability in dynamic environments. To consolidate the discussion, Table 1 provides a comparative overview of representative VNE approaches, including heuristic, DRL, GNN, federated learning (FL), and KG-based methods, benchmarked against the proposed dynamic KG framework. This summary highlights the trade-offs among scalability, computational overhead, adaptability, and performance metrics, clarifying where the proposed solution advances the state of the art.

Table 1. Comparative analysis of representative VNE approaches for 6G datacenters

Approach	Representative works	Key features	Scalability & adaptability	Overhead & limitations	Reported metrics
Heuristic	VNE-NR [20]	Node ranking; shortest-path mapping	Medium scalability; low adaptability	Low overhead; static features; brittle under dynamics	Improved performance, but limited adaptability
DRL	Asynchronous advantage actor-critic (A3C) +GCN [21]; DeepViNE [22]	End-to-end node/ link policy learning	Medium scalability; moderate adaptability	High overhead; long convergence; sensitive to training	Runtime 0.219–0.476 s; acceptance 82–87% @1100-time units
GNN + RL	GCN+RL [23]; GraphViNE [24]	GCN encoders + RL policies	Medium–high scalability; moderate adaptability	Medium–high overhead; convergence delays	Acceptance 85–90% @1100-time units
FL	Horizontal FL-VNE [15]	Cross-domain learning	Medium scalability; moderate adaptability	High overhead (BFS link mapping); scalability issues	Accuracy gains reported; runtime not detailed
KG (non-VNE)	[17]–[19]	Semantic/relational modeling	Medium scalability; low adaptability	Medium overhead; static KGs; limited inference	Metrics reported for anomaly/link prediction
Proposed	This study	Dynamic KG + hidden inference engine	High scalability; high adaptability	Low–medium overhead; requires semantic enrichment	Runtime 0.65–0.85 s; utilization 85%; energy −24.9%; acceptance 94% @900-time units

Unlike previously reported KG-based methods, which are typically static and struggle to capture deeper relational features as graph size increases, the proposed framework employs a time-evolving, semantically enriched KG integrated with a hidden-relation inference engine operating directly within the VNE pipeline. This integration enables the inference of indirect or distant resource connections, accelerates stabilization of the acceptance ratio, and delivers consistent energy reductions, while also avoiding the heavy training overheads associated with DRL- and GNN-based approaches. To address these limitations, this paper presents an intelligent VNE framework based on dynamic KG which is able to dynamically optimize resource allocation and improve the latent computational capabilities by inferring hidden relationships between the network assets and resources. The proposed solution offers a novel mechanism for improving the efficiency and scalability of VNE in 6G-datacenters.

2. PROPOSED SYSTEM

In this paper, the proposed framework is designed based on the semantically enriched KG model that is developed to unify the nodes and edges generation with technical specifications and semantics description that represent the datacenter assets. In the proposed framework, the core element is the KG-based novel inference algorithm, which is designed based on the semantically enriched KG model, to enable adaptable and dynamic VNE, and to reason and infer the latent relationships among physical infrastructure components in the substrate network. The proposed framework also includes proactive resource allocation module, fault anticipation process, and intelligent traffic monitoring and management agents, as depicted in Figure 1.

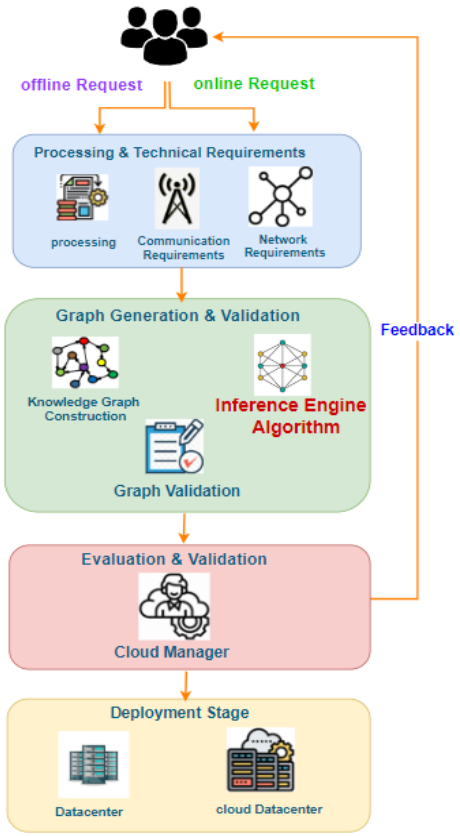

Figure 1. The proposed system

2.1. Technical description collection and validation phase

This phase is responsible for collecting and validating detailed technical descriptions of datacenter assets, including servers, memory units, processors, and storage devices. The collected technical descriptions undergo preprocessing steps such as data cleaning, normalization, and feature extraction. Following the collection of technical descriptions, the validation process is initiated to ensure that the collected information meets the standards. In this process we adopted and applied a capacity validation (CV) rule given in (1) to assess whether the available resources are sufficient to meet operational demands.

$$CV = \left(\frac{R_{available}}{R_{required}}\right) \times 100 \quad (1)$$

$R_{available}$ represents the total available resources and $R_{required}$ denotes the necessary resources. By utilizing predefined thresholds, the CV enhances load balancing, reduces computational bottlenecks, and guarantees effective resource management in the following phases.

During semantic enrichment step, each datacenter asset is described using tuple of normalized attributes including CPU capacity, memory, storage, and bandwidth. These values are first normalized to a [0, 1] range. Next, entity linking is performed by mapping each attribute to an ontology term representing its

semantic category. This ontology-based labeling ensures that heterogeneous resources are represented in a unified schema within KG. By embedding nodes with both structural identifiers and semantic tags, KG captures not only resource quantities but also their contextual meaning, which is critical for subsequent inference.

2.2. Training and validating the proposed model

In this phase, unstructured descriptions are transformed into semantically structured KG that models the datacenter assets. To assess the correctness of the inferred relations within the KG, the model is validated using standard classification metrics including accuracy, precision, recall, and F1-score [25]. Accuracy given in (2), measures the ratio of correctly classified instances, precision and recall given in (3), evaluate prediction correctness and detection capability, respectively. The F1-score given in (4) is adopted as a balanced metric to jointly account for false positives and false negatives.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

$$Recall = \frac{TP}{TP+FN} \text{ and } Precision = \frac{TP}{TP+FP} \tag{3}$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \tag{4}$$

2.3. Generation and validation of the knowledge graph

The visualization of semantically structured elements that represent the datacenter assets, and the interconnections and interactions between these assets and provide enhanced semantic reasoning and decision-making is shown in Figure 2. The sufficiency condition (S) given by (5) is used as a critical validation parameter that reveals whether the selected resources are capable for continuing system operations.

$$S = \sum_{i=1}^{n}(A_i - R_i) \tag{5}$$

A_i and R_i denote the available and required resources at node i, respectively, and n is the absolute number of nodes. If $S < 0$, means the system has resources restrictions and requires the integration of additional nodes.

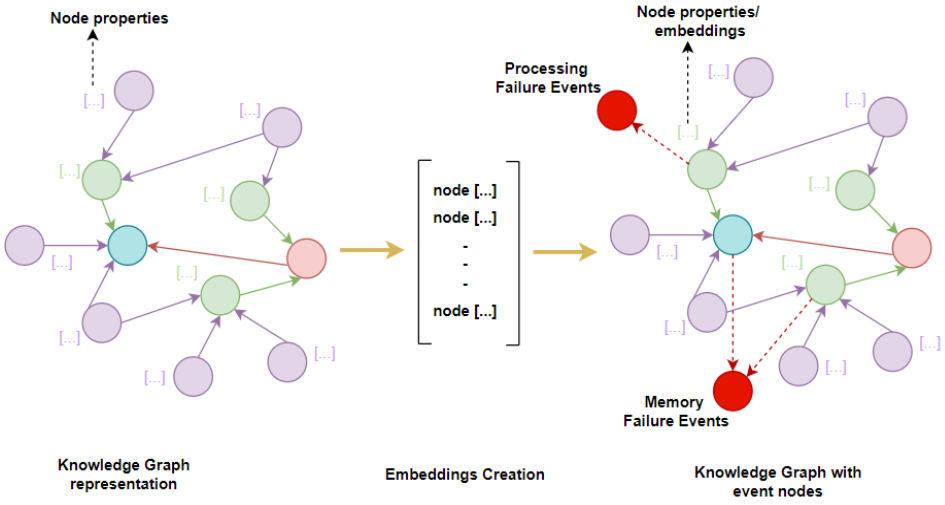

Figure 2. Knowledge graph construction

2.4. Hidden relations phase: virtual network embedding in dynamic environments

A VNR is represented as $VNR_i=(GV_i, t_a, t_e)$, where t_a and t_e denote the arrival and departure times, respectively. The VNE process maps the virtual network GV_i (N_V, L_V) onto a subgraph of the physical substrate network GP' (N'_P, L'_P) that satisfies the resource constraints. Figure 3 illustrates an example of virtual nodes and links embedded onto the physical infrastructure [23], where Figure 3(a) shows the condition before VNRs embedding and Figure 3(b) shows the condition after VNRs embedding.

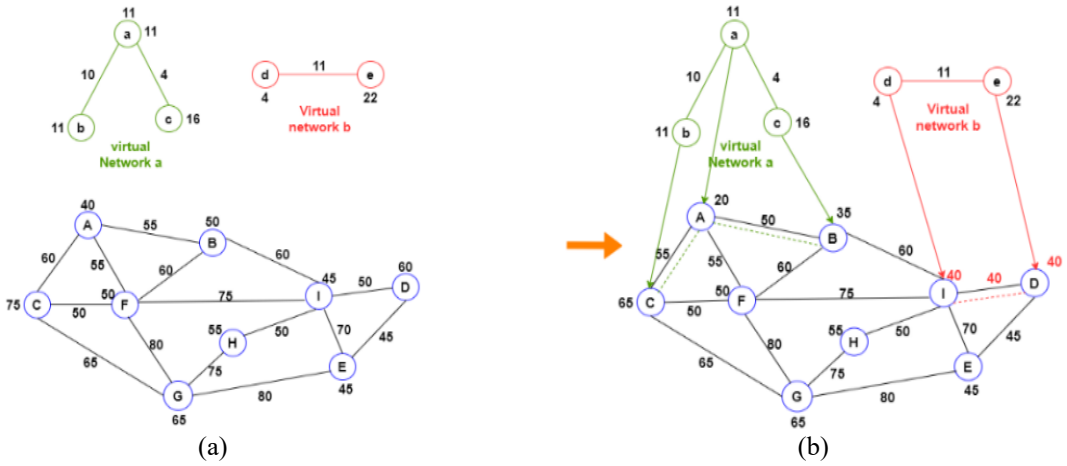

Figure 3. Diagram of the physical and virtual network structures: (a) before VNRs embedding and (b) after VNRs embedding

In practical 6G datacenter environments, virtual networks change over time due to dynamic user demands, topology reconfiguration, and varying resource requirements. This defines the dynamic VNE problem, where embedding decisions must continuously adapt to changes in virtual topology and resource demand. Figure 4 shows representative structural and resource changes across successive time intervals. To support this dynamic behavior, the proposed framework integrates hidden-relation inference within a time-evolving KG, enabling adaptive embedding decisions without reprocessing the entire topology. To handle this dynamic behavior, we propose a VNE framework based on hidden relation inference within a time-evolving KG. Latent relationships among nodes and links as shown in Figure 5 are inferred using statistical modeling and link prediction, allowing for real-time adaptation to topology modifications and fluctuating resource demands.

In real-time operation, updates to the virtual network are managed through a sliding observation window T. At the beginning of each window, new events are detected, and the KG is updated accordingly. Each modification triggers re-inference for only the changed subgraph, rather than re-processing the entire topology, which reduces overhead. Eligibility of inferred links is then validated against the predefined thresholds γ (semantic affinity) and $R_{\text{threshold}}$ (resource feasibility). If both conditions are satisfied, the relation is added to the candidate embedding set; otherwise, it is discarded. This incremental update mechanism enables the system to adapt quickly to topology and demand variations without restarting the embedding process.

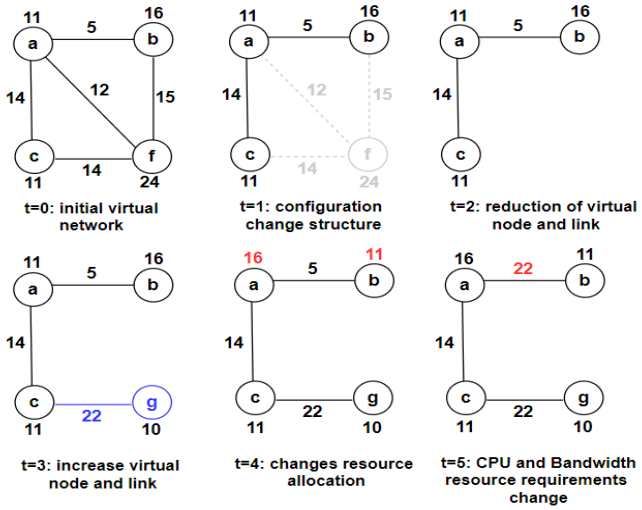

Figure 4. Example illustrating the operation of dynamic VNE

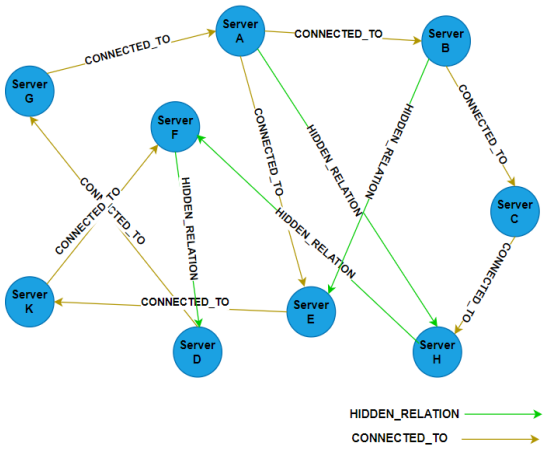

Figure 5. Hidden relation in knowledge graph

2.4.1. Knowledge-based latent relations modeling

The proposed system introduces an inference-driven framework that leverages a semantically structured KG to identify latent node dependencies and enables dynamic resource-aware decision-making through a binary indicator function $\in_{uv}$, as defined in (6), that quantifies the existence of implicit relations between nodes.

$$\in_{uv} = \begin{cases} 1, & f(u,v) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The subsequent mathematical expressions in (7) to (14) systematically outline the core eligibility requirements and optimization criteria that underpin the proposed inference mechanism.

$$\emptyset(u,v) = \sigma\big(A(u)\ W\ A(v)\big) \tag{7}$$

$$f(u,v) = \begin{cases} 1, & \text{if } \emptyset(u,v) \geq \gamma \text{ and } \mathrm{pd}(u,v) \leq R_{\text{threshold}} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$\mathrm{HCC}(u) = \begin{cases} 1, & \exists v \in V\ \{u\} \text{ such that } f(u,v) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$s(u) = \sum_{v \in V \setminus \{u\}} \mathbb{I}\ (f(u,v) = 1\ \wedge\ \emptyset(u,v) \geq \gamma) \tag{10}$$

$$e(u) = \sum_{v \in V \setminus \{u\}} \mathbb{I}\ (f(u,v) = 0\ \wedge\ \mathrm{pd}(u,v) \leq R_{\text{threshold}}) \tag{11}$$

$$s(u) + e(u) = |V| - 1 \Rightarrow u \in V\ S \tag{12}$$

$$\Omega\ =\ \sum_{u \in V\ S} \sum_{V \in V \setminus \{u\}} f(u,v) \tag{13}$$

$$\text{Maximize } \Omega \text{ subject to } \mathrm{HCC}(u) = 1, \forall u \in V\ S \tag{14}$$

In (7) shows the semantic affinity between any pair of nodes is quantified using a similarity function $\emptyset(u,v)$. The output $\emptyset(u,v) \in [0,1]$ denotes the probabilistic confidence of a latent semantic linkage. In (8) indicates the legitimacy of such inferred relations. To regulate participation in the embedding process, we define a connectivity constraint $\mathrm{HCC}(u)$ in (9) that evaluates the existence of at least one valid inferred link for a node u. In (10) denotes the strong relational intensity of a node u as $s(u)$, representing the cardinality of semantically robust inferences it holds with others in the graph. Conversely, the weak inference degree $e(u)$, is denoted in (11) enumerates topologically valid yet semantically marginal connections. A node u qualifies for inclusion in the final embedding set $V\ S$ f the aggregate of its strong and weak inferences spans the entirety of the graph minus itself, as stated in (12). In (13) shows the global inferential coherence of the

subgraph; we define the inference power metric Ω as the summation of all confirmed inferences across the selected node set. In (14) indicate the overarching optimization task aims to maximize the inference utility Ω constrained by the connectivity condition $\mathrm{HCC}(u) = 1$ for every candidate node $u \in V\,S$. The procedural steps are outlined in Algorithm 1, and its flowchart representation is illustrated in Figure 6.

Algorithm 1. Inference engine for hidden paths prediction
Input: G(V, E), resource requirements R, observation window T
1: Initialize VS ← ∅; n ← |V|; C ← n; mpd(G) ← n – 1
2: for each time step t ∈ T do
3: Monitor VNR events (arrival, update, deletion)
4: Update G by modifying affected nodes/edges
5: Mark changed nodes for re-inference
6: while ∃ unprocessed u ∈ V do
7: if mpd(G) < R_threshold then
8: Perform community detection CD(G)
9: Select u where state(u) = 0 ∧ pd(u) ≥ R_threshold
10: while s(u) + e(u) = n − 1 do
11: VS ← VS ∪ {u}
12: u ← next valid unprocessed vertex
13: end while
14: mpd(G) ← 0
15: for each u ∈ V do
16: Select v where pd(u, v) ≥ R_threshold
17: Probe connection strength f(u, v)
18: if f(u, v) = ∅ then
19: e(u)++, e(v)++; state(u)--, state(v)--
20: else
21: s(u)++, s(v)++
22: if HCC(u) = 0 or HCC(v) = 0 then
23: return ∅
24: if s(u) + e(u) = n − 1 then VS ← VS ∪ {u}
25: if s(v) + e(v) = n − 1 then VS ← VS ∪ {v}
26: end for
27: end while
28: end for
29: return VS with associated inference metrics

The computational complexity of the algorithm can be analyzed as follows: i) monitoring and updating the KG across $|V|$ nodes requires $O(|V|)$; ii) community detection (line 8) is bounded by $O(|E|\,log|V|)$; and iii) affinity scoring and validation (lines 16–25) require $O(|V|^2)$ in the worst case. Overall, the total runtime per observation $window\ T$ is approximately $O(|V|^2 + |E|\,log|V|)$. Which is tractable for medium- to large-scale topologies when compared with DRL training loops.

A critical challenge in dynamic environments is concept drift, where the statistical properties of VNRs evolve over time. To address this, the framework continuously re-validates semantic similarity values $\emptyset(u, v)$ (6) and thresholds (7) at each observation window. If previously inferred relations no longer meet the semantic or resource thresholds, they are pruned from the KG, ensuring that embedding remains consistent with current network conditions.

2.5. Optimization and decision-making

This phase evaluates the effectiveness of the proposed inference-driven VNE framework using runtime efficiency, resource utilization, load balancing, and fault recovery metrics. These indicators quantify the impact of hidden relation inference on embedding performance under dynamic constraints.

i) Runtime acceleration: runtime is measured as the time required to embed a VNR in the substrate network, as given in (15).

$$Running\ Time = T_{embedding} \tag{15}$$

where $T_{embedding}$ is the time required to generate and correctly embed a VNR in the substrate network.

ii) Resource utilization efficiency: this metric is intended to assess how well the inference-based latent relation detection algorithm (Algorithm 1) finds hidden links between network items. Resource utilization reflects the effectiveness of inferred relations in guiding node and path selection.

iii) Decision validation: the load balance index (LBI) as in (16) and fault recovery efficiency as in (17) are used to evaluate workload distribution and system resilience.

$$LBI = 1 - \frac{\sigma_w}{\mu_w} \tag{16}$$

$$FRE = \frac{MTTF}{MTTF+MTTR} \tag{17}$$

iv) System evaluation phase: in (18) is used to evaluate the overall effectiveness and scalability of the proposed system. A higher acceptance ratio means better load distribution, better embedding, and increased flexibility in response to changing network demands.

$$Acceptance\ Ratio = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} V_S}{\sum_{t=0}^{T} V} \tag{18}$$

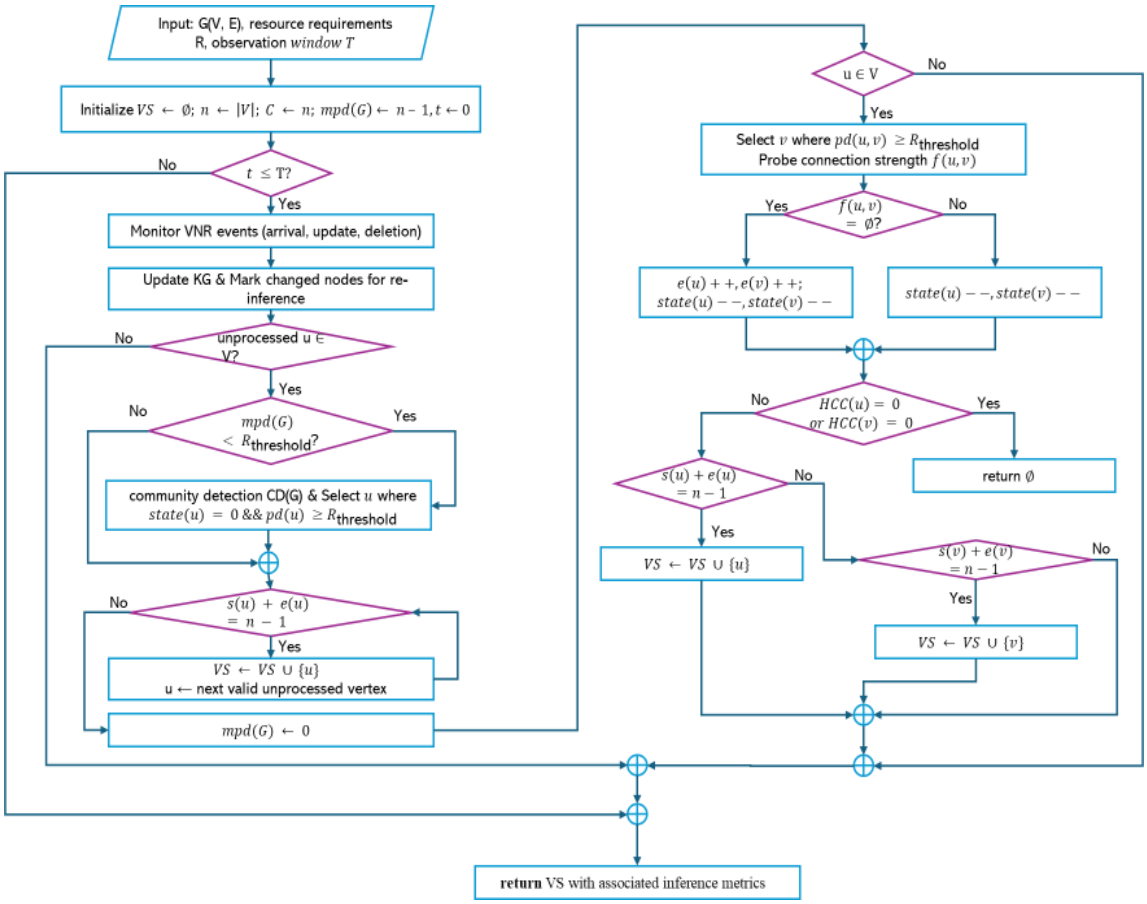

Figure 6. Flowchart of Algorithm 1 illustrating the main procedural steps

3. SIMULATION SETUP

Simulations were conducted using Mininet [26] and ONOS [27] under varying substrate sizes, resource capacities, and VNR arrival patterns to evaluate scalability and performance. The simulation experiments have been conducted using variety of CPUs ranging from 5 to 50 with up to 100 processing nodes, and 500 communication links with bandwidth ranging from 10 to 100 Mbps. The VNRs varied from 2 to 50 virtual nodes, and different arrival rate, to test scalability and the algorithm's efficiency in terms of runtime and resource utilization in 6G datacenters.

3.1. Dataset schema and preprocessing

Datacenter assets were represented as a semantically enriched KG with normalized node and link attributes. The preprocessing includes: i) type harmonization (SI units), ii) min–max normalization of numerical attributes to [0, 1], iii) deduplication of exact/near-duplicate assets, iv) ID canonicalization for entity linking, v) outlier clipping at 1st/99th percentiles, and vi) ontology tagging (e.g., Compute.CPU, Network.Bandwidth) for semantic enrichment. This ontology tagging was used in similarity function $\varphi(u, v)$.

3.2. Hyperparameter table

All relevant experimental parameters and symbols used in this study are summarised in Table 2. These include the γ, $R_{\text{threshold}}$, T, seeds K, virtual network size, substrate size, CPU units, bandwidth limits, community detection, and convergence guard. These parameters determine the control behavior of the algorithm (e.g. early termination and $R_{\text{threshold}}$) as well as the simulation environment and resources constraints (e.g. CPU, bandwidth, and virtual network size of the nodes). In order to ensure a fair comparison of the trials and the reproducibility of the results, the trials shall be set up and clearly reported.

Table 2. Simulation parameters and hyperparameter settings

Symbol / name	Meaning	Value/ range
γ	Semantic affinity threshold in (6 and 7)	0.65 (default), swept 0.5–0.8
$R_{\text{threshold}}$	Resource/feasibility cutoff (path/capacity)	0.6 of residual capacity (link+node)
T	Observation window (re-inference period)	50-time units
Seeds K	Independent runs per scenario	10
Virtual network size	VNR virtual nodes per request	2–32 (also tested 2–50 in stress)
Substrate size	Physical nodes/links	up to 100 nodes/500 links
CPU units	Per node capacity (normalized)	5–50 units
Bandwidth limits	Per link capacity	10–100 Mbps
Community detection	Trigger & method	When avg. $pd(u) < R_{\text{threshold}}$
Early-stop	Convergence guard	No change in acceptance for 3 windows

4. RESULTS AND DISCUSSION

The performance of the VNE framework was evaluated through a range of scenarios. It was assessed using key performance metrics, including runtime, nodes and links utilization, load balancing, energy efficiency, and acceptance ratio. To validate the results, our results compared with the results of state-of-the-art algorithms, including GCN+RL [23], GraphViNE [24], A3C+GCN [21], and DeepViNE [22].

4.1. Runtime acceleration

Table 3 reports runtime results under varying conditions. Unless otherwise stated, all reported results represent the average of 10 runs, with standard deviation indicated. Significance was tested using paired t-tests at a 95% confidence level ($p < 0.05$). Compared to A3C+GCN [21] where they presented a runtime of 0.219±0.006 seconds at an arrival rate of 4 to 20 VNRs per 100-time units at resource request rate of 0.227±0.007 seconds, and 0.476±0.008 seconds, at node size expansion from 2 to 32 nodes, the proposed solution showed a consistent improvements in the scalability and runtime stability under different datacenter network complexities.

Table 3. The average runtime in a second

Test type	Test details	Average runtime (seconds)
Arrival rate	Varying arrival rates from 4 to 20 VNRs per 100-time units	0.650±0.014
Resource request	Varying CPU and bandwidth demand from 0 to 30 units	0.750±0.021
Node size expansion	Increasing virtual node size from 2 to 32 nodes	0.850±0.017

4.2. Resource utilization

Figure 7 illustrates the resource utilization achieved by the proposed algorithm. These results have been compared with the results of lately published research such as in [21], [22], [24], where they achieved 0.796±0.001, 0.800±0.001, and 0.800±0.001 node utilization, respectively, and 0.750±0.001, 0.850±0.001, and 0.400±0.001 link utilization, respectively, as summarized in Figure 7. Taking all together, these comparative insights highlight the efficiency of the proposed algorithm, confirm that the proposed solution outperforms the existing solutions in terms of resource node utilization, and the link utilization was comparable to [24] 0.800±0.001, with no statistically significant difference $p \approx 0.98$. These findings confirm

that the proposed system improves both node and link utilization, offering significant gains across most baselines while performing competitively with [24] in link utilization.

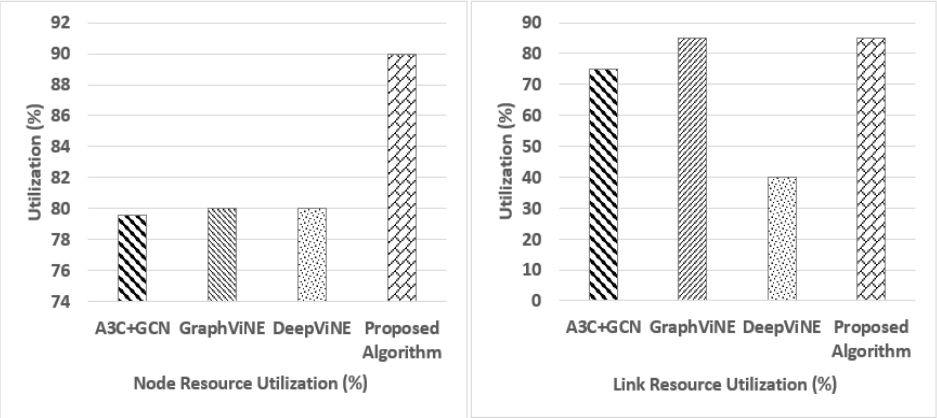

Figure 7. Nodes and links utilization results of the proposed algorithm compared with that in [21], [22], [24]

4.3. Load balancing and system stability

The proposed algorithm achieved a node resource balance score exceeding 0.905±0.004 while it succeeded in maintaining link load imbalance ratio of 0.250±0.003. The comparison with [21] as summarized in Table 4 shows that the load imbalance ratio efficiency of the proposed algorithm exceeded that in [21] by approximately 0.23 where they attained a balance score of 0.670±0.003. Meanwhile, the comparison also showed that the proposed algorithm successfully kept a lower link load imbalance ratio by approximately 0.13 compared with that in [21] where they demonstrated a higher imbalance ratio of 0.381±0.002.

Table 4. Comparison of node resource balance score and link imbalance ratio with baseline [21]

Metric	Proposed hidden algorithm	A3C+GCV
Node resource balance score	>0.905±0.004	0.670±0.003
Link load imbalance ratio	0.250±0.003	0.381±0.002

4.4. Energy efficiency and computational overhead reduction

A set of simulation scenarios have been conducted to evaluate the energy consumption of the VNE process under different VNR sizes. The simulation results showed that the proposed algorithm reduced energy consumption by 24.90±0.01, 24.80±0.01, and 23.90±0.01 for small, medium, and large VNRs, respectively, compared to that attained when the simulation is performed using the baseline. Table 5 highlights the reduction of energy consumption using the proposed hidden inference algorithm compared with that presented in [28], where they showed a reduction in energy consumption 24.80±0.01, 24.80±0.01, and 23.90±0.01 for small, medium, and large VNRs, respectively, compared to the baseline. However, the comparative results in Table 5 show a slight improvement of the proposed algorithm over that in [28], the proposed algorithm demonstrates better resource orchestration, consistency, sustainability, stability, and lower variance during the embedding process across varying VNR sizes, compared with the energy efficient algorithm in [28] that shows notable fluctuations and higher computational overheads.

Table 5. Energy consumption comparison with that in [28]

Algorithm	Small-sized VNRs	Medium-sized VNRs	Large-sized VNRs
Proposed hidden inference algorithm	24.90±0.01	24.80±0.01	23.90±0.01
Energy-efficient algorithm [28]	24.80±0.01	24.80±0.01	23.90±0.01

4.5. Acceptance ratio for virtual network requests

The last set of simulation scenarios have been devoted to evaluating the model acceptance ratio over time under different and varying VNR sizes and conditions. Figure 8 illustrates this result compared with

those reported in leading publications [21]–[24]. In Figure 8, the acceptance ratio was measured over 1200 simulation unit time to show the necessary time for reaching the maximum and stable acceptance ratio compared with that published in [21]–[24]. Comparative analysis shows that the proposed model achieved an acceptance ratio of approximately up to 0.9365±0.0092 after 900-time units, while those in [21]–[24] achieved a maximum acceptance ratio of approximately 0.8481±0.0086, 0.8963±0.0073, 0.8693±0.0153, and 0.8248±0.0087 after 1100-time units.

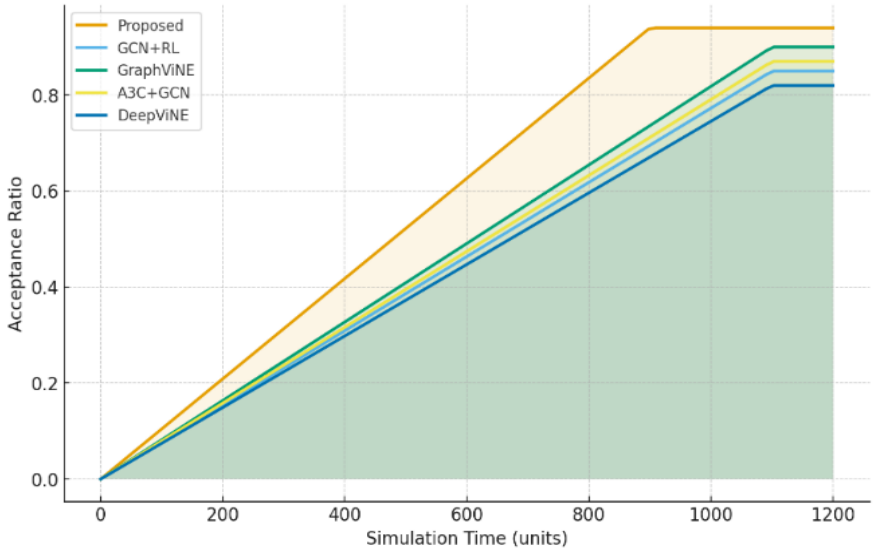

Figure 8. Comparison of acceptance ratio with those reported in leading publications [21]–[24]

Analyzing the solutions in [21]–[24] that yielded the acceptance ratio results illustrated in Figure 8, indicates that in [23], the objective function based on a self-defined fitness matrix, requires significant computation time which is the reason for the large convergence time. However, Habibi *et al.* [24] achieved a maximum acceptance ratio of 89.6%. It was noticed that their solution requires longer convergence time especially under high-demand conditions, likely due to its less adaptive resource allocation strategy. Also, the solution presented in [21] demonstrates lower efficiency and long convergence time for embedding large VNRs. At last, the performance degradation and long convergence time in [22], most probably due to the deployment of automated feature extraction through DRL and static topology assumptions. As shown in Figure 8, the proposed method achieves higher acceptance than baselines including [23], [24].

5. CONCLUSION

This paper presents a modified VNE framework that is based on a novel hidden inference algorithm and a KG representation of the substrate network and is able to locate not only directly connected resources but also infer hidden connections and identify indirectly connected resources which sets a new benchmark for VNE performance for 6G datacenters. The proposed framework achieved execution times ranging from 0.65 to 0.85 seconds, showing consistent runtime stability under different VNR sizes. The framework also achieved 90% of the node and 85% link utilization, while it successfully achieved a node balance score exceeding 0.90 and a link imbalance ratio of 0.25. Furthermore, the framework achieved stable energy reductions up to 24.9% under different VNR sizes. Most notably, the proposed framework demonstrated significant improvement of the VNR acceptance ratio reaching approximately 94% after 900-time units. Future research will focus on enhancing the KG's semantic scope and integrating predictive analytics for a variety of practical 6G datacenters.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Shourok Abdelrahim	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓
Samy Ghoniemy	✓	✓				✓		✓	✓	✓	✓	✓		
Mohamed Aborizka	✓		✓	✓			✓			✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are openly available in GitHub at https://github.com/shourok/Datacenter_Dataset.git.

REFERENCES

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020, doi: 10.1109/MNET.001.1900287.

[2] T. S. Rappaport *et al.*, "Wireless communications and applications above 100 GHz: Opportunities and challenges for 6G and beyond," *IEEE Access*, vol. 7, pp. 78729–78757, 2019, doi: 10.1109/ACCESS.2019.2921522.

[3] S. Cherrared, S. Imadali, E. Fabre, G. Gossler, and I. G. B. Yahia, "A survey of fault management in network virtualization environments: challenges and solutions," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1537–1551, 2019, doi: 10.1109/TNSM.2019.2948420.

[4] H. Cao, H. Zhu, and L. Yang, "Collaborative attributes and resources for single-stage virtual network mapping in network virtualization," *Journal of Communications and Networks*, vol. 22, no. 1, pp. 61–71, 2020, doi: 10.1109/JCN.2019.000045.

[5] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Towards 6G networks: use cases and technologies," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, 2020, doi: 10.1109/MCOM.001.1900411.

[6] S. Wu, N. Chen, A. Xiao, P. Zhang, C. Jiang, and W. Zhang, "AI-empowered virtual network embedding: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 2, pp. 1395–1426, 2025, doi: 10.1109/COMST.2024.3424533.

[7] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, "Artificial intelligence-enabled cellular networks: a critical path to beyond-5G and 6G," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 212–217, 2020, doi: 10.1109/MWC.001.1900323.

[8] N. Chen, S. Shen, Y. Duan, S. Huang, W. Zhang, and L. Tan, "Non-Euclidean graph-convolution virtual network embedding for space–air–ground integrated networks," *Drones*, vol. 7, no. 3, 2023, doi: 10.3390/drones7030165.

[9] S. Zhang, C. Wang, J. Zhang, Y. Duan, X. You, and P. Zhang, "Network resource allocation strategy based on deep reinforcement learning," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 86–94, 2020, doi: 10.1109/OJCS.2020.3000330.

[10] X. Deng, J. Sun, and J. Lu, "Graph neural network-based efficient subgraph embedding method for link prediction in mobile edge computing," *Sensors*, vol. 23, no. 10, 2023, doi: 10.3390/s23104936.

[11] G. Kibalya, J. Serrat, J.-L. Gorricho, H. Yao, and P. Zhang, "A novel dynamic programming inspired algorithm for embedding of virtual networks in future networks," *Computer Networks*, vol. 179, 2020, doi: 10.1016/j.comnet.2020.107349.

[12] M. Lu, Y. Gu, and D. Xie, "A dynamic and collaborative multi-layer virtual network embedding algorithm in SDN based on reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2305–2317, 2020, doi: 10.1109/TNSM.2020.3012588.

[13] X. Xiao, "DVNE-DRL: dynamic virtual network embedding algorithm based on deep reinforcement learning," *Scientific Reports*, vol. 13, no. 1, 2023, doi: 10.1038/s41598-023-47195-5.

[14] P. Zhang, C. Wang, C. Jiang, and A. Benslimane, "Security-aware virtual network embedding algorithm based on reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1095–1105, 2021, doi: 10.1109/TNSE.2020.2995863.

[15] P. Zhang, N. Chen, S. Li, K.-K. R. Choo, C. Jiang, and S. Wu, "Multi-domain virtual network embedding algorithm based on horizontal federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3363–3375, 2023, doi: 10.1109/TIFS.2023.3279587.

[16] Y. Li, J. Huang, Q. Sun, T. Sun, and S. Wang, "Cognitive service architecture for 6G core network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7193–7203, 2021, doi: 10.1109/TII.2021.3063697.

[17] K. Mitropoulou, P. Kokkinos, P. Soumplis, and E. Varvarigos, "Anomaly detection in cloud computing using knowledge graph embedding and machine learning mechanisms," *Journal of Grid Computing*, vol. 22, no. 1, 2024, doi: 10.1007/s10723-023-09727-1.

[18] A. Mitropoulou, P. Kokkinos, and E. Varvarigos, "Identifying network congestion using knowledge graphs and link prediction," in *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, 2023, pp. 1–6, doi: 10.1145/3603166.3632129.

[19] D. Yu, Y. Yang, R. Zhang, and Y. Wu, "Knowledge embedding based graph convolutional network," in *Proceedings of the Web Conference 2021*, 2021, pp. 1619–1628, doi: 10.1145/3442381.3449925.
[20] A. Hashmi and C. Gupta, "VNE-NR: a node-ranking method for performing topology-aware and resource-driven virtual network embedding," in *2020 11th International Conference on Computing, Communication and Networking Technologies*, 2020, pp. 1–6, doi: 10.1109/ICCCNT49239.2020.9225533.
[21] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: a deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020, doi: 10.1109/JSAC.2020.2986662.
[22] M. Dolati, S. B. Hassanpour, M. Ghaderi, and A. Khonsari, "DeepViNE: virtual network embedding with deep reinforcement learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops*, 2019, pp. 879–885, doi: 10.1109/INFCOMW.2019.8845171.
[23] P. Zhang, C. Wang, N. Kumar, W. Zhang, and L. Liu, "Dynamic virtual network embedding algorithm based on graph convolution neural network and reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9389–9398, 2022, doi: 10.1109/JIOT.2021.3095094.
[24] F. Habibi, M. Dolati, A. Khonsari, and M. Ghaderi, "Accelerating virtual network embedding with graph neural networks," in *2020 16th International Conference on Network and Service Management*, 2020, pp. 1–9, doi: 10.23919/CNSM50824.2020.9269128.
[25] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-56706-x.
[26] K. K. Sharma and M. Sood, "Mininet as a container-based emulator for software defined networks," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 12, pp. 681–685, 2014.
[27] P. Berde *et al.*, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, 2014, pp. 1–6, doi: 10.1145/2620728.2620744.
[28] P. Zhang, E. Wang, Z. Luo, Y. Bi, K. Liu, and J. Wang, "Energy-efficient virtual network embedding: a deep reinforcement learning approach based on graph convolutional networks," *Electronics*, vol. 13, no. 10, 2024, doi: 10.3390/electronics13101918.

BIOGRAPHIES OF AUTHORS

Shourok Abdelrahim is an assistant lecturer at Canadian International College in Cairo. She received her master's degree in Computer Science from Arab Academy for Science and Technology, Cairo, Egypt in 2019. She studied Computer Science in Misr University for Science and Technology, Cairo, Egypt in 2011. She published a paper in ACM. She is working on her master's thesis in network security and steganograph. She can be contacted at email: s.mohamed58@student.aast.edu or shorouk_abdelrahim@cic-cairo.com.

Prof. Samy Ghoniemy is a full professor of Computer Systems, ICT consultant, and AI strategist with over 33 years of academic, research, and industrial experience. He earned his Ph.D. in Computer Engineering from Carleton University, Canada, and has since authored over 100 peer-reviewed publications and 8 books across networks, AI, medical informatics, and HPC systems. His recent research explores 6G, cloud data centers, cybersecurity and intelligent healthcare systems, with notable national and international collaborations. He is frequently invited as a keynote speaker at international forums. He is a senior member of IEEE, ACM, and OSA, and holds patents in cloud document security and telecommunications systems. He can be contacted at email: samy.ghoniemy@gu.edu.eg.

Prof. Mohamed Aborizka is a professor of Artificial Intelligence and dean of the College of Computing and Information Technology at Arab Academy for Science, Technology & Maritime Transport, with over 30 years of experience in the ICT sector. He has published more than 80 papers in leading journals and conferences across AI, machine learning, NLP, blockchain, cybersecurity, and eLearning. He has supervised over 50 Ph.D. and master's theses and held senior academic leadership roles throughout Egypt. His contributions to education and technology have been recognized with the shield of excellence from Egypt's Higher Education Supreme Council. He has also been honored with the prestigious deputy medal by a former President of Egypt. He can be contacted at email: m.aborizka@aast.edu.