Boundary-Contiguous Initial Placement for Structured Network-Service Embedding

Amir Javadpour*, Forough Ja'fari†, Tarik Taleb‡
*MOSA!C Lab / ICTFICIAL Oy, Finland †Sharif University of Technology, Iran ‡Ruhr University Bochum, Germany
Corresponding author: a.javadpour87@gmail.com

***Abstract*—This paper studies structured aggregate service embedding, a constrained initial-placement problem in network virtualization in which each online request specifies an aggregate compute demand and an inter-node bandwidth demand and must occupy a connected interval on a linear or ring substrate. Unlike classical graph-to-graph virtual network embedding, the request is not represented as a virtual graph; this narrower scope is stated explicitly. We formulate a lexicographic objective that first maximizes accepted requests and then minimizes the number of residual free-capacity segments, and we develop a boundary-contiguous placement rule that evaluates bandwidth-feasible segment boundaries. Under homogeneous demands and nonbinding intra-segment bandwidth, a node-packing normal form and a boundary-compaction result are established for linear substrates. The procedure requires $O(RV)$ time and $O(V)$ auxiliary memory for R requests and V substrate nodes. Across 600 paired synthetic instances, the acceptance-ratio and node-utilization differences are marginal and are interpreted descriptively; the clearest effects concern link use and fragmentation. The method reduces average link utilization by 22.91% relative to LFBM and 10.70% relative to Random, and reduces residual fragmentation by 14.36% and 36.08%, respectively. CenterSpread uses less link bandwidth and mapping cost, but yields lower acceptance and substantially higher fragmentation. The method is therefore positioned as a low-complexity fragmentation-aware initializer for path and cycle substrates, rather than as a general solution to arbitrary-graph VNE.**

***Index Terms*—Structured service embedding, boundary-contiguous placement, residual fragmentation, online resource allocation.**

I. Introduction

Network virtualization enables isolated services to share substrate infrastructure and has motivated extensive work on virtual network embedding (VNE) [1], [2], [5]. In classical VNE, a request is a graph whose virtual nodes and links are mapped to substrate nodes and paths under computing and bandwidth constraints [6].

The problem considered here is deliberately narrower. Each online request is an aggregate service demand that may be divided over consecutive substrate nodes, while the occupied nodes must form one connected interval and the traversed substrate links must satisfy a common bandwidth demand. We therefore refer to the problem as *structured aggregate service embedding*. It is related to VNE as an initial resource-placement subproblem, but it is not presented as a replacement for general graph-to-graph VNE.

Path and cycle abstractions are appropriate when the controllable substrate is an ordered corridor, chain, access segment, or ring. They also permit direct control of residual continuity: an interior placement can split otherwise usable capacity into disconnected pieces even when total residual capacity remains sufficient. The method is not topology-agnostic, and arbitrary graphs are outside the analytical and experimental scope.

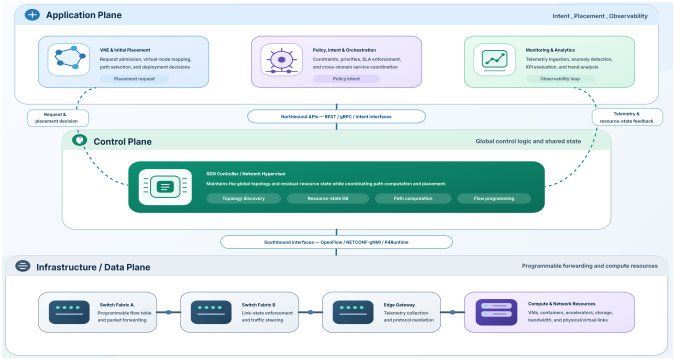


Fig. 1. SDN-oriented deployment context for controller-side monitoring and structured initial placement.

An SDN controller may maintain the residual resource state and invoke the placement rule. Figure 1 is retained to show this controller-side deployment context; however, the algorithm does not depend on a specific SDN protocol or controller implementation. The architecture is therefore presented as an enabling environment rather than as a methodological contribution [7].

Recent VNE research includes reinforcement-learning, metaheuristic, admission-control, and traffic-aware formulations [8]–[11], [13]. These methods address broader graph-based or dynamic objectives, whereas initialization quality can itself materially affect later search quality and supportability [12].

The contribution is not the generic idea of contiguous packing alone. It is the combined formulation of 1) an online lexicographic acceptance–fragmentation objective for bandwidth-segmented path/cycle substrates, 2) a deterministic boundary-candidate rule with explicit assignment and usage functions, and 3) structural normal-form results under clearly stated assumptions. The procedure is implementable by linear scans, giving $O(RV)$ time and $O(V)$ auxiliary memory. The evaluation uses the same 600 synthetic instances for all methods and interprets the small acceptance and utilization differences conservatively; the principal descriptive gains are lower fragmentation and lower link use relative to LFBM and Random.

II. Related Work

VNE has been studied through mathematical programming, ranking heuristics, metaheuristics, and learning-based methods [2], [6]. Reinforcement-learning approaches address dynamic graph embedding, coordinated node–link mapping, and joint admission control, but introduce state, reward, training, and reproducibility requirements [8]–[11]. Traffic-aware and emulation-oriented studies additionally target controller operation and realistic network behavior [13], [14].

The present study is not directly comparable with those general graph-VNE systems because its request object and substrate class are narrower. Its closest conceptual connection is initialization-aware

TABLE I
CONTEXTUAL POSITIONING RELATIVE TO BROADER VNE STUDIES.

Work	Y	Type	Strength	Gap
Satpathy *et al.* [6]	25	Survey	Recent VNE taxonomy	No embedding method
Lim *et al.* [8]	23	Survey	RL-based VNE review	No deterministic mapper
Xiao *et al.* [9]	23	DRL	Handles dynamic settings	More complex, less interpretable
Duan *et al.* [10]	24	PPO	Joint node–link mapping	Needs training/features
Rubio-Loyola and Aguilar-Fuster [12]	24	Metaheur.	Good online initialization insight	Still search-based
Wang *et al.* [11]	24	HDRL	Admission and allocation jointly	Heavier than initial mapping
Minardi *et al.* [13]	24	SDN	Traffic-aware operation	Broader than initial mapping
Kumar *et al.* [14]	24	Emul.	Realistic testbed	Not a mapping method

embedding: Rubio-Loyola and Aguilar-Fuster [12] show that initialization can affect acceptance, revenue, and search quality. Here, that issue is isolated for aggregate requests on ordered substrates, where residual fragmentation can be represented exactly and boundary compaction can be analyzed.

Accordingly, the cited learning and metaheuristic methods provide research context rather than direct experimental baselines. The experiments below use controlled placement ablations with the same request model so that the effect of boundary selection can be isolated without conflating different virtual-graph definitions, training budgets, or optimization objectives.

Table I is contextual: the listed methods solve broader graph-based problems, while this paper isolates a structured initial-placement subproblem for which residual segments and boundary compaction can be studied explicitly.

III. STRUCTURED SERVICE MODEL AND PROBLEM FORMULATION

A. Structured substrate and request model

We consider a path or cycle substrate and an ordered stream of aggregate service requests. A request is not a virtual graph: it specifies one scalar compute demand and one inter-node bandwidth demand, and its allocation may be split only across consecutive substrate nodes. The system is represented by

$$\mathcal{S} = \{\mathcal{N}, \mathcal{R}, \mathcal{M}\}, \tag{1}$$

where $\mathcal{N}$ denotes the substrate resource state, $\mathcal{R} = \{r_1, r_2, \ldots, r_R\}$ is the ordered set of arrived requests, and $\mathcal{M}$ is a candidate mapping solution. The structured substrate is encoded by a $2 \times V$ matrix

$$\mathcal{N} = \begin{bmatrix} n_{1,1} & n_{1,2} & \cdots & n_{1,V} \\ n_{2,1} & n_{2,2} & \cdots & n_{2,V} \end{bmatrix}, \tag{2}$$

where the first row stores node capacities and the second row stores edge bandwidths. More precisely,

$$n_{i,j} = \begin{cases} c_j, & i = 1, \\ b_{j,j+1}, & i = 2,\ 1 \le j < V, \\ b_{V,1}, & i = 2,\ j = V \text{ in the ring case}, \\ 0, & i = 2,\ j = V \text{ in the linear case}. \end{cases} \tag{3}$$

Here c_j is a scalar residual compute capacity and $b_{j,j+1}$ is residual bandwidth. In the experiments, c_j is a normalized CPU-equivalent unit; memory, storage, and GPU capacity are not modeled separately. A multi-resource extension would replace c_j and v_i by vectors and impose componentwise feasibility, but that extension is outside the present results. The ordered representation applies only to path and cycle substrates.

Each request $r_i \in \mathcal{R}$ is characterized by aggregate compute demand $v_i > 0$ and bandwidth demand $e_i > 0$ between consecutive placement components when more than one substrate node is used. The online model permits heterogeneous requests; the structural results in section V use a homogeneous, nonbinding-bandwidth special case and do not claim general VNE optimality.

B. Mapping representation and feasibility

A mapping solution is written as

$$\mathcal{M} = \{m_1, m_2, \ldots, m_R\}, \tag{4}$$

where m_i is the mapping of request r_i. Each mapped request is represented by

$$m_i = \{m_1^i, m_2^i, \ldots, m_{M_i}^i\}, \tag{5}$$

where M_i is the number of placement components used to realize request r_i, and each placement component is described by a ternary tuple

$$m_j^i = \{x_j^i, c_j^i, d_j^i\}. \tag{6}$$

Here x_j^i is the substrate node index on which the jth placement component of request r_i is placed, c_j^i is the capacity allocated there, and $d_j^i \in \{-1, 0, 1\}$ indicates the direction used to reach the next virtual node. The value $d_j^i = 1$ indicates traversal toward increasing substrate indices, $d_j^i = -1$ indicates traversal toward decreasing indices, and $d_j^i = 0$ is used for the last virtual node in the request path. For every accepted request,

$$\sum_{j=1}^{M_i} c_j^i = v_i. \tag{7}$$

To make the optimization target explicit, we define the binary acceptance variable

$$z_i = \begin{cases} 1, & \text{if request } r_i \text{ is accepted and feasibly embedded}, \\ 0, & \text{otherwise}. \end{cases} \tag{8}$$

The primary objective is to maximize the number of accepted requests,

$$\max \sum_{i=1}^{R} z_i. \tag{9}$$

Subject to the same acceptance level, the secondary objective is to preserve residual continuity of the substrate, because continuity directly affects future supportability in linear and ring topologies. For linear substrates, let $\Gamma(\mathcal{M})$ denote the number of maximal residual free-capacity segments after performing mapping $\mathcal{M}$. The preferred solution is therefore lexicographically characterized by first maximizing (9) and then minimizing $\Gamma(\mathcal{M})$. For ring substrates, the same idea is applied to cyclic free segments after fixing a feasible break point.

Every accepted request must occupy one connected interval in the path case or one connected cyclic interval in the ring case. This is a defining restriction of the structured service model, not a property asserted for arbitrary virtual graphs.

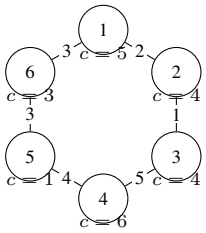


Fig. 2. Ring substrate α; node indices and capacities are shown in black, and edge labels denote bandwidth.

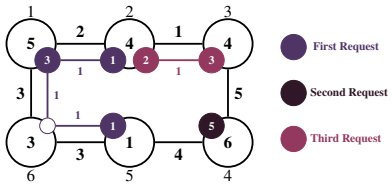


Fig. 3. Sample feasible mapping for substrate network α.

C. Assignment and usage functions

The placement-assignment function is defined as

$$va_{\mathcal{M}}(i,j,k) = \begin{cases} 1, & x_j^i = k, \\ 0, & x_j^i \neq k, \end{cases} \tag{10}$$

meaning that the jth placement component of request r_i is assigned to substrate node k if and only if $va_{\mathcal{M}}(i,j,k) = 1$.

The edge assignment function is defined as

$$ea_{\mathcal{M}}(i,j,k) = \begin{cases} 0, & j = M_i, \\ 1, & x_j^i < x_{j+1}^i,\ d_j^i = 1,\ x_j^i \le k < x_{j+1}^i, \\ 0, & x_j^i < x_{j+1}^i,\ d_j^i = 1,\ k < x_j^i, \\ 0, & x_j^i < x_{j+1}^i,\ d_j^i = 1,\ k \ge x_{j+1}^i, \\ 0, & x_j^i < x_{j+1}^i,\ d_j^i = -1,\ x_j^i \le k < x_{j+1}^i, \\ 1, & x_j^i < x_{j+1}^i,\ d_j^i = -1,\ k < x_j^i, \\ 1, & x_j^i < x_{j+1}^i,\ d_j^i = -1,\ k \ge x_{j+1}^i, \\ 0, & x_j^i > x_{j+1}^i,\ d_j^i = 1,\ x_{j+1}^i \le k < x_j^i, \\ 1, & x_j^i > x_{j+1}^i,\ d_j^i = 1,\ k < x_{j+1}^i, \\ 1, & x_j^i > x_{j+1}^i,\ d_j^i = 1,\ k \ge x_j^i, \\ 1, & x_j^i > x_{j+1}^i,\ d_j^i = -1,\ x_{j+1}^i \le k < x_j^i, \\ 0, & x_j^i > x_{j+1}^i,\ d_j^i = -1,\ k < x_{j+1}^i, \\ 0, & x_j^i > x_{j+1}^i,\ d_j^i = -1,\ k \ge x_j^i. \end{cases} \tag{11}$$

This definition preserves the directional-path idea while keeping the notation consistent.

The substrate usage functions are then defined as

$$cu_{\mathcal{M}}(k) = \sum_{i=1}^{R} \sum_{j=1}^{M_i} c_j^i \, va_{\mathcal{M}}(i,j,k), \tag{12}$$

and

$$bu_{\mathcal{M}}(k) = \sum_{i=1}^{R} \sum_{j=1}^{M_i} e_i \, ea_{\mathcal{M}}(i,j,k). \tag{13}$$

A mapping solution is feasible if

$$cu_{\mathcal{M}}(k) \le n_{1,k}, \qquad \forall k \in \{1,2,\ldots,V\}, \tag{14}$$

and

$$bu_{\mathcal{M}}(k) \le n_{2,k}, \qquad \forall k \in \{1,2,\ldots,V\}. \tag{15}$$

D. Illustrative example

A useful example clarifies the assignment and usage functions. Consider the ring substrate α in Figure 2. Node indices, capacities, and bandwidth labels are rendered in high-contrast black.

For this example, the node-capacity row is

$$\mathcal{N}_\alpha = \{5,4,4,6,1,3\}, \tag{16}$$

TABLE II
VALUES OF THE PLACEMENT-ASSIGNMENT FUNCTION $va_{\mathcal{M}_\alpha}$.

	k=1	k=2	k=3	k=4	k=5	k=6
$va_{\mathcal{M}_\alpha}(1,1,k)$	0	0	0	0	0	1
$va_{\mathcal{M}_\alpha}(1,2,k)$	1	0	0	0	0	0
$va_{\mathcal{M}_\alpha}(1,3,k)$	0	1	0	0	0	0
$va_{\mathcal{M}_\alpha}(2,1,k)$	0	0	0	1	0	0
$va_{\mathcal{M}_\alpha}(3,1,k)$	0	1	0	0	0	0
$va_{\mathcal{M}_\alpha}(3,2,k)$	0	0	1	0	0	0

TABLE III
VALUES OF THE EDGE ASSIGNMENT FUNCTION $ea_{\mathcal{M}_\alpha}$.

	k=1	k=2	k=3	k=4	k=5	k=6
$ea_{\mathcal{M}_\alpha}(1,1,k)$	1	0	0	0	0	0
$ea_{\mathcal{M}_\alpha}(1,2,k)$	0	0	0	0	1	1
$ea_{\mathcal{M}_\alpha}(1,3,k)$	0	0	0	0	0	0
$ea_{\mathcal{M}_\alpha}(2,1,k)$	0	0	0	0	0	0
$ea_{\mathcal{M}_\alpha}(3,1,k)$	0	1	0	0	0	0
$ea_{\mathcal{M}_\alpha}(3,2,k)$	0	0	0	0	0	0

and the substrate edge-bandwidth matrix is

$$\mathcal{E}_\alpha = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 3 \\ 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 & 0 & 3 \\ 3 & 0 & 0 & 0 & 3 & 0 \end{bmatrix}. \tag{17}$$

Suppose the ordered request capacities are $\{7,5,6\}$ and the common edge-bandwidth requirement is 1. A sample feasible mapping is shown in Figure 3.

For this mapping,

$$\mathcal{M}_\alpha = \{m_1, m_2, m_3\}, \tag{18}$$

with

$$m_1 = \{\{6,1,-1\},\{1,5,1\},\{2,1,0\}\}, \tag{19}$$

$$m_2 = \{\{4,5,0\}\}, \tag{20}$$

and

$$m_3 = \{\{2,2,1\},\{3,4,0\}\}. \tag{21}$$

The corresponding values of the vertex-assignment and edge-assignment functions are listed in Table II and Table III.

Using Equation 12, Equation 13, Table II, and Table III, the resulting resource usage values are

$$\begin{aligned} &cu_{\mathcal{M}_\alpha}(1) = 5, \quad cu_{\mathcal{M}_\alpha}(2) = 3, \quad cu_{\mathcal{M}_\alpha}(3) = 4, \\ &cu_{\mathcal{M}_\alpha}(4) = 5, \quad cu_{\mathcal{M}_\alpha}(5) = 0, \quad cu_{\mathcal{M}_\alpha}(6) = 1, \end{aligned} \tag{22}$$

and

$$\begin{aligned} &bu_{\mathcal{M}_\alpha}(1) = 1, \quad bu_{\mathcal{M}_\alpha}(2) = 1, \quad bu_{\mathcal{M}_\alpha}(3) = 0, \\ &bu_{\mathcal{M}_\alpha}(4) = 0, \quad bu_{\mathcal{M}_\alpha}(5) = 1, \quad bu_{\mathcal{M}_\alpha}(6) = 1. \end{aligned} \tag{23}$$

Algorithm 1 Boundary-contiguous initial placement on an ordered substrate

Require: Ordered requests $\mathcal{R}$ and residual substrate state $\mathcal{N}$
Ensure: Mapping set $\mathcal{M}$ and acceptance variables z_i
$\mathcal{M} \leftarrow \emptyset$
for each $r_i = (v_i, e_i)$ in arrival order **do**
  Scan the bandwidth row and form maximal segments whose links satisfy e_i
  Scan each segment from both boundaries and generate feasible boundary candidates
  if no candidate provides aggregate capacity v_i **then**
    $z_i \leftarrow 0$ and continue
  Choose the candidate minimizing $(\Delta\Gamma, h_i, \text{boundary index})$ lexicographically
  Commit its consecutive capacity allocations and traversed-link bandwidth
  $z_i \leftarrow 1$; append the resulting tuple sequence to $\mathcal{M}$

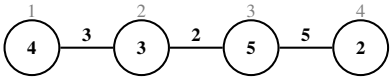


Fig. 4. Substrate network β.

These values satisfy Equation 14 and Equation 15, so the mapping is feasible. The example shows that the refined notation remains fully compatible with the figures and assignment tables.

IV. Boundary-Contiguous Initial Placement

Contiguous packing itself is classical. The specific mechanism used here combines bandwidth-induced segmentation, evaluation of both boundaries of every feasible segment, and lexicographic selection by acceptance feasibility and fragmentation increase. Requests are processed online in arrival order; no training, population search, or iterative global optimization is used.

A. Linear-topology strategy

For a linear substrate, links with residual bandwidth below e_i partition the path into maximal feasible segments. Both boundaries of each segment are inspected. A candidate is feasible when consecutive residual node capacities can supply v_i without crossing an infeasible link. Feasible candidates are ranked first by the increase in residual-segment count $\Delta\Gamma$, then by hop count h_i, and finally by boundary index for deterministic tie breaking.

To clarify the linear strategy, consider the substrate network β shown in Figure 4. Suppose that the ordered requests are $\{3, 2, 8\}$ and the common bandwidth requirement is 2. Starting from the left boundary, the first request is mapped to node 1, the second request is split across nodes 1 and 2, and the third request occupies the remaining contiguous portion of nodes 2, 3, and 4. The resulting mapping is shown in Figure 5.

B. Ring-topology strategy

For a ring, every bandwidth-infeasible edge is a natural break point and the resulting feasible arcs are processed as linear segments. If all links are feasible, one break point is selected by a single scan using the same local fragmentation score, after which Algorithm 1 is applied. This is a cycle-specific construction; no corresponding optimality claim is made for arbitrary graphs.

V. Structural Properties and Validity Conditions

The following results are normal-form statements for a restricted special case. They are not general optimality theorems for classical VNE.

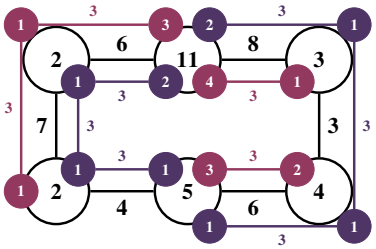


Fig. 5. Boundary-oriented contiguous mapping result for substrate network β.

Assumption 1. *The analysis concerns one linear feasible segment with node capacities $c_1, \ldots, c_L$. Each admitted request has compute demand v, may be divided across consecutive nodes, and has no one-to-one virtual-node placement constraint.*

Assumption 2. *All analyzed requests have the same demand v, and intra-segment link bandwidth is nonbinding for the compaction operation. The online algorithm itself still checks heterogeneous v_i and e_i.*

A. Node-packing normal form

Theorem 1 (Node-packing normal form)**.** *If q homogeneous requests are feasible on the segment, then there is a feasible mapping of the same q requests in which occupied capacity forms a prefix or suffix, every occupied interior node is saturated, and at most one occupied boundary node is partially used.*

Proof. Feasibility implies $qv \le \sum_{k=1}^{L} c_k$. Starting from the left boundary, allocate the total load qv greedily: saturate node 1, then node 2, and continue until the load is exhausted. The occupied nodes form a prefix, all but the last are saturated, and the last is used by at most its residual capacity. Partitioning the cumulative allocated load at multiples of v produces q consecutive request allocations. By Assumption 2, the links crossed by this compaction remain feasible. The same construction applies from the right boundary. □

The transformations in Figure 6 illustrate this normal form; the figures are explanatory and do not extend the stated assumptions.

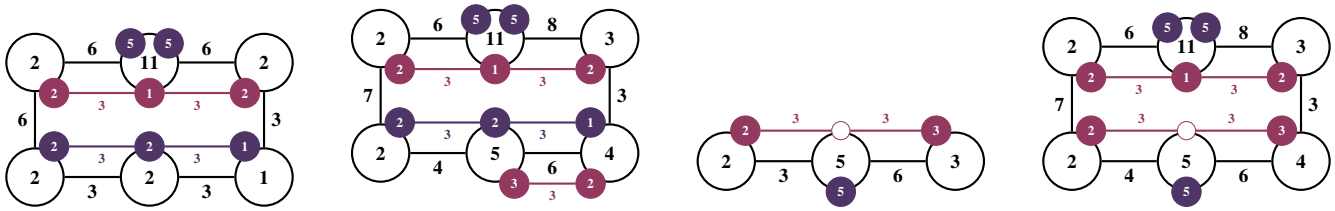
B. Boundary-compaction optimality in the restricted linear case

Proposition 1. *Under Assumption 1–Assumption 2, the maximum accepted-request count is*

$$q^\star = \min\left\{R, \left\lfloor \frac{\sum_{k=1}^{L} c_k}{v} \right\rfloor\right\}. \tag{24}$$

A left- or right-boundary greedy mapping admits $q^\star$ requests and, among mappings with this acceptance count, leaves at most one nonempty residual free-capacity segment.

Proof. No mapping can admit more than $\lfloor \sum_k c_k / v \rfloor$ requests because each accepted request consumes v compute units. The construction in Theorem 1 maps exactly $q^\star v$ units from either boundary and therefore attains the upper bound. Because the occupied nodes form one prefix or suffix, all unused capacity lies in one residual suffix or prefix, except for zero-capacity nodes. Hence the residual-segment count is minimal for the attained acceptance level. □



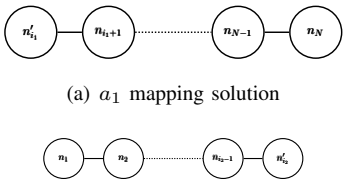
(a) Connected request subgraph used in the first saturation step for substrate network β.

(b) Intermediate mapping after the first boundary compaction step.

(c) Second connected request subgraph used in the saturation argument.

(d) Final mapping after repeated boundary compaction in substrate network β.

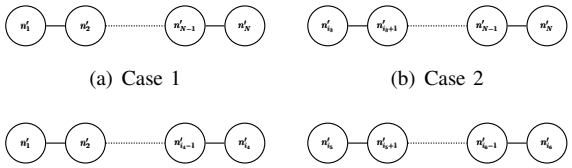
Fig. 6. Illustration of repeated boundary compaction over substrate network β.



(a) a_1 mapping solution

(b) a_2 mapping solution

Fig. 7. Residual-capacity patterns induced by the two boundary-contiguous strategies.



(a) Case 1

(b) Case 2

(c) Case 3

(d) Case 4

Fig. 8. Representative fragmented residual-capacity patterns for non-boundary-contiguous mappings.

C. Computational complexity

Proposition 2. *For R requests and V substrate nodes, the algorithm requires $O(RV)$ time and $O(V)$ auxiliary memory for a path, or for a ring after a break point is selected by a linear scan.*

Proof. For each request, one scan identifies bandwidth-feasible segments. Scanning the disjoint segments from both boundaries visits at most $2V$ node positions, and candidate comparison is constant time per segment. The selected allocation is committed during the same stored scan information. Thus each request costs $O(V)$ time. Segment descriptors, residual capacities, and candidate endpoints require $O(V)$ memory. No wall-clock or energy claim is inferred from this asymptotic result. □

The patterns in Figure 7 and Figure 8 illustrate the difference between boundary-compacted and fragmented residual capacity.

Remark 1. *The compaction result is restricted to a linear feasible segment with splittable homogeneous demands and nonbinding intra-segment bandwidth. A ring can use it only after a break point is fixed. It does not establish optimality for heterogeneous demands, binding link capacities, multiple resource dimensions, or arbitrary graphs.*

VI. Performance Evaluation and Discussion

TABLE IV
Simulation setup.

Item	Setting
Environment	Python 3.11, fixed seed
Substrate	20 nodes; linear, ring
Node capacity	Uniform $[3, 10]$
Edge bandwidth	Uniform $[1, 6]$
Request counts	30, 50, 70
Request capacity	Uniform $[3, 8]$
Request bandwidth	Uniform $[1, 3]$
Trials	100 per topology–load pair
Baselines	EIMSDN, LFBM, CenterSpread, Random

A Python 3.11 simulation evaluates the path and cycle settings only. The same randomly generated instance is reused for every method, creating a paired descriptive comparison. The study does not represent arbitrary-graph VNE, and no inferential significance claim is made because confidence intervals were not retained with the aggregate outputs.

A. Simulation setting and compared methods

The simulated substrate contains $V = 20$ substrate nodes. For each trial, the node capacities are independently drawn from the discrete uniform range $[3, 10]$. The substrate edge bandwidths are independently drawn from $[1, 6]$. Three offered-load levels are considered by generating 30, 50, and 70 online requests, respectively. Each request capacity demand is drawn from $[3, 8]$, and each request bandwidth demand is drawn from $[1, 3]$. Both linear and ring substrate topologies are evaluated. For every topology-load combination, 100 random trials are performed. This yields 600 distinct structured instances in total, and each compared method is executed on the same 600 instances.

Four same-model placement rules are compared. EIMSDN evaluates feasible segment boundaries and minimizes fragmentation increase; LFBM accepts the first feasible boundary placement; CenterSpread starts near the segment center; and Random samples a feasible candidate. These are controlled ablations, not claimed state-of-the-art general-VNE baselines. A direct numerical comparison with graph-based RL or metaheuristic systems would require a different request representation, objective, and training/search budget. At the largest tested load, the $O(RV)$ scan bound corresponds to at most $70 \times 20 = 1400$ substrate-position inspections per instance, excluding plotting and file input/output.

B. Evaluation metrics

The first metric is the acceptance ratio,

$$\mathrm{AR} = \frac{\sum_{i=1}^{R} z_i}{R}. \tag{25}$$

TABLE V
OVERALL AVERAGES ACROSS ALL SCENARIOS (AR: ACCEPTANCE RATIO, NU: NODE UTILIZATION, LU: LINK UTILIZATION, MC: MAPPING COST, FRAG.: FRAGMENTATION).

Method	AR	NU	LU	MC	Frag.
EIMSDN	0.536	0.989	0.619	7.217	0.865
LFBM	0.534	0.988	0.803	7.767	1.010
CenterSpread	0.532	0.979	0.453	6.734	1.873
Random	0.535	0.984	0.693	7.430	1.353

The second metric is the final node-utilization ratio,

$$\mathrm{NU} = \frac{1}{\sum_{k=1}^{V} n_{1,k}^{\mathrm{init}}} \sum_{k=1}^{V} cu_{\mathcal{M}}(k), \tag{26}$$

and the third metric is the final link-utilization ratio,

$$\mathrm{LU} = \frac{1}{\sum_{k=1}^{V} n_{2,k}^{\mathrm{init}}} \sum_{k=1}^{V} bu_{\mathcal{M}}(k). \tag{27}$$

To capture the operational cost of an accepted mapping, we also report the average mapping cost

$$\mathrm{MC} = \frac{1}{\sum_{i=1}^{R} z_i} \sum_{i=1}^{R} z_i\big(v_i + e_i h_i\big), \tag{28}$$

where h_i is the number of substrate edges traversed by the accepted embedding of request r_i. Finally, because continuity preservation is central to the proposed method, the final residual-fragmentation count is reported as the number of maximal free substrate segments that remain after the full request sequence has been processed. Lower fragmentation values are better because they indicate a larger and more usable connected remainder for future requests.

C. Quantitative results

The eight-panel comparison in Figure 9 summarizes the resource-optimization performance of the proposed method. The four columns report acceptance ratio, node utilization, link utilization, and final residual fragmentation, while the top and bottom rows correspond to the linear and ring substrates, respectively. This layout shows that the proposed strategy not only accepts requests, but also uses substrate resources in a more structured way.

Acceptance and node-utilization differences are small: the average acceptance ratios are 0.5357, 0.5340, 0.5346, and 0.5315 for EIMSDN, LFBM, Random, and CenterSpread, respectively, while EIMSDN's average node utilization is 0.9886. These marginal differences are descriptive and are not presented as statistically established superiority. The larger effects occur in the targeted continuity metrics. Relative to LFBM and Random, EIMSDN reduces average link utilization by 22.9% and 10.7% and reduces fragmentation to 0.865 from 1.010 and 1.353. CenterSpread attains lower link utilization, but its fragmentation is 1.873 and its acceptance is lower.

The mapping-cost results show the same trade-off. EIMSDN reduces average cost by about 7.1% relative to LFBM and 2.9% relative to Random. CenterSpread is 7.17% cheaper than EIMSDN and uses fewer links, but accepts fewer requests and leaves 53.83% more fragmentation relative to the proposed method. It is therefore not dominated; it favors immediate path cost, whereas EIMSDN favors residual continuity.

D. Discussion of the simulation outcomes

The aggregated relative changes reported in Table VII further confirm that the proposed method improves overall supportability and residual usability rather than optimizing a single metric in isolation.

TABLE VI
MEDIUM-LOAD RESULTS (50 REQUESTS).

Topo.	Method	AR	NU	LU	MC	Frag.
Linear	EIMSDN	0.484	0.992	0.636	7.113	0.69
	LFBM	0.482	0.989	0.814	7.632	0.96
	CenterSpread	0.478	0.978	0.451	6.616	2.01
	Random	0.483	0.985	0.682	7.227	1.36
Ring	EIMSDN	0.483	0.991	0.613	7.200	0.80
	LFBM	0.481	0.990	0.799	7.776	0.86
	CenterSpread	0.480	0.982	0.448	6.709	1.65
	Random	0.480	0.987	0.710	7.517	1.17

TABLE VII
RELATIVE BENEFIT OF EIMSDN; POSITIVE VALUES FAVOR EIMSDN.

Baseline	AR gain	NU gain	LU red.	Frag. red.	MC red.
LFBM	+0.31%	+0.08%	+22.91%	+14.36%	+7.08%
Random	+0.20%	+0.43%	+10.70%	+36.08%	+2.86%
CenterSpread	+0.79%	+1.02%	-36.61%	+53.83%	-7.17%

Note: Reduction columns are computed as $(\text{baseline} - \text{EIMSDN})/\text{baseline}$; negative values indicate that the baseline uses less of that resource.

The evaluation supports a narrower conclusion than uniform superiority: EIMSDN trades a modest acceptance and node-utilization advantage for substantially lower fragmentation than all three ablations, and for lower link use than LFBM and Random. CenterSpread remains preferable when immediate link use or mapping cost is the sole objective. External validity is limited by the 20-node synthetic path/cycle substrates. The scalar compute model, absence of wall-clock and energy measurements, and lack of multi-resource tests further limit deployment claims. The simple ablations and absence of confidence intervals prevent claims against state-of-the-art graph-VNE methods or claims of statistical significance. These limitations are consistent with positioning the method as a structured initial-placement component rather than a complete VNE solver.

VII. CONCLUSION

This paper formulated structured aggregate service embedding for online requests with scalar compute and inter-node bandwidth demands on path and cycle substrates. The model is explicitly narrower than classical graph-to-graph VNE. The boundary-contiguous method combines bandwidth-induced segmentation with lexicographic acceptance–fragmentation selection, and the revised theory establishes a node-packing normal form and restricted boundary-compaction optimality only for homogeneous demands with nonbinding intra-segment bandwidth. Its implementation requires $O(RV)$ time and $O(V)$ memory. Across 600 paired synthetic instances, acceptance and node-utilization differences were marginal, whereas EIMSDN reduced link use relative to LFBM and Random and produced the lowest residual fragmentation. CenterSpread retained lower immediate link use and cost, confirming an explicit cost–continuity trade-off. The evidence supports the method as a low-complexity initializer for ordered substrates, not as a general arbitrary-graph VNE solution. Future work should examine graph decomposition, multi-resource demands, stronger same-model baselines, per-trial statistical analysis, and controller-level runtime validation.

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.

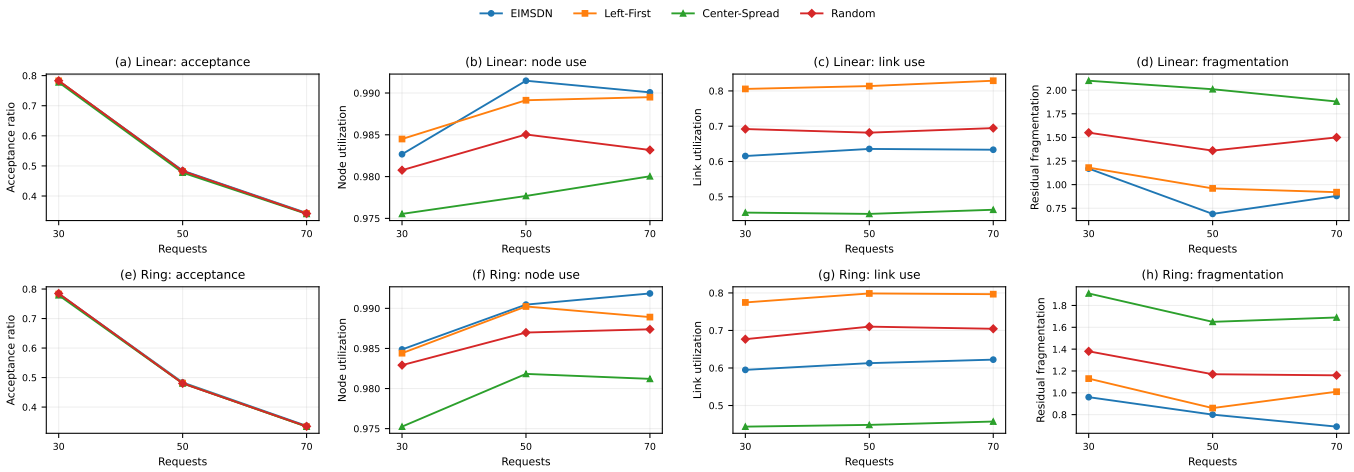


Fig. 9. Acceptance ratio, node utilization, link utilization, and final fragmentation for linear (top) and ring (bottom) substrates. The plot shows trade-offs rather than uniform dominance.

REFERENCES

[1] 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.

[2] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[3] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3930–3946, 2023.

[4] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "5G slice mutation to overcome distributed denial of service attacks using reinforcement learning," in *Proc. 17th Int. Conf. Security Inf. Netw. (SIN)*, 2024, pp. 1–9.

[5] 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.

[6] A. Satpathy, M. N. 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.

[7] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[8] H.-K. Lim, I. Ullah, Y.-H. Han, and S.-Y. Kim, "Reinforcement learning-based virtual network embedding: A comprehensive survey," *ICT Express*, vol. 9, no. 5, pp. 983–994, 2023.

[9] X. Xiao, W. Wang, S. Li, X. Zhao, and M. Guo, "DVNE-DRL: dynamic virtual network embedding algorithm based on deep reinforcement learning," *Scientific Reports*, vol. 13, Art. no. 20010, 2023.

[10] Z. Duan and T. Wang, "Towards learning-based energy-efficient online coordinated virtual network embedding framework," *Computer Networks*, vol. 239, Art. no. 110139, 2024.

[11] 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.

[12] J. Rubio-Loyola and C. Aguilar-Fuster, "Novel Initialization Functions for Metaheuristic-Based Online Virtual Network Embedding," *Journal of Network and Systems Management*, vol. 32, no. 3, Art. no. 46, 2024.

[13] M. Minardi, T. X. Vu, I. Maity, C. Politis, and S. Chatzinotas, "Traffic-Aware Virtual Network Embedding With Joint Load Balancing and Datarate Assignment for SDN-Based Networks," *IEEE Transactions on Network and Service Management*, vol. 21, no. 5, pp. 4936–4948, 2024.

[14] K. K. T. G. Kumar, S. Tomar, S. K. Addya, A. Satpathy, and S. G. Koolagudi, "EFraS: Emulated framework to develop and analyze dynamic virtual network embedding strategies over SDN infrastructure," *Simulation Modelling Practice and Theory*, vol. 134, Art. no. 102952, 2024.