Rule-Aware VNE for SDN Resource Management

Amir Javadpour, Forough Ja'fari, Tarik Taleb, and Masoud Shokrnezhad

***Abstract*—Software-defined networking (SDN) provides a programmable substrate for network virtualization, slicing, and edge–cloud orchestration. Virtual network embedding (VNE), however, consumes not only node and bandwidth resources but also switch flow-table entries and controller operations whenever forwarding paths are installed or migrated. This paper presents Time-Window-Based Efficient Initial Mapping for Software-Defined Networks (TW-EIMSDN), an online VNE framework that separates immediate admission and initial embedding from deferred, gain-thresholded rule rewriting. The formulation jointly represents admission, node placement, path selection, bandwidth, delay, flow-table capacity, residual-resource balance, and rule-update cost. A controller-side heuristic uses residual-resource-aware node scoring, rule-pressure-aware path scoring, and selective make-before-break migration at time-window or eligible resource-release events. In an event-driven evaluation comprising 1,000 requests and 30 paired random seeds on 50-node/150-link substrates, TW-EIMSDN achieves an acceptance ratio of 0.884 $\pm$ 0.011 compared with 0.870 $\pm$ 0.011 for the no-selective-rewrite ablation, a paired gain of 1.41 percentage points. Relative to an aggressive dynamic-rewrite baseline, it reduces total rule-update operations by 74.3% while incurring only 1.49 percentage points lower acceptance. Sensitivity, scalability, and candidate-path analyses expose the trade-off among responsiveness, forwarding-state stability, solution quality, and runtime. The results support TW-EIMSDN as a controlled middle ground between static embedding and indiscriminate remapping rather than as a universally dominant optimizer.**

***Index Terms*—software-defined networking, virtual network embedding, online resource allocation, flow-table occupancy, rule rewriting, time-window scheduling.**

I. Introduction

Network softwarization is a key principle behind modern programmable infrastructures, including cloud data centers, edge systems, 5G/6G slicing environments, and service-oriented transport networks. SDN separates the control plane from the data plane and allows a logically centralized controller to install forwarding behavior through flow rules. Together with network function virtualization (NFV), SDN enables multiple logical services to share a common physical infrastructure while preserving different performance and service-level requirements [1], [2]. In this setting, a service can be represented as a VNR, whose virtual nodes require computation, switching, or service-processing resources, and whose virtual links require bandwidth and delay support.

Mapping VNRs onto a shared physical infrastructure is the VNE problem. A VNE policy must decide whether to accept an arriving VNR, where to place its virtual nodes, and which substrate paths should carry its virtual links. The problem remains computationally difficult, and it becomes more challenging in online settings because VNRs arrive sequentially, remain active for finite lifetimes, and release resources at different times [3]. Recent work has improved VNE through coordinated node-link mapping, mixed-integer optimization, graph heuristics, meta-heuristics, and deep reinforcement learning (DRL). Learning-based methods are especially attractive because they can exploit temporal and topological information in dynamic networks [4]–[8]. Nevertheless, most VNE models still focus primarily on acceptance rate, revenue, and resource cost. They rarely expose the operational cost of installing and rewriting SDN flow rules as a first-class optimization term.

This omission matters in practical SDN deployment. A virtual link embedded on a substrate path is implemented through rules installed in the switches along that path. When a virtual link is migrated, the controller must compute a new path, install new rules, redirect traffic consistently, and remove obsolete rules. Flow-table capacity is limited, often because high-speed match-action tables rely on expensive memory such as TCAM, and frequent rule updates can increase control-plane delay or temporary inconsistency [9]–[11]. Therefore, an embedding policy that looks efficient at the abstract graph level can still be expensive at the SDN implementation level if it repeatedly rewrites paths and consumes switch-table resources.

Time-Window-Based Efficient Initial Mapping for Software-Defined Networks (TW-EIMSDN) is motivated by the need to make VNE decisions aware of SDN rule-update overhead. Instead of remapping the entire network after every request or resource change, the controller performs an effective initial mapping and postpones unnecessary rule rewriting. The central design principle is to separate *admission*, *initial embedding*, and *selective rewriting*. A new VNR is processed online and receives an initial feasible embedding if node, link, delay, and rule-table constraints are satisfied. Existing virtual links are not migrated merely because a shorter path becomes available. They are reconsidered only at time-window boundaries or after eligible resource-release events, and only if the migration produces positive net gain after accounting for rule-update cost.

Unlike conventional VNE formulations that optimize logical node and link resources and then assume that SDN forwarding updates are mechanically applied, TW-EIMSDN embeds rule-table occupancy and rewrite cost directly into both initial path selection and delayed migration decisions. This distinction is important because two embeddings with similar bandwidth consumption may impose very different operational burdens on the SDN controller and switch flow tables. Therefore, the proposed framework treats forwarding-state management as part of the embedding decision rather than as a post-processing

Amir Javadpour and Masoud Shokrnezhad are with MOSA!C Lab / ICTFICIAL Oy, Finland.
Forough Ja'fari is with Sharif University of Technology, Iran.
Tarik Taleb is with Ruhr University Bochum, Germany.
Corresponding author: Amir Javadpour (e-mail: a.javadpour87@gmail.com).

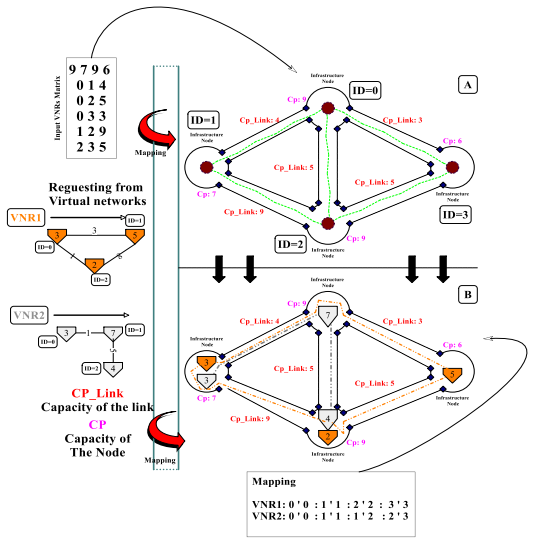


Fig. 1. Rule-rewrite-aware virtual mapping.

step after virtual-link routing.

Fig. 1 illustrates this rule-aware view. Conventional dynamic mapping may trigger broad remapping after each arrival or topology/resource update. TW-EIMSDN instead stores the active mapping state in the controller database and updates only affected rules. This design is particularly useful when VNR lifetimes are short or highly heterogeneous, because rewriting a soon-to-expire virtual link may consume more controller and switch resources than it saves.

The main contributions of this paper are summarized as follows.

- We propose TW-EIMSDN, a time-window- and rule-rewrite-aware online VNE framework for SDN-enabled virtualized networks that extends beyond initial node-link mapping by maintaining switch-rule occupancy as controller state.
- We provide a mathematical model that jointly captures admission, node mapping, path-based virtual-link embedding, bandwidth feasibility, delay constraints, switch flow-table capacity, embedding cost, residual-resource imbalance, and rule-rewrite cost.
- We design a controller-side algorithm that performs residual-resource-aware initial embedding and enables delayed selective rewriting only when an explicit net-gain function (resource benefit minus rule-addition and rule-deletion cost) exceeds a stability threshold.
- We evaluate acceptance, embedding cost, link and switch utilization, accepted migrations, rule additions/deletions, skipped low-gain rewrites, confidence intervals, parameter sensitivity, scalability, controller runtime, and the candidate-path trade-off.

The remainder of the paper is organized as follows. Section II reviews related work and positions the contribution. Section III presents the system model and optimization formulation. Section IV describes the TW-EIMSDN controller workflow. Section V gives the algorithms and complexity discussion. Section VI reports the simulation setting and result interpretation. Section VIII concludes the paper.

II. Related Work and Positioning

A. Recent VNE and NFV Resource Allocation

Recent VNE research has moved from isolated node-link mapping toward online resource allocation under admission, scalability, and generalization constraints. The 2025 VNE survey by Satpathy *et al.* consolidates recent advances and emphasizes that VNE remains central to efficient virtual-network operation under heterogeneous substrate resources [3]. In parallel, benchmark-oriented studies argue that NFV resource allocation needs reproducible simulators, unified baselines, and evaluation dimensions beyond a single effectiveness metric, including scalability and generalization. These recent studies, together with emerging reproducible benchmarking environments [12], motivate a clearer separation between algorithmic mapping quality and deployment-level overhead, which is the main concern of TW-EIMSDN.

B. AI-Driven and Learning-Based Embedding

Learning-based VNE has become a dominant direction because online embedding decisions have long-term effects on acceptance, cost, and residual-resource fragmentation. DVNE-DRL models dynamic topology and resource changes through deep reinforcement learning [13]; PPO-VNE coordinates virtual-node and virtual-link embedding through proximal policy optimization and hybrid feature extraction [5]; HRL-ACRA explicitly combines admission control and resource allocation in a hierarchical policy [6]; and recent edge-oriented RL formulations optimize cost and acceptance for 5G/edge environments [7]. Energy-aware studies further integrate graph convolutional representations and traffic variation into DRL-based embedding decisions [8]. These works improve decision intelligence, but they usually evaluate VNE at the logical resource layer. TW-EIMSDN is complementary: it targets the SDN execution layer by penalizing avoidable path rewrites and switch-rule pressure.

C. Network Slicing, Life-Cycle Automation, and SDN/NFV

Network slicing has expanded the VNE problem into an end-to-end orchestration problem spanning access, transport, core, cloud, and edge domains. Recent surveys show that single-domain optimization is insufficient for 5G/6G slicing, because slices evolve across preparation, commissioning, operation, scaling, and termination phases [2]. In these life-cycle phases, admission control, resource instantiation, reconfiguration, and decommissioning are all relevant to VNE-like decisions. TW-EIMSDN focuses on a narrower but operationally important part of this life cycle: how an SDN controller should install and update forwarding state when VNRs arrive, expire, or release resources. This focus is relevant because a technically feasible embedding can still be undesirable when it triggers excessive switch-table churn.

D. SDN Rule-Table and Flow-Update Management

The SDN literature increasingly treats flow-rule updates as a resource-management problem rather than a purely mechanical

TABLE I
COMPACT POSITIONING OF TW-EIMSDN.

Stream	Main focus	Position of TW-EIMSDN
VNE surveys and benchmarks [3]	Taxonomy, reproducibility, scalability	They clarify the VNE landscape, while TW-EIMSDN adds a controller-side policy for reducing SDN rule rewrites during online embedding.
AI/DRL-based VNE [5]–[8], [13]	Learned admission, mapping, cost, and energy optimization	These methods improve mapping decisions, but usually do not model rule-table pressure and rewrite churn as explicit costs.
5G/6G slicing automation [2]	Slice life-cycle management and AI-based orchestration	They motivate automation, while TW-EIMSDN focuses on substrate-level SDN rule-update cost in VNE.
SDN rule management [9]–[11], [14]	Rule installation, flow updates, TCAM usage, and controller overhead	These works study rule operations directly, but are not tightly integrated with online VNR admission and virtual-link embedding.
Benchmark baselines [15]	Online VNE, time-window adjustment, and resource usage	TW-EIMSDN keeps the same comparison context, but adds explicit rewrite cost, rule-table pressure, and delayed selective migration.
TW-EIMSDN	Rule-aware online VNE	Separates admission, initial embedding, and selective time-window rewriting to balance acceptance, cost, link usage, and switch-resource usage.

controller action. Flow-table update operations can be expensive in TCAM-based switches, and dependency-aware updates can become algorithmically complex [9]. Reactive and proactive rule-installation schemes also introduce different trade-offs among setup delay, table occupancy, and policy-update overhead [10]. Recent studies on flow-service optimization and mobility-aware flow placement further show that the placement and timing of rules affect service delay and network cost [11], [14]. TW-EIMSDN therefore treats rule rewriting as an explicit cost term and uses time windows to avoid low-benefit migrations.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. Substrate Network and VNR Model

The SDN-enabled substrate network is represented by an undirected graph

$$G^s = (N^s, E^s), \tag{1}$$

where N^s and E^s denote substrate nodes and links. Each substrate node $u \in N^s$ has node capacity C_u, residual node capacity $C_u^{\mathrm{res}}(t)$, and maximum flow-table capacity F_u. Each substrate link $(u, v) \in E^s$ has bandwidth capacity B_{uv}, residual bandwidth $B_{uv}^{\mathrm{res}}(t)$, propagation or forwarding delay D_{uv}, and unit bandwidth cost c_{uv}.

A VNR k is represented by

$$G_k^v = (N_k^v, E_k^v), \tag{2}$$

where N_k^v and E_k^v are the virtual-node and virtual-link sets. Each virtual node $i \in N_k^v$ requires node resource d_i^k. Each virtual link $e = (i, j) \in E_k^v$ requires bandwidth b_e^k and may have an end-to-end delay bound $\bar{D}_e^k$. The request arrives at time A_k, remains active for lifetime τ_k, and expires at $A_k + \tau_k$. VNRs are admitted immediately upon arrival; the time window of length Δ is used only to schedule deferred rewrite evaluation.

TABLE II
NOTATION USED IN THE PROPOSED MODEL.

Symbol	Meaning
G^s	SDN-enabled substrate graph
N^s, E^s	Substrate nodes and links
C_u, F_u	Node capacity and flow-table capacity of substrate node u
B_{uv}, D_{uv}	Bandwidth capacity and delay of substrate link (u, v)
G_k^v	Virtual network request k
d_i^k, b_e^k	Node demand and virtual-link bandwidth demand
z_k	Binary admission variable of VNR k
$\mathcal{K}(t)$	Active VNRs plus the arriving request under temporary evaluation
$\mathcal{M}(t)$	Virtual-link migrations accepted at decision epoch t
A_k	Arrival time of VNR k
$x_{i,u}^k$	Binary variable for mapping virtual node i to substrate node u
$y_{e,p}^k$	Binary variable for mapping virtual link e to substrate path p
$\mathcal{R}_u(t)$	Number of active flow rules in switch u
Δ	Controller-side time-window length

B. Admission and Node Mapping

At decision epoch t, let $\mathcal{K}(t) = \mathcal{A}(t) \cup \{k_{\mathrm{arr}}\}$ denote the currently active VNRs together with the arriving request under temporary evaluation. When no arrival is being evaluated, $\mathcal{K}(t) = \mathcal{A}(t)$. All capacity and cost expressions below are evaluated over $\mathcal{K}(t)$ so that expired VNRs no longer consume substrate resources.

Let $z_k \in \{0, 1\}$ denote whether VNR k is accepted. The node-mapping variable is

$$x_{i,u}^k = \begin{cases} 1, & \text{if virtual node } i \in N_k^v \text{ is mapped to } u \in N^s, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

If VNR k is accepted, each virtual node must be mapped to exactly one substrate node; otherwise, no node is mapped:

$$\sum_{u \in N^s} x_{i,u}^k = z_k, \quad \forall i \in N_k^v, \ \forall k \in \mathcal{K}(t). \tag{4}$$

In the reported implementation, distinct virtual nodes belonging to the same VNR are not colocated on one substrate node:

$$\sum_{i \in N_k^v} x_{i,u}^k \le 1, \quad \forall u \in N^s, \ \forall k \in \mathcal{K}(t). \tag{5}$$

The node-capacity constraint is

$$\sum_{k \in \mathcal{K}(t)} \sum_{i \in N_k^v} d_i^k x_{i,u}^k \le C_u, \quad \forall u \in N^s. \tag{6}$$

If virtual node i can only be placed in a feasible substrate set $\Omega_i^k \subseteq N^s$, the location or policy constraint is

$$x_{i,u}^k = 0, \quad \forall u \notin \Omega_i^k. \tag{7}$$

C. Path-Based Virtual-Link Mapping

Let $\mathcal{P}$ be the set of candidate simple substrate paths. For path $p \in \mathcal{P}$, $s(p)$ and $t(p)$ denote its start and end nodes, δ_{uv}^p indicates whether substrate link (u, v) belongs to p, and η_u^p indicates whether switch u needs a forwarding rule for p. The path-selection variable for virtual link $e = (i, j)$ is

$$y_{e,p}^k = \begin{cases} 1, & \text{if virtual link } e \text{ of VNR } k \text{ is routed on } p, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Each virtual link of an accepted VNR must select exactly one substrate path:

$$\sum_{p\in\mathcal{P}} y_{e,p}^k = z_k, \quad \forall e \in E_k^v,\ \forall k \in \mathcal{K}(t). \tag{9}$$

Path endpoints must be consistent with the node mapping. For $e=(i,j)$, this can be enforced by

$$y_{e,p}^k \le x_{i,s(p)}^k, \quad y_{e,p}^k \le x_{j,t(p)}^k, \tag{10}$$

for all $k \in \mathcal{K}(t)$, $e=(i,j) \in E_k^v$, and $p \in \mathcal{P}$. If the substrate graph is treated as undirected, both orientations of each candidate path are included in $\mathcal{P}$.

The link-capacity constraint is

$$\sum_{k\in\mathcal{K}(t)} \sum_{e\in E_k^v} \sum_{p\in\mathcal{P}} b_e^k \delta_{uv}^p y_{e,p}^k \le B_{uv}, \quad \forall (u,v) \in E^s. \tag{11}$$

The delay constraint is

$$\sum_{p\in\mathcal{P}} \left(\sum_{(u,v)\in E^s} D_{uv}\delta_{uv}^p \right) y_{e,p}^k \le \bar{D}_e^k z_k, \tag{12}$$

for all $e \in E_k^v$ and $k \in \mathcal{K}(t)$.

D. Switch Rule-Table Constraint

The number of active rules in switch u at time t is modeled as

$$\mathcal{R}_u(t) = \sum_{k\in\mathcal{K}(t)} \sum_{e\in E_k^v} \sum_{p\in\mathcal{P}} \eta_u^p y_{e,p}^k. \tag{13}$$

Equation (13) follows a conservative per-VNR/per-virtual-link rule accounting model. In other words, each embedded virtual link is assumed to require forwarding state along its selected substrate path. This assumption does not rely on rule aggregation or wildcard compression. If a switch can aggregate multiple forwarding entries in practice, the actual table usage may be lower; however, the proposed model remains useful because it prevents the embedding algorithm from selecting paths that are feasible in bandwidth but risky in terms of flow-table pressure. The flow-table feasibility condition is

$$\mathcal{R}_u(t) \le F_u, \quad \forall u \in N^s. \tag{14}$$

This constraint is important because two paths with the same bandwidth cost may differ substantially in their rule-table footprint.

E. Cost Model and Optimization Objective

The embedding cost of accepted VNRs active at decision epoch t is

$$C_{\mathrm{emb}}(t) = \sum_{k\in\mathcal{K}(t)} \sum_{i\in N_k^v} \sum_{u\in N^s} c_u d_i^k x_{i,u}^k + \sum_{k\in\mathcal{K}(t)} \sum_{e\in E_k^v} \sum_{p\in\mathcal{P}} c_p b_e^k y_{e,p}^k, \tag{15}$$

where $c_p = \sum_{(u,v)\in E^s} c_{uv}\delta_{uv}^p$.

If virtual link e of VNR k is migrated from old path p_o to new path p_n, its rule-rewrite cost is

$$C_{\mathrm{rw}}^{k,e}(p_o,p_n) = \mu_{\mathrm{add}} \left|\mathcal{R}(p_n) \setminus \mathcal{R}(p_o)\right| + \mu_{\mathrm{del}} \left|\mathcal{R}(p_o) \setminus \mathcal{R}(p_n)\right|, \tag{16}$$

where μ_{add} and μ_{del} are the costs of adding and deleting switch rules. Here, $\mathcal{R}(p)$ denotes the set of switch-level forwarding entries required to realize path p. Therefore, the set difference in (16) captures only the rules that must be newly installed or removed during migration. If the old and new paths share some switches or forwarding entries, those common rules do not contribute to the rewrite cost. This definition encourages path migration only when the new route provides a meaningful resource benefit beyond the operational cost of updating the forwarding state. Let $\mathcal{M}(t)$ denote the set of virtual-link migrations accepted at decision epoch t. The aggregate rewrite cost is

$$C_{\mathrm{rw}}(t) = \sum_{(k,e)\in\mathcal{M}(t)} C_{\mathrm{rw}}^{k,e}\left(p_o^{k,e}, p_n^{k,e}\right). \tag{17}$$

To discourage residual-resource fragmentation, we define the link-load imbalance as

$$C_{\mathrm{imb}}(t) = \sum_{(u,v)\in E^s} \left(\ell_{uv}(t) - \bar{\ell}(t)\right)^2, \tag{18}$$

where

$$\ell_{uv}(t) = \frac{B_{uv} - B_{uv}^{\mathrm{res}}(t)}{B_{uv}} \tag{19}$$

and $\bar{\ell}(t)$ is the average substrate-link load ratio. The controller-level objective at a decision epoch is

$$\max \sum_{k\in\mathcal{K}(t)} r_k z_k - \alpha C_{\mathrm{emb}}(t) - \beta C_{\mathrm{rw}}(t) - \gamma C_{\mathrm{imb}}(t) - \zeta \sum_{u\in N^s} \frac{\mathcal{R}_u(t)}{F_u}, \tag{20}$$

subject to (4)–(14). Here, r_k is the revenue of accepting VNR k, while α, β, γ, and ζ tune the relative importance of embedding cost, rewrite cost, residual-resource balance, and flow-table pressure.

IV. Proposed TW-EIMSDN Framework

TW-EIMSDN is implemented as a controller-side resource-management module. The controller maintains four state components: residual substrate resources, active VNR mappings, switch flow-table occupancy, and a priority queue of virtual links that may benefit from future rewriting. The framework operates at two time scales. At the fast scale, arriving VNRs are processed online and activated immediately if feasible. At the slow scale, a time-window mechanism triggers selective rewriting at controlled epochs.

A. Controller State

At time t, the controller state is

$$S(t) = \{C^{\mathrm{res}}(t), B^{\mathrm{res}}(t), \mathcal{R}(t), \mathcal{A}(t), \mathcal{Q}(t)\}. \tag{21}$$

The active set $\mathcal{A}(t)$ stores all accepted VNRs and their remaining lifetimes. The priority queue $\mathcal{Q}(t)$ stores active virtual links ranked by their expected rewriting benefit. The state is updated after VNR arrivals, VNR expirations, rule installations, and rule deletions.

B. Initial Mapping Score

For a newly arriving VNR, the controller first computes a feasible node mapping. A substrate node is attractive when it has high residual node capacity and is adjacent to links with sufficient remaining bandwidth. The residual-resource score of substrate node u is

$$\Psi_u(t) = \lambda_c \frac{C_u^{\mathrm{res}}(t)}{C_u} + \lambda_b \frac{\sum_{v:(u,v)\in E^s} B_{uv}^{\mathrm{res}}(t)}{\sum_{v:(u,v)\in E^s} B_{uv}} - \lambda_f \frac{\mathcal{R}_u(t)}{F_u}. \tag{22}$$

where λ_c, λ_b, and λ_f balance node capacity, adjacent-link bandwidth, and rule-table pressure. Virtual nodes are sorted by decreasing demand and incident bandwidth, then mapped to feasible substrate nodes with high $\Psi_u(t)$.

After node mapping, each virtual link is routed over a candidate path satisfying bandwidth, delay, and rule-table constraints. For virtual link e and candidate path p, the path score is

$$\Phi_e^k(p) = \omega_1 c_p b_e^k + \omega_2 h(p) + \omega_3 \sum_{(u,v)\in p} \frac{b_e^k}{B_{uv}^{\mathrm{res}}(t)+\epsilon} + \omega_4 \sum_{u\in p} \frac{\mathcal{R}_u(t)}{F_u}. \tag{23}$$

where $h(p)$ is the number of switches requiring rule installation and ϵ is a small constant preventing division by zero. The first term captures bandwidth cost, the second term approximates path rule footprint, the third term penalizes bottleneck links, and the fourth term penalizes switches with high rule-table pressure.

C. Time-Window and Release-Triggered Rewriting

At the end of each time window, the controller evaluates whether selected active virtual links should be migrated. A VNR expiration can trigger an additional assessment only when the maximum substrate-link utilization is at least 0.70 and at least $\Delta/2$ time units have elapsed since the previous assessment. This guard prevents repeated low-value evaluations after closely spaced release events. For active virtual link e of VNR k currently routed on p_o, the footprint is

$$\Theta_e^k(p_o) = b_e^k |p_o| + \chi h(p_o), \tag{24}$$

where $|p_o|$ is the path length and χ converts rule usage into a comparable resource term. The bottleneck pressure is

$$\Pi_e^k(p_o) = \max_{(u,v)\in p_o} \ell_{uv}(t). \tag{25}$$

The remaining-lifetime factor is defined as

$$L_k(t) = \frac{\max\{A_k + \tau_k - t, 0\}}{\tau_k}. \tag{26}$$

A link with very short remaining lifetime should have low rewrite priority. Thus, the priority score is

$$\Omega_e^k = \lambda_1 \Theta_e^k(p_o) + \lambda_2 \Pi_e^k(p_o) + \lambda_3 \sum_{u\in p_o} \frac{\mathcal{R}_u(t)}{F_u} + \lambda_4 L_k(t). \tag{27}$$

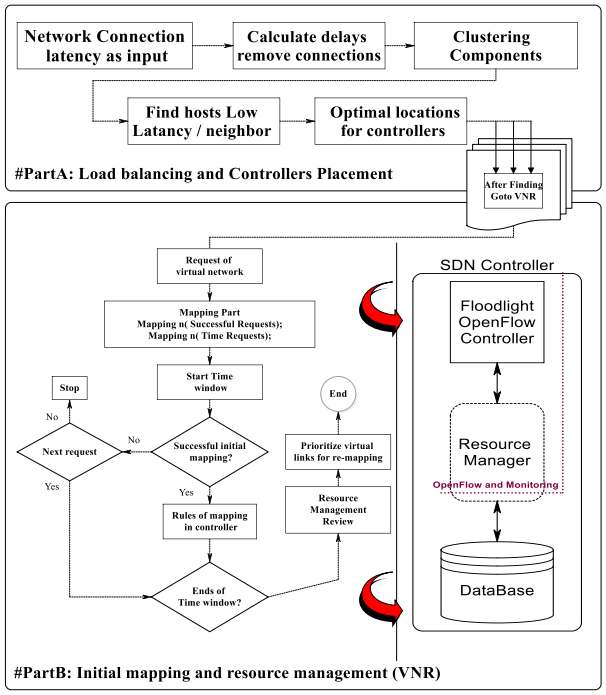


Fig. 2. Controller-side workflow of TW-EIMSDN. New requests are embedded online, while non-urgent path rewriting is delayed until time-window or eligible resource-release events.

A candidate migration from p_o to p_n is accepted only if the net gain is positive:

$$G_e^k(p_o, p_n) = \left[\Phi_e^k(p_o) - \Phi_e^k(p_n)\right] - \kappa C_{\mathrm{rw}}^{k,e}(p_o, p_n) > \Gamma, \tag{28}$$

where $\Gamma \geq 0$ is a stability margin. This condition prevents frequent rewrites when the expected improvement is small.

D. Consistent Rule Update

For accepted migrations, the controller follows an install-before-remove update order. It first installs new rules on switches in $p_n \setminus p_o$, then redirects the affected traffic to the new path, and finally deletes obsolete rules in $p_o \setminus p_n$. This order reduces the chance of transient packet loss compared with deleting old rules before the new path is ready.

V. Algorithms and Complexity

A. Initial Embedding

Algorithm 1 summarizes the online initial embedding process for each arriving VNR. For each request, the controller maps virtual nodes using residual-resource and rule-pressure scores, then maps virtual links using feasible path scoring. A VNR is accepted only when both node and link mapping are feasible.

B. Selective Rule Rewriting

Algorithm 2 is activated at every time-window boundary and after an eligible VNR-expiration event satisfying the release-trigger guard. It ranks active virtual links by priority

Algorithm 1 Online initial embedding of an arriving VNR

Require: Substrate graph G^s, arriving VNR k, residual resources, and rule-table state
Ensure: Admission decision and updated controller state
Set temporary admission flag $z_k \leftarrow 1$ and initialize an empty temporary mapping
Sort virtual nodes by decreasing node demand and incident bandwidth
for each virtual node $i \in N_k^v$ **do**
  Build the feasible substrate set using (5), (6), and (7)
  Select $u^* = \arg\max_u \Psi_u(t)$ using (22)
  if no feasible u^* exists **then**
    Release all temporary reservations and set $z_k \leftarrow 0$
    return rejected
  Temporarily reserve u^* for virtual node i
for each virtual link $e \in E_k^v$ **do**
  Generate K candidate paths between the mapped endpoints
  Remove paths violating bandwidth, delay, or rule-table constraints
  Select $p^* = \arg\min_p \Phi_e^k(p)$ using (23)
  if no feasible p^* exists **then**
    Release all temporary reservations and set $z_k \leftarrow 0$
    return rejected
  Temporarily reserve the resources and forwarding entries of p^*
Commit the temporary mapping and install only the selected flow rules
Add k to $\mathcal{A}(t)$ and update residual resources and $\mathcal{Q}(t)$
return accepted

Algorithm 2 Priority-based selective rule rewriting

Require: Active set $\mathcal{A}(t)$, residual resources, rule-table state, priority queue $\mathcal{Q}(t)$, trigger type q, and previous assessment time t_{last}
Ensure: Updated paths and switch rules for selected virtual links
Remove expired VNRs and release their node, link, and rule-table resources
if q is a release trigger and $\big(\max_{(u,v)\in E^s} \ell_{uv}(t) < 0.70$ or $t - t_{\text{last}} < \Delta/2\big)$ **then**
  return the updated controller state without migration assessment
Compute Ω_e^k for all active virtual links using (27)
Sort virtual links in descending order of Ω_e^k
Let m_A be the number of active virtual links
Retain at most $m_Q = \min\{\lceil 0.25 m_A \rceil, 30\}$ highest-priority links
for each retained virtual link e in the priority queue **do**
  Let p_o be the current substrate path of e
  Generate feasible alternative paths $\{p_n\}$
  Compute $G_e^k(p_o, p_n)$ using (28)
  if there exists p_n with maximum gain above Γ **then**
    Install new rules on switches in $p_n \setminus p_o$
    Redirect the affected traffic consistently to p_n
    Delete obsolete rules on switches in $p_o \setminus p_n$
    Update residual resources, rule tables, and database entries
Set $t_{\text{last}} \leftarrow t$

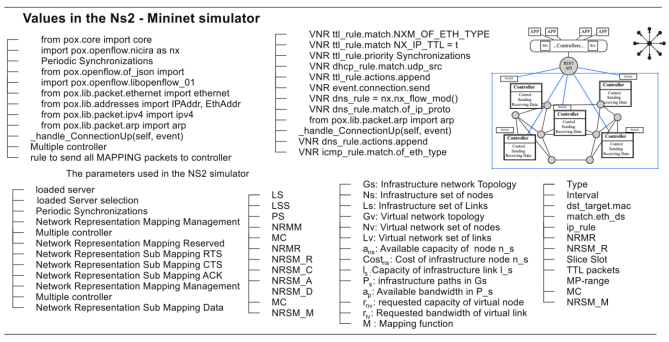


Fig. 3. Evaluation for SDN-enabled virtual network resource management.

and considers alternative paths only for high-priority links. A rewrite is performed only if the migration satisfies all constraints and produces positive net gain.

C. Computational Complexity

Let $|N^s| = n_s$, $|E^s| = m_s$, $|N_k^v| = n_k$, and $|E_k^v| = m_k$. Computing node scores requires $O(n_s + m_s)$ time. Greedy node mapping requires $O(n_k n_s)$ in the worst case. If K candidate paths are generated for each virtual link using a shortest-path routine, link mapping requires approximately

$$O(m_k K(m_s + n_s \log n_s)) . \tag{29}$$

Therefore, the initial embedding complexity for one VNR is

$$O(n_s + m_s + n_k n_s + m_k K(m_s + n_s \log n_s)) . \tag{30}$$

For selective rewriting, let m_A be the number of active virtual links and let $m_Q \le \min\{\lceil 0.25 m_A \rceil, 30\}$ be the bounded subset retained for alternative-path evaluation. The complexity is

$$O(m_A \log m_A + m_Q K(m_s + n_s \log n_s)) , \tag{31}$$

where $m_A \log m_A$ corresponds to priority sorting and the second term corresponds to path generation for the retained links. This bounded evaluation policy is less expensive than global remapping when $m_Q \ll m_A$ or when few links satisfy the positive-gain condition.

VI. Simulation Setup and Results

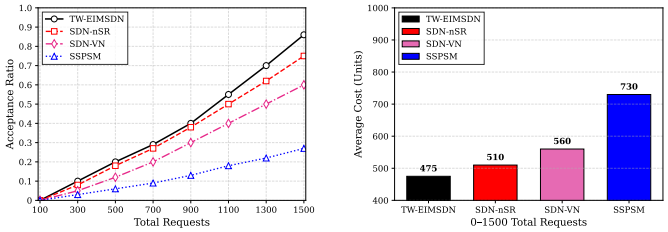
The evaluation uses a self-contained event-driven Python 3.13 simulator with NetworkX 3.6.1 and NumPy 2.3.5. Every reported table is reproducible from the supplied script and CSV outputs. Each substrate is a connected synthetic graph constructed from a random spanning tree plus uniformly sampled additional edges; it is not presented as a named real topology. All methods receive the same graph, capacities, request stream, and seed in each paired run.

The main study uses 50 substrate nodes, 150 links, 1,000 VNRs, and 30 paired independent seeds. Inter-arrival times are exponential with rate 0.45 and lifetimes are exponential with mean 140. VNRs contain 2–7 nodes. Node demands are uniform on $[6, 20]$ and link demands on $[5, 18]$. Each request graph is connected and receives additional links with probability 0.42. The reported utilization is the mean online occupancy sampled after every arrival, not a final cumulative allocation. Thus, utilization may fall when earlier VNRs expire even while the cumulative acceptance ratio rises.

The baselines are: 1) SDN without selective rewriting (SDN-nSR), an ablation sharing TW-EIMSDN's admission and initial mapping but disabling all delayed rewrites; 2) the dynamic SDN virtual-network remapping baseline (SDN-VN), an aggressive variant that evaluates a larger active subset and accepts any positive path-score improvement without a rewrite penalty; and 3) static shortest-path substrate mapping (SSPSM), a static path-based heuristic. The latter two are controlled implementations inspired by dynamic SDN and path-based VNE literature [15], [16]; they are not claimed as bit-identical reproductions of external code.

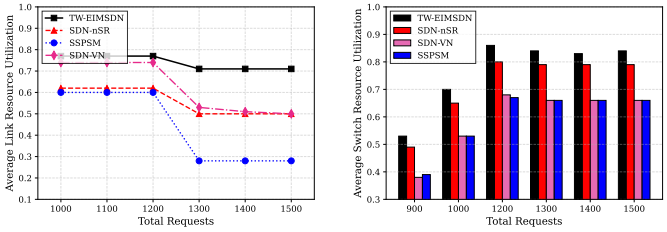
TABLE III
COMPLETE EXPERIMENTAL AND ALGORITHMIC PARAMETERS.

Parameter	Value / definition
Substrate	connected-random; $\|N^s\| = 50$, $\|E^s\| = 150$ (main)
Node/link/table capacity	$C_u \sim U[80, 140]$; $B_{uv} \sim U[70, 120]$; $F_u \sim U\{50, 100\}$
Link delay	$D_{uv} \sim U[1, 8]$; VNR delay bound $U[24, 55]$
Arrival/lifetime	exponential rate 0.45; exponential mean 140
VNR size/demands	2–7 nodes; $d_i^k \sim U[6, 20]$; $b_e^k \sim U[5, 18]$
Candidate search	$K = 8$; maximum 9 hops
Node score	$(\lambda_c, \lambda_b, \lambda_f) = (0.48, 0.37, 0.15)$
Path score	$(\omega_1, \omega_2, \omega_3, \omega_4) = (0.34, 0.18, 0.31, 0.17)$
Priority score	$(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (0.36, 0.33, 0.20, 0.11)$
Rewrite control	$\Delta = 25$, $\Gamma = 0.005$, $\kappa = 0.08$, top fraction 0.25, budget 30
Release trigger	additional assessment only if max link use ≥ 0.70 and at least $\Delta/2$ elapsed
Main statistics	30 paired seeds; mean and 95% confidence interval
Secondary studies	sensitivity: 10 seeds/700 VNRs; scale: 6 seeds with 12 VNRs per substrate node; K: 8 seeds/500 VNRs



(a) Acceptance rate (b) Average cost

Fig. 4. Acceptance rate and average cost comparison.



(a) Average link usage (b) Average switch usage

Fig. 5. Average link and switch resource usage.

VII. RESULTS AND DISCUSSION

A. Main Performance and Numerical Confidence Intervals

Fig. 4 presents the acceptance-rate and average-cost comparisons. Table IV provides the numerical basis for the performance claims. TW-EIMSDN reaches 0.884 ± 0.011 acceptance, compared with 0.870 ± 0.011 for SDN-nSR. Because the same seed is used for each method, the paired mean gain is 1.41 percentage points with a 95% paired interval of [0.88, 1.94]. This gain is obtained with an average-cost increase of 0.86% relative to the ablation. The aggressive SDN-VN variant attains higher acceptance, but it performs substantially more path migrations and rule updates. The result therefore demonstrates a trade-off, not simultaneous dominance on every metric.

B. Rule-Rewrite Overhead and Ablation

The rule-update counters in Table V are measured outcomes rather than metric definitions. TW-EIMSDN performs 347 ± 18 accepted migrations and 1496 ± 84 add/delete operations, while skipping 4341 ± 90 low-gain candidates. SDN-VN performs 5830 ± 103 rule-update operations. Thus, TW-EIMSDN reduces rewrite activity by 74.3% relative to aggressive rewriting. SDN-nSR has no migration overhead by construction, which is why it is an ablation rather than an independent published benchmark.

C. Link and Flow-Table Utilization

Fig. 5 presents the average link- and switch-utilization results. The utilization curves are state averages sampled after each request arrival. They are not monotonically cumulative because expirations release node, link, and table resources. Consequently, mean link usage can decline over an interval while cumulative acceptance increases; the two quantities use different denominators and temporal semantics. This definition explains why the two trajectories can move in different directions during the same interval.

D. Sensitivity to Δ and Γ

Across the tested grid, acceptance ranges from 0.861 to 0.874, whereas mean rewrite activity ranges from 173 to 1242. Shorter windows and smaller margins generally create more opportunities for migration; larger margins suppress marginal changes. The setting $(\Delta, \Gamma) = (25, 0.005)$ provides a balanced compromise between acceptance performance and rewrite overhead and is therefore used in the main experiments.

E. Scalability

The scalability study increases the substrate from 50/150 to 150/450 nodes/links while keeping average degree approximately constant. Runtime rises from 3.20 to 6.82 ms per request; acceptance remains between 0.785 and 0.886 under the tested request load. These results support bounded practical scaling for the implemented heuristic but do not establish scalability to arbitrary carrier-size networks.

F. Candidate-Path Bound K

Increasing K expands the route search but also increases controller computation. In the tested settings, runtime changes from 1.85 to 4.61 ms/request and acceptance from 0.788 to 0.903. The observed variation confirms that K is a computational and solution-quality parameter, not an innocuous implementation constant. The bounded value $K = 8$ is used in the main study as a compromise.

VIII. CONCLUSION

This paper presented TW-EIMSDN, a rule-rewrite-aware online VNE framework that separates immediate admission and initial rule installation from deferred selective migration. The mathematical model and controller workflow incorporate node and link resources, delay, flow-table capacity, residual pressure, and explicit rule-addition/deletion cost.

Across 30 paired seeds, TW-EIMSDN achieves 0.884 ± 0.011 acceptance compared with 0.870 ± 0.011 for the no-rewrite ablation. Its cost is slightly higher than the ablation

TABLE IV
PERFORMANCE SUMMARY OVER 30 PAIRED SEEDS (MEAN ± 95% CI).

Method	Acceptance	Avg. cost	Link use	Table use	N_{rw}	ms/request
TW-EIMSDN	0.884 ± 0.011	226.03 ± 1.75	0.608 ± 0.008	0.295 ± 0.004	1496 ± 84	2.86 ± 0.06
SDN-nSR	0.870 ± 0.011	224.11 ± 1.52	0.611 ± 0.006	0.292 ± 0.004	0 ± 0	2.06 ± 0.04
SDN-VN	0.899 ± 0.011	227.34 ± 2.03	0.607 ± 0.006	0.297 ± 0.004	5830 ± 103	2.50 ± 0.04
SSPSM	0.795 ± 0.014	207.36 ± 2.01	0.515 ± 0.008	0.247 ± 0.005	0 ± 0	1.78 ± 0.03

TABLE V
MEASURED RULE-UPDATE OVERHEAD (MEAN ± 95% CI). AGGREGATE N_{rw} IS COMPUTED PER RUN BEFORE AVERAGING; INDEPENDENTLY ROUNDED COMPONENT MEANS MAY THEREFORE DIFFER BY ONE OPERATION.

Method	N_{mig}	N_{add}	N_{del}	N_{rw}	N_{skip}
TW-EIMSDN	347 ± 18	595 ± 35	902 ± 50	1496 ± 84	4341 ± 90
SDN-nSR	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
SDN-VN	1310 ± 24	2638 ± 47	3192 ± 58	5830 ± 103	0 ± 0

TABLE VI
SENSITIVITY SUMMARY (10 SEEDS PER SETTING).

Δ	Γ	Acceptance	Avg. cost	N_{rw}
20	0.000	0.872 ± 0.025	226.7 ± 2.8	1242 ± 143
20	0.005	0.874 ± 0.023	226.5 ± 3.5	1085 ± 108
20	0.025	0.867 ± 0.024	225.9 ± 3.2	711 ± 80
20	0.050	0.866 ± 0.024	224.5 ± 2.5	430 ± 53
50	0.000	0.870 ± 0.019	225.4 ± 2.7	790 ± 82
50	0.005	0.871 ± 0.023	225.9 ± 2.9	706 ± 65
50	0.025	0.872 ± 0.021	224.0 ± 3.2	463 ± 38
50	0.050	0.872 ± 0.023	225.7 ± 3.7	280 ± 30
100	0.000	0.872 ± 0.023	224.0 ± 3.3	463 ± 36
100	0.005	0.872 ± 0.025	224.2 ± 2.3	433 ± 35
100	0.025	0.865 ± 0.022	223.4 ± 2.9	291 ± 29
100	0.050	0.861 ± 0.020	224.2 ± 2.9	173 ± 26

TABLE VII
SCALABILITY RESULTS (6 SEEDS PER SCALE WITH PROPORTIONALLY SCALED OFFERED LOAD).

Nodes	Links	Acceptance	N_{rw}	ms/request
50	150	0.886 ± 0.028	828 ± 105	3.20 ± 0.16
100	300	0.827 ± 0.013	1384 ± 80	5.02 ± 0.11
150	450	0.785 ± 0.015	1648 ± 80	6.82 ± 0.11

TABLE VIII
K-SENSITIVITY RESULTS (8 SEEDS PER SETTING).

K	Acceptance	Avg. cost	ms/request
3	0.788 ± 0.018	207.8 ± 5.0	1.85 ± 0.03
5	0.865 ± 0.012	221.0 ± 3.4	2.39 ± 0.06
8	0.889 ± 0.019	225.7 ± 4.1	3.22 ± 0.05
12	0.903 ± 0.027	229.5 ± 3.7	4.61 ± 0.09

because it admits additional, harder requests, but it reduces rule-update activity by 74.3% relative to aggressive dynamic rewriting while preserving most of that method's acceptance. The sensitivity study shows that Δ and Γ directly control rewrite frequency and responsiveness; the scale and K studies quantify the runtime consequences of larger substrates and broader path search.

Future work should validate the mechanism on named real topologies and an SDN testbed, incorporate hardware-specific table and acknowledgment costs, and evaluate the gain filter as an auxiliary objective within reproducible PPO-VNE or HRL-ACRA implementations.

ACKNOWLEDGMENT

The work in this paper was supported in part by the Federal Ministry of Research, Technology, and Space (BMFTR), Germany, through the Project 6GEM+ under Grant 16KIS2411; and in part by the European Union through the 6G-Path project under Grant 101139172.

REFERENCES

[1] A. Javadpour, "Improving resources management in network virtualization by utilizing a software-based network," *Wireless Personal Communications*, vol. 106, no. 2, pp. 505–519, 2019.
[2] E. Thomatos, A. Sgora, A. Tsipis, and P. Chatzimisios, "AI methods in network slice life-cycle phases: A survey," *Electronics*, vol. 14, no. 20, p. 4053, 2025.
[3] A. Satpathy, M. Narayan Sahoo, C. Swain, P. Bellavista, M. Guizani, K. Muhammad, and S. Bakshi, "Virtual network embedding: Literature assessment, recent advancements, opportunities, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 6, pp. 3861–3914, 2025.
[4] S. M. Mirmohseni, C. Tang, and A. Javadpour, "Using markov learning utilization model for resource allocation in cloud of thing network," *Wireless Personal Communications*, vol. 115, no. 1, pp. 653–677, 2020.
[5] Z. Duan and T. Wang, "Towards learning-based energy-efficient online coordinated virtual network embedding framework," *Computer Networks*, vol. 239, p. 110139, 2024.
[6] T. Wang, L. Shen, Q. Fan, T. Xu, T. Liu, and H. Xiong, "Joint admission control and resource allocation of virtual network embedding via hierarchical deep reinforcement learning," *IEEE Transactions on Services Computing*, vol. 17, no. 3, pp. 1001–1015, 2024.
[7] C. L. Moreira, C. A. Kamienski, and R. A. C. Bianchi, "5G and edge: A reinforcement learning approach for virtual network embedding with cost optimization and improved acceptance rate," *Computer Networks*, vol. 247, p. 110434, 2024.
[8] 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, p. 1918, 2024.
[9] W. Wang, L. Yang, X. Yang, and J. Wang, "Reducing flow table update costs in software-defined networking," *Sensors*, vol. 23, no. 23, p. 9375, 2023.
[10] M. Hussain, R. Amin, R. Gantassi, A. H. Alshehri, J. Frnda, and S. M. Raza, "Efficient handling of ACL policy change in SDN using reactive and proactive flow rule installation," *Scientific Reports*, vol. 14, no. 1, p. 14976, 2024.
[11] M. Jiménez-Lázaro, J. Berrocal, and J. Galán-Jiménez, "Flow-based service time optimization in software-defined networks using deep reinforcement learning," *Computer Communications*, vol. 216, pp. 221–233, 2024.
[12] A. Javadpour, F. Ja'fari, T. Taleb, C. Benzaïd, P. R. Tomas, L. Rosa, J. Proença, and L. Cordeiro, "A reinforcement learning approach to virtual network embedding problems in 5g networks," *IEEE Transactions on Network Science and Engineering*, 2026.
[13] X. Xiao, "DVNE-DRL: Dynamic virtual network embedding algorithm based on deep reinforcement learning," *Scientific Reports*, vol. 13, no. 1, p. 19789, 2023.
[14] G. Huang, I. Ullah, H. Huang, and K. T. Kim, "Predictive mobility and cost-aware flow placement in SDN-based IoT networks: A Q-learning approach," *Journal of Cloud Computing*, vol. 13, no. 1, p. 26, 2024.
[15] A. Javadpour, "Improving resources management in network virtualization by utilizing a software-based network," *Wireless Personal Communications*, vol. 106, no. 2, pp. 505–519, 2019.
[16] A. Javadpour, F. Ja'fari, P. Pinto, and W. Zhang, "Mapping and embedding infrastructure resource management in software defined networks," *Cluster Computing*, vol. 26, no. 1, pp. 461–475, 2023.