TopoFormer: An Efficient Link-Set Prediction Architecture for Ad Hoc Network Topology Generation

Félix Marcoccia
Inria Paris
Thales
felix.marcoccia@inria.fr

Victor Fagoo
Thales
victor.fagoo@thalesgroup.com

Gilles Monzat
Thales
gilles@monzat.fr

Cédric Adjih
Inria Saclay
cedric.adjih@inria.fr

Thomas Watteyne
twatteyne@gmail.com

Paul Mühlethaler
Inria Paris
paul.muhlethaler@inria.fr

Abstract

In this paper, we present TopoFormer, a powerful architecture for predicting links between communication nodes in mobile networks. The goal is to imitate, in real time, the results of a costly combinatorial algorithm that generates topologies for networks with directional antennas. These antennas offer excellent performance but require complex, interdependent steering decisions in real time. Our Transformer-based architecture is enhanced with efficient components that add useful inductive biases, making it suitable for environments where scaling is limited. A key contribution is the introduction of directional density encodings, which help the attention mechanism better separate nodes in dense clusters. Equipped with our modules, a single Transformer block of dimension 12 achieves over 95 % accuracy, reducing the gap to optimality by half compared to a plain 1-block Transformer while requiring only 12 % more computation. Using two blocks, the model comes close to perfect accuracy.

1 Introduction

1.1 Link Topology Generation for Mobile Ad Hoc Networks

We address the generation of link topologies in wireless mobile networks using directional antennas. We tackle real-time link topology generation in MANETs with directional antennas, selecting links that form a bipartite backbone for antenna steering under sectorization, range, and interference constraints. The bipartiteness corresponds to the need to allocate emission and reception slots to each node, alternately. Fig. 1 illustrates our problem. While protocols like OLSR [1] and AODV [2] adapt well with omnidirectional antennas, they incur high interference; directional links boost throughput [3] but require complex, interdependent decisions. Existing approaches, UAV placement, topology control, adaptive beam/power tuning [4–8], or greedy link selection [9], are often sub-optimal or mobility-dependent. Combinatorial optimization produces quality static solutions [10, 11], but is too slow for mobility; instead, we learn to replicate these patterns from feasible topologies for fast and robust generation.

The challenge is exacerbated by the limited computational resources of mobile nodes, which must also perform signal decoding, neighbor position prediction, and topology updates. Thus, the topology predictor must be efficient to preserve resources for other network tasks. Our approach is to replace a computationally expensive algorithm with an efficient neural network, capable of producing similar outputs in real time. While prior works on neural networks for wireless systems mainly address routing and performance prediction [12–14], we focus on efficient link topology generation.

Applying autoregressive [15, 16] methods, GANs [17, 18], or VAEs [19, 20] to node-conditioned topology generation problem is not trivial [21], mostly because of the node-conditioning constraint. Diffusion-based approaches [21, 22] are more effective, but they demand substantial computation power. We rather follow a one-step supervised learning setting and enhance a Transformer, well-suited to point clouds [23, 24]—with a set of computationally efficient modules that provide useful structural inductive biases, partially avoiding the need to widen layers, which would increase computation quadratically. Embedded architectures, notably small planes or UAVs, which generally do not allow for massive parallel computation, could typically benefit from such modules.

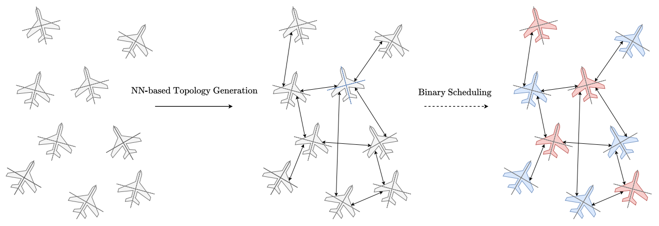

Figure 1: Given a set of aerial nodes, a neural network is used to generate a link topology. We then assign the nodes a binary emission-reception scheduling.

1.2 Message Passing and Its Expressivity

In graph learning, rich node representations can be built via message passing [25–27], graph convolutions [19, 28, 29], or attention [30–32]. When edges are missing or must be predicted, GNNs require heuristic edges to enable message passing, such as the latent-space k-NN used in [33]. However, this scheme is equivalent to local-masked attention with a fixed query vector, limiting its ability to capture long-range dependencies and non-local patterns, an essential capability for edge prediction [23, 24]. While GNNs suffer from oversmoothing due to the message-passing bottleneck [34], attention-based models face similar issues [35]: as layers accumulate, node representations become indistinguishable. This is critical in small architectures with few, thin layers.

Handcrafted node features can help but are often unsuitable here as they typically rely on predefined edges. Laplacian positional encodings [36] require a predefined edge set, Graphormer [23] encodes hop-based distances, and Graph-BERT [24] orders features by shortest-path hops. Without edges, such encodings degrade to simple Euclidean distances, scalar proximity measures that many models already recover. We address this by introducing **Directional Density Encodings (DDE)**: a node's DDE quantifies the density of neighbors along each axis of its local reference frame. In 2D, this yields a four-dimensional geometric signature capturing direction-weighted density, providing a strong relative positional cue even in edge-free settings and improving multiple link prediction.

Furthermore, we incorporate cross-attentive modulation (CAM) tokens [21] to assist the message passing mechanism in adapting its behavior to the node layout, while also enabling a form of global control. These additional modules and features allow TopoFormer to hold particularly well-suited inductive biases for the prediction of full link topologies for any node layout.

1.3 Towards more Expressive Link-Set Prediction

Pairwise link prediction generally assumes that link candidates' embeddings can be aligned pairwise. In [37], link prediction is formulated as a graph algorithm problem: deciding the existence of an edge is associated with evaluating a function over node pairs, which a message-passing mechanism can be trained to approximate.

Although expressive and capable of good generalization, this paradigm assumes that a single decoding function, generally an inner product, can capture the many different relationships that underlie the existence of a link, which we assume could be improved using a mechanism that can model the different relations that can qualify a pair of nodes to form a link. The feature aggregator must then learn to precisely align nodes pairwise in the latent space to predict links, while having to capture complex patterns and global properties to ensure that these predicted links are correct. In the context of mobile networks, the heterogeneous nature of the topologies combined with the need for small and shallow neural architectures tend to challenge the capabilities of the feature aggregator.

As a simple way to enhance the capabilities of the prediction layer, we equip TopoFormer with a multichannel factorization method that can be interpreted as a flexible generalization of RESCAL [38] or TuckER [39] for one-dimensional link prediction. In our case, static relation-specific matrices are replaced by learned parameters that perform factorization over an arbitrary number of dimensions. It allows us to enhance the expressivity of the prediction layer while computationally scaling linearly with the number of channels.

2 Problem Description

Our work is motivated by a network topology problem that consists in finding an optimal set of links given a set of nodes described by their positions. It corresponds to a typical combinatorial problem that optimizes the link topology of a wireless network with directional antennas. The topologies feature a small (<32) number of nodes.

We consider graphs with $n \in [16, 32]$ nodes V, where each node $v \in V$ is specified by 2D coordinates $\mathbf{x}_v \in \mathbb{R}^2$. Given these inputs, the model outputs an adjacency matrix $E \in \{0,1\}^{n\times n}$ encoding the predicted links (with $E_{ij} = 1$ iff a link from i to j is present).

Valid topologies are defined by several constraints and properties:

- Each node may have at most four links.
- There are strict link sectorization constraints regarding the placement of antennas.
- Interference between links must be minimized, which implies avoiding acute angles between emitting nodes and their unintended receiving neighbor nodes.
- The link topology must be connected.

The problem cannot be trivially linearized and exhaustive methods are too heavy to be carried out in any reasonable amount of time, even to create a dataset. We therefore rely on a heuristic algorithm (Alg. 1) that follows:

Algorithm 1 Greedy Topology Generation

```
Require: $V$, $d(\cdot,\cdot)$ the distance function, antenna sets $\{\mathcal{A}_i\}$, sector map $\{\mathcal{Q}_i\}$ (e.g. NE/NW/SE/SW), threshold $\tau$, parity $s : V \to \{0, 1, \perp\}$, throughput model $T(\cdot)$ (simplified in App. 6.1)
Ensure: $G = (V, E)$, updated parity $s$
 1: $E \leftarrow \emptyset$
 2: $P \leftarrow \{\{i,j\} \subseteq V \mid (s_i \neq s_j) \lor (s_i = \perp \lor s_j = \perp)\}$
 3: order $P$ by ascending $d(i,j)$
 4: while $(V,E)$ not connected and $P \neq \emptyset$ do
 5:     take shortest pair $\{i,j\}$ from $P$
 6:     if $s_i, s_j \neq \perp$ and $s_i = s_j$ then continue
 7:     if $s_i, s_j \neq \perp$ and $s_i \neq s_j$ then        ▷ direction fixed: TX $s$=0 → RX $s$=1
 8:         $(a^\star, b^\star, t^\star) \leftarrow \arg\max_{a\in\mathcal{A}_i,\, b\in\mathcal{A}_j}\ t(i\to j, a, b \mid E)$
 9:     else        ▷ at least one $s = \perp$
10:         $(a^\star, b^\star, t^\to) \leftarrow \arg\max_{a,b}\ t(i\to j, a, b \mid E)$
11:         $(\tilde{a}^\star, \tilde{b}^\star, t^\leftarrow) \leftarrow \arg\max_{a,b}\ t(j\to i, a, b \mid E)$
12:         if $t^\to \geq t^\leftarrow$ then
13:             fix $s_i \leftarrow 0$ (if $\perp$), $s_j \leftarrow 1$ (if $\perp$);   $t^\star \leftarrow t^\to$
14:         else
15:             fix $s_j \leftarrow 0$ (if $\perp$), $s_i \leftarrow 1$ (if $\perp$);   $(a^\star, b^\star) \leftarrow (\tilde{a}^\star, \tilde{b}^\star)$;   $t^\star \leftarrow t^\leftarrow$
16:     sector constraint: let $q_i(a^\star) \in \mathcal{Q}_i$, $q_j(b^\star) \in \mathcal{Q}_j$; require $\deg_{q_i}(i) \leq 0$ and $\deg_{q_j}(j) \leq 0$
17:     if $t^\star \geq \tau$ and sector constraint holds then
18:         $E \leftarrow E \cup \{(i,j)\}$        ▷ activate with $(a^\star, b^\star)$ and fixed direction
19:         lock sectors: increment $\deg_{q_i}(i)$, $\deg_{q_j}(j)$
20: return $(V, E, s)$
```

The nodes follow realistic trajectories representative of small fleets of manned aircraft. The generation algorithm evaluates throughput and interference with a high-fidelity simulation of signal propagation and antenna characteristics; a simplified formulation is given in Appendix 6.1.

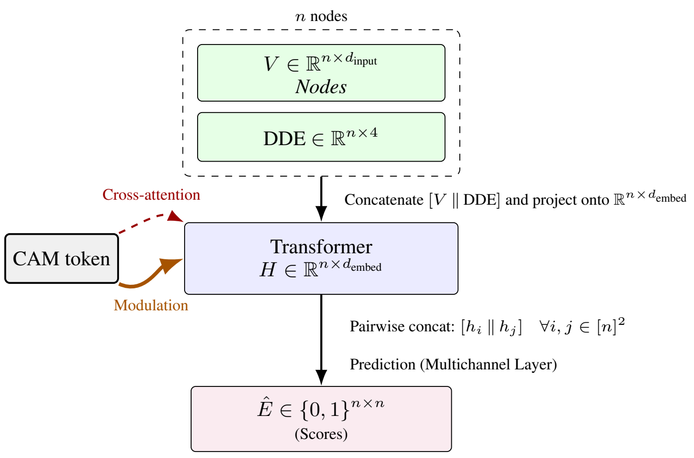
n nodes
$V \in \mathbb{R}^{n \times d_{\text{input}}}$
Nodes
DDE $\in \mathbb{R}^{n \times 4}$
Concatenate $[V \parallel \text{DDE}]$ and project onto $\mathbb{R}^{n \times d_{\text{embed}}}$
Cross-attention
CAM token
Modulation
Transformer
$H \in \mathbb{R}^{n \times d_{\text{embed}}}$
Pairwise concat: $[h_i \parallel h_j] \quad \forall i, j \in [n]^2$
Prediction (Multichannel Layer)
$\hat{E} \in \{0,1\}^{n \times n}$
(Scores)

Figure 2: The architecture of TopoFormer.

The topologies obtained achieve about 95 % of the throughput obtained using an exact and exhaustive algorithm (too slow even for data generation), they allow for a **1.7** times higher theoretical throughput than if using a standard omnidirectional protocol topology for a 16-node-network, and a **3.7** factor of improvement for a 32-node-network. The algorithm is at least $\mathcal{O}(m^2)$ complex even without considering implicit parity checks, m being the number of edges. It is not suitable for real-time use, but can be used to generate a dataset. Indeed, while its execution time is approximately one second for 16-node-instances on a modern CPU, it may increase when the initial iterations fail to naturally converge to viable solutions and explodes when dealing with more than 30 nodes. Imitating such graphs with a neural network demands both global control capabilities in order to provide plausible large-scale connectivity patterns and the ability to model the different kinds of relationship between nodes that can lead to the presence of a link. Accurately reproducing the patterns and construction rules of such dataset graphs would enable finding high-performance complex topologies in real time.

3 Our Architecture

In this section, we detail the neural architecture that we adopt in order to imitate the results of our costly algorithm.

3.1 Overview of Our Architecture

Inspired by the effectiveness of attention-based models to deal with point clouds [23, 24], we adopt the Transformer [30] as the backbone of our architecture. We implement it with no sequence-based encodings [30], in order to keep permutation invariance, nor spectral [36] positional encoding, since the edges are the object we seek to predict. Given the limited width of the layers in TopoFormer, a trade-off arises between using a small number of attention heads or assigning a low dimensionality to each head. We opt for a larger number of lightweight (1- or 2-dimensional) attention heads. Our empirical results presented in Table 3 justify this choice.

The nodes are enriched with directional density encodings, which we detail in Sec. 3.2. The model is equipped with multichannel prediction layers, as introduced in Sec. 3.3.

To enhance its adaptability to heterogeneous topologies, we augment it with CAM tokens [21], which have been designed to facilitate the generation of valid network topologies with respect to the global layout of the nodes. They provide dynamic modulation of the backbone model's behavior based on the overall node layout, captured through cross-attention, without overloading or biasing the self-attention block. They allow the model to efficiently adapt its behavior to the heterogeneous nature of network topologies. Such tokens are inspired by global tokens such as CLS [40] tokens or registers [41], while their modulation process consists in the conditioning of a FiLM [42] layer. Such tokens allow us to capture global patterns without directly increasing the dimension of the attention mechanism, with a linear complexity scaling with respect to the number of nodes.

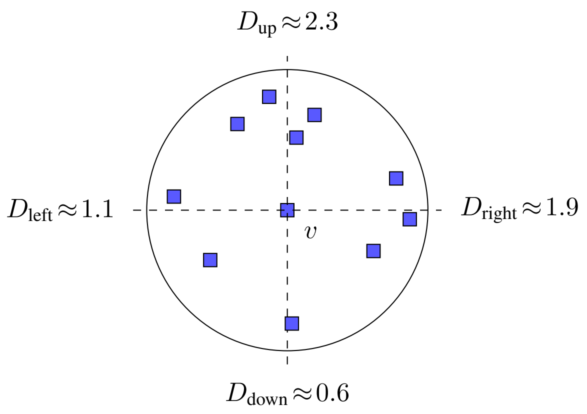
$D_{\text{up}} \approx 2.3$
$D_{\text{left}} \approx 1.1$
$D_{\text{right}} \approx 1.9$
v
$D_{\text{down}} \approx 0.6$

Figure 3: Illustration of the computation of DDEs for a node v.

We choose to condition the modulation solely on the CAM token representation, as this proved to be more effective in our experiments under a low-parameter regime. This contrasts with the original formulation, which applies modulation through an elementwise product between each node embedding and the CAM token value.

Let $H \in \mathbb{R}^{N\times d}$ denote the node embeddings at layer ℓ, and $H^{\ell}_{\text{CAM}} \in \mathbb{R}^{1\times d}$ the corresponding CAM token representation. The modulation is computed as follows:

$$H^{\ell}_{\text{CAM}} \leftarrow \text{CrossAttn}\big(H^{\ell-1}_{\text{CAM}}, H, H\big), \quad \gamma^{\ell}, \beta^{\ell} \leftarrow \text{FFN}\big(H^{\ell}_{\text{CAM}}\big),$$
$$H^{\text{updated}} = \gamma^{\ell} \odot H + \beta^{\ell}.$$

Here, $\odot$ denotes the Hadamard product. The feed-forward network (FFN) is implemented as a simple sequence of linear layers with ReLU activations. This formulation allows the CAM token to globally summarize the graph state and to generate adaptive modulation parameters $(\gamma^{\ell}, \beta^{\ell})$ applied uniformly across all node embeddings.

Fig. 2 shows the architecture of our framework.

3.2 Directional Density Encoding

We equip TopoFormer with directional density encodings in order for the nodes to be easily distinguishable by the attention mechanism, especially when they are located within dense clusters. Fig. 3 shows an illustration of the computation of DDEs for an arbitrary node.

Let v have position $\mathbf{x}_v = (x_v, y_v) \in \mathbb{R}^2$ and define $\Delta_{uv} = \mathbf{x}_u - \mathbf{x}_v$, $r_{uv} = \|\Delta_{uv}\|$, $\hat{\Delta}_{uv} = \Delta_{uv}/\max(r_{uv}, \varepsilon)$ with $\varepsilon > 0$, and $K(r) = \exp(-r^2/(2\sigma^2))$.

Using $[z]_+ = \max(0, z)$, set

$$\begin{aligned} D_{\text{right}}(v) &= \sum_{u\neq v} K(r_{uv})\,[\,\hat{\Delta}_{uv,x}\,]_+, \qquad & D_{\text{left}}(v) &= \sum_{u\neq v} K(r_{uv})\,[-\hat{\Delta}_{uv,x}]_+, \\ D_{\text{up}}(v) &= \sum_{u\neq v} K(r_{uv})\,[\,\hat{\Delta}_{uv,y}\,]_+, \qquad & D_{\text{down}}(v) &= \sum_{u\neq v} K(r_{uv})\,[-\hat{\Delta}_{uv,y}]_+. \end{aligned} \tag{1}$$

One can also derive a learnable and continuous generalization of DDEs to an arbitrary number of dimensions by introducing H learnable directional vectors, $\{\mathbf{q}_h\}_{h=1}^{H} \subset \mathbb{R}^d$, which act as heads similar to attention, without a softmax. For a given node v with position $\mathbf{x}_v \in \mathbb{R}^d$, the h-th directional density component is computed as:

$$D_h(v) = \sum_{\substack{u\in V \\ u\neq v}} \phi\left(\|\mathbf{x}_u - \mathbf{x}_v\|\right) \cdot \mathbf{q}_h^{\top}\left(\mathbf{x}_u - \mathbf{x}_v\right) \tag{2}$$

where:

- $\mathbf{q}_h \in \mathbb{R}^d$ is a learnable direction vector (the h-th head),
- $\mathbf{x}_v, \mathbf{x}_u$ are the latent positions of nodes v and u respectively,
- $\phi : \mathbb{R}^+ \to \mathbb{R}^+$ is a distance-based weighting function, such as $\phi(r) = \exp(-r^2/2\sigma^2)$.

The resulting vector $\mathbf{d}(v) = [D_1(v), \dots, D_H(v)] \in \mathbb{R}^H$ encodes the anisotropic density of neighboring nodes around v along each learned direction. Unlike standard attention mechanisms, this formulation does not use a softmax normalization, allowing each head to accumulate density information instead of normalizing it, thereby preserving both directionality and local concentration, allowing them to "count" easily.

Throughout this paper, we prefer the non-learnable version of DDEs, as it is much more computationally efficient.

In practice, because the Gaussian weight $\phi(r) = \exp\big(-r^2/2\sigma^2\big)$ decays rapidly, only nodes lying within a fixed radius $r \approx 3\sigma$ contribute non-negligibly to the directional density encoding (DDE). To avoid inspecting all $\sim N^2$ pairs, we first insert the N node positions into a kd-tree [43], a binary space-partitioning structure that recursively splits the point set along coordinate axes, yielding a balanced search tree of depth $\mathcal{O}(\log N)$. A radius query on this tree visits only the buckets intersecting the hypersphere of radius r around a query node and therefore returns, on average, a constant number k of nearby neighbours, independent of N for bounded density. With k treated as a small constant, the overall complexity is $\mathcal{O}(N \log N)$, and the aggregation of the four DDE components adds only an $\mathcal{O}(Nk)$ linear pass over the reported neighbours. Hence the truncated scheme retains nearly all the relevant mass while reducing the theoretical cost from quadratic to near-linear.

3.3 Multichannel scoring mechanisms.

To model the diversity of structural factors that can lead to link formation, ranging from local motifs to more global topological cues, we explore two bilinear scoring strategies inspired by relational representation learning. Fig. 4 illustrates the principle of such multichannel prediction layers.

First, we introduce a diagonal bilinear formulation using learnable channels $\{c_i\}_{i=1}^C \subset \mathbb{R}^d$ that interact with the pair embedding $p \in \mathbb{R}^d$ via element-wise (Hadamard) multiplication. Each pair p is obtained using the pairwise concatenation of the nodes. Each channel produces a score through a shared projection, the score of an edge prediction $\hat{e}$ corresponding to the node pair p is hence given by:

$$\hat{e} = \sum_{i=1}^{C} w^\top \, \mathrm{LN}(c_i \odot p), \tag{3}$$

where w is a common weight vector and LN denotes a LayerNorm applied prior to aggregation. This mechanism corresponds to a multichannel, low-rank-factorization-like of a diagonal bilinear form:

$$\hat{e} = \sum_{i=1}^{C} \sum_{k=1}^{d} w_k \, \mathrm{LN}(c_{i,k} p_k). \tag{4}$$

While this Hadamard-channels formulation is not strictly more expressive than the neurons of a simple linear layer, it offers several practical advantages. In particular, it enforces a channel-wise separation of parameters that improves optimization, keeps channel–pair Hadamard products in structured form longer before summation, which stabilizes training and preserves richer interactions.

Following the same general idea, we also implement a dense bilinear variant faithful to a RESCAL [38] formulation, where K relation-specific matrices $R^{(k)} \in \mathbb{R}^{d \times d}$ interact with the pair embedding following the equation:

$$\hat{e} = \sum_{k=1}^{K} \tilde{p}^\top R^{(k)} \tilde{p}, \quad \tilde{p} = \mathrm{LN}(p), \tag{5}$$

This dense formulation enables the model to capture complex cross-dimensional interactions and more nuanced structural dependencies. However, throughout the paper, we adopt the diagonal multichannel formulation as our default, since it offers nearly equivalent performance while being significantly more efficient and stable to train. These learned channels could also be parametrized by a feature aggregator to dynamically adapt to different or more complex network settings.

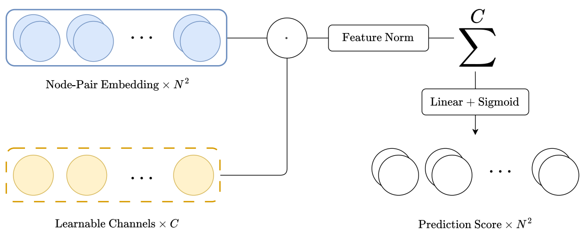

Figure 4: Learnable channel values are multiplied with the node-pair embeddings. The result is summed over the channels and scaled before being passed to a sigmoid. " · " denotes either a Hadamard product or a matrix multiplication, depending on the model version, between each channel with each node pair.

4 Results

We conduct our experiments with an Intel Xeon(R) E5-2650v3 @ 2.30 GHz CPU and a Tesla T4 GPU. Using only CPU, TopoFormer runs in less than 100 milliseconds. The models are trained using standard binary cross-entropy and are equipped with one Transformer block unless stated otherwise. The prediction layer consists in the concatenation of node pairs fed into a small MLP. In addition to attention-based baselines, we benchmark a graph neural network in which virtual edges connect each node to the neighbors in its communication range. The convolutional variant can also be seen as a 1-layer DCGNN [33]. The model described as TopoFormer is a Transformer equipped with CAM tokens, DDEs and a multichannel prediction layer. The uncertainty is computed as the standard deviation over bootstrapped measurements. The 2-block TopoFormer also follows a second disjoint CAM block iteration, in addition to the supplementary Transformer block. The embedding size is $d = 12$, which corresponds to the smallest dimension yielding satisfactory accuracy. CAM-enhanced ablations use a single CAM token.

Training uses the AdamW optimizer [44]. The learning rate linearly decays from 10^{-3} to 10^{-6} throughout training, while the weight decay is set to 10^{-3} and disabled during the final 20 epochs. Models are trained using batch size 64 for at most 4700 epochs with random seed 123 for reproducibility. Early stopping based on validation loss is applied when convergence is reached, and the model achieving the highest validation accuracy is reported in bold.

4.1 Accuracy

Table 1: Accuracy of different ablations and models on our test set. The FLOPs column indicates the relative percentage of computational cost compared to that of the standard 1-block Transformer.

Method	**Accuracy (%)**	**FLOPs**
Graph Convolutional Network [45]	87.4±.4	-14%
Graph Attention Network [31]	88.1±.4	-8%
Graph Transformer [32]	89.9±.3	+8%
Transformer	90.9±.2	Reference
Transformer w/ CAM	92.9±.2	+7%
Transformer w/ DDE	93.7±.2	+3%
TopoFormer w/ Multichannel Hadamard	95.1±.2	+12%
TopoFormer w/ Multichannel Factorization	**95.2±.2**	+14%
2-block Transformer	99.0±.1	+89%
TopoFormer w/ 2 blocks	**99.9±.1**	+100%

Table 1 shows that TopoFormer outperforms the baseline Transformer, and significantly outclasses a range of GNN models. The results underline that DDEs have a high impact, and that the rest of the architecture enhancements also bring about an improvement for a low computational overhead. The bilinear multichannel prediction offers the best accuracy but seems less computation-efficient than the Hadamard-based version. Attention-based models perform better than locality-based graph models.

Table 2: Comparison of the results of our Hadamard multichannel prediction layer with the pairwise linear prediction, both without any backbone network. *Variance* is used as a proxy for the ability to clearly separate 0s and 1s.

Prediction Layer	**Accuracy (%)**	**Variance**
Pairwise Linear Layer	79.42±.5	0.038
Large Pairwise Linear Layer	82.89 ±.5	0.057
Multichannel Hadamard	**86.91±.5**	**0.086**

Table 3: Exploration of the head number/head size trade-off for our model.

Ablation	**Accuracy (%)**
One 12D head	94.07±.3
Two 6D heads	94.46±.3
Three 4D heads	94.81±.3
Six 2D heads	**95.16±.2**

Learnable DDEs were less computationally efficient than their handcrafted counterpart, their results are not showcased for clarity. Please note that, in a Transformer model, increasing the embedding dimension from 12 to 14, the next non-prime number (required for multi-head attention), would raise the number of FLOPs required by roughly 27%.

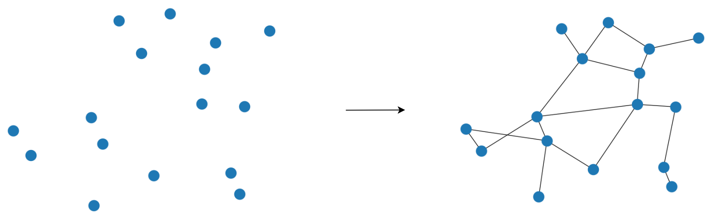

Figure 5: Example of a neural-network-generated 16-node topology.

Fig. 5 shows an example of a generated topology using our architecture.

In Table 2, we isolate the effect of the prediction layers by measuring the accuracy on our prediction task without using any complex backbone network, with the nodes simply being fed to a small node-wise MLP. The *Large Pairwise Linear Layer* is made k times larger than the *Pairwise Linear Layer*, k being the number of channels in the *Multichannel Hadamard* in order to verify that the sheer number of parameters is not enough to explain the performance increase. The Hadamard-based multichannel layer appears to possess some intrinsic expressivity that allows it to discriminate links better than simple linear pairwise matching. Table 3 shows that equipping the backbone Transformer with numerous but small attention heads yields better accuracy than fewer but larger attention heads.

4.2 Throughput and Application

In order to be operational, the generated topologies should be post-processed following Alg.2, which is on average is $\mathcal{O}(m \, log \, m)$ and runs in less than 300 milliseconds, which yields a significant speed-up as compared to the original greedy algorithm.

In Table 4, we report theoretical throughputs as upper bounds on the simultaneous information exchange in the network. Topologies are obtained through the post-processing algorithm 2, which derives a parity-based communication scheme, throughput is computed following the computations simplified in App. 6.1, mainly reflecting link density and interference. The omnidirectional baseline uses OLSR-like topologies with a communication range of about one quarter of the network

Algorithm 2 Post-Processing and Parity Assignment

```
Require: $G_{\text{pred}} = (V, E_{\text{pred}})$, parity $s : V \to \{\text{TX}, \text{RX}, \text{UNCOLORED}\}$, sectors $\{\mathcal{Q}_i\}$, model
    $T(\cdot)$
Ensure: $G = (V, E)$ bipartite (TX→RX), sector-feasible, connected
 1: $E \leftarrow E_{\text{pred}}$
        ▷ (A) Sector pruning: at most one edge per sector and node, preserve connectivity
 2: for all $v \in V$ do
 3:     for all $q \in \mathcal{Q}_v$ do
 4:         while $|\{(v,u) \in E : u \in q\}| > 1$ do
 5:             pick edge $(v,u)$ with lowest per-link score $t(v,u \,|\, E)$
 6:             if $(V, E \setminus \{(v,u)\})$ remains connected then
 7:                 $E \leftarrow E \setminus \{(v,u)\}$
 8:             elsebreak
        ▷ (B) Bipartite projection via 2-coloring (TX/RX) and removal of monochromatic edges
 9: initialize $s(v) \leftarrow \text{UNCOLORED}$ if undefined
10: for all each connected component of $(V, E)$ do
11:     pick root $r$, set $s(r) \leftarrow \text{TX}$, push $r$ on stack $S$
12:     while $S \neq \emptyset$ do
13:         $u \leftarrow \textsc{Pop}(S)$
14:         for all $(u,v) \in E$ do
15:             if $s(v) = \text{UNCOLORED}$ then
16:                 $s(v) \leftarrow \text{opposite}(s(u))$, push $v$
17: $bad \leftarrow \{(u,v) \in E : s(u) = s(v)\}$        ▷ monochromatic
18: sort $bad$ by ascending $t(u,v \,|\, E)$
19: for all $e \in bad$ (in order) do
20:     if $(V, E \setminus \{e\})$ remains connected then
21:         $E \leftarrow E \setminus \{e\}$
22: return $(V, E)$
```

Table 4: Avg instantaneous throughput upper bound (Mbps)

	OLSR-like Omni.	Huang et al.	MST+greedy	Transformer	TopoFormer	Target
Throughput ↑	47.24	52.89	63.46	69.36	**74.77**	79.33

diameter. Our approach halves the gap between Transformer and target throughput, and significantly outperforms the omnidirectional case by leveraging directional antennas. Since the target algorithm achieves 95 % of optimal throughput, TopoFormer reaches ∼90 % with one block (and nearly yields the same throughput as the target's with two blocks). We observed that approximately 80 % of the throughput improvement over the standard Transformer can be attributed to reduced interference. Topologies produced by the method of Huang et al. [7], with parity scheduling assigned ex post, reach about one-third lower throughput than the target, primarily due to residual interference and limited antenna coordination. The MST+greedy baseline, which extends a minimum spanning tree with additional links under antenna constraints, achieves about 19% lower throughput. Their average execution times are approximately 700 ms and 1 s on CPU, respectively. The geometric interference management in our topologies enables high network capacity, whereas omnidirectional topologies are limited by local congestion, resulting in a striking difference in throughput.

5 Conclusion

We presented TopoFormer, a Transformer-based architecture to infer link sets from node sets, mimicking a complex network algorithm for steering directional antennas in mobile ad hoc networks. Efficient modules enhance expressivity without enlarging the model, which only requires minimal post-processing allowing for a fast scheduling enabling high network capacity.

References

[1] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings. IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century.*, pages 62–68, 2001. 1

[2] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999. 1

[3] Su Yi, Yong Pei, and Shivkumar Kalyanaraman. On the capacity improvement of ad hoc wireless networks using directional antennas. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '03, pages 108–116, New York, NY, USA, June 2003. Association for Computing Machinery. 1

[4] Antonio Guillen-Perez and Maria-Dolores Cano. Flying ad hoc networks: A new domain for network communications. *Sensors*, 18(10):3571, oct 2018. 1

[5] Zhenming Wang, Qin Zhang, and Hai Li. UAV mobile control strategy based on potential field method. In Tao Lei and Dehai Zhang, editors, *Sixth International Conference on Information Science, Electrical, and Automation Engineering (ISEAE 2024)*, volume 13275, page 132753L. International Society for Optics and Photonics, SPIE, 2024.

[6] Zhuochuan Huang and Chien-Chung Shen. A comparison study of omnidirectional and directional mac protocols for ad hoc networks. volume 1, pages 57–61 vol.1, 12 2002.

[7] Zhuochuan Huang, Chien-Chung Shen, Chavalit Srisathapornphat, and Chaiporn Jaikaeo. Topology control for ad hoc networks with directional antennas. 12 2002. 9

[8] Qing Li and Imrich Chlamtac. A framework for topology control protocol using directional antennas in wireless networks. In *Proceedings of the IEEE International Conference on Communications*, volume 5, pages 2958–2962. IEEE, 2005. 1

[9] Lichun Bao and J.J. Garcia-Luna-Aceves. Transmission scheduling in ad hoc networks with directional antennas. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, MobiCom '02, page 48–58, New York, NY, USA, 2002. Association for Computing Machinery. 1

[10] Wei Feng, Yong Li, Depeng Jin, Li Su, and Sheng Chen. Millimetre-Wave Backhaul for 5G Networks: Challenges and Solutions. *Sensors*, 16(6):892, June 2016. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute. 1

[11] Amal Benhamiche, Wesley da Silva Coelho, and Nancy Perrot. Routing and Resource Assignment Problems in Future 5G Radio Access Networks. June 2019. 1

[12] Krzysztof Rusek, José Suárez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert Cabellos-Aparicio. RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN. *IEEE Journal on Selected Areas in Communications*, 38(10):2260–2270, October 2020. Conference Name: IEEE Journal on Selected Areas in Communications. 1

[13] Abdelhadi Azzouni, Raouf Boutaba, and Guy Pujolle. Neuroute: Predictive dynamic routing for software-defined networks, 2017.

[14] Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Krzysztof Rusek, Shihan Xiao, Xiang Shi, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Routenet-fermi: Network modeling with graph neural networks. *IEEE/ACM Transactions on Networking*, 31(6):3080–3095, December 2023. 1

[15] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models, 2018. 2

[16] Petar Veličković, Lars Buesing, Matthew C. Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks, 2020. 2

[17] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs, 2022. 2

[18] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets, 2017. 2

[19] Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016. 2

[20] Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders, February 2018. arXiv:1802.03480 [cs]. 2

[21] Félix Marcoccia, Cédric Adjih, and Paul Mühlethaler. Netdiff: Deep graph denoising diffusion for ad hoc network topology generation, 2024. 2, 4

[22] Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization, 2023. 2

[23] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021. 2, 4

[24] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations, 2020. 2, 4

[25] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017. 2

[26] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

[27] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks, 2017. 2

[28] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017. 2

[29] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018. 2

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. arXiv:1706.03762 [cs]. 2, 4

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. 7

[32] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021. 2, 7

[33] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2019. 2, 7

[34] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks, 2023. 2

[35] Xinyi Wu, Amir Ajorlou, Zihui Wu, and Ali Jadbabaie. Demystifying oversmoothing in attention-based graph neural networks, 2024. 2

[36] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks, 2022. 2, 4

[37] Andrew Dudzik and Petar Veličković. Graph neural networks are dynamic programmers, 2022. 2

[38] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA, 2011. Omnipress. 3, 6

[39] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. 3

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 4

[41] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2023. 4

[42] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. 4

[43] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. In *Communications of the ACM*, volume 18, pages 509–517, 1975. 6

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 7

[45] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. 7

6 Appendix

6.1 Sum-Rate Throughput and Interference

Parity constraint. Nodes are partitioned into V_0 and V_1 via parities $s : V \to \{0, 1\}$. We only allow links across the partition: if $(i, j) \in E$ then $s_i \neq s_j$. Directions follow parity (TX: s=0 to RX: s=1).

Sum-rate objective. The per-graph throughput is:

$$\begin{aligned} T \;=\; &\sum_{\substack{(i,j)\in E\\ s_i=0,\, s_j=1}} e_{ij}\;\log_2\left(1+\frac{\frac{R(i,j)}{d(i,j)^2}}{\text{noise}+\displaystyle\sum_{\substack{(k,l)\in E\\ k\neq i,\, s_k=0,\, s_l=1}} e_{kl}\ \text{interference}\big((i,j),(k,l)\big)}\right) \\ &+\sum_{\substack{(i,j)\in E\\ s_i=0,\, s_j=1}} e_{ij}\;\log_2\left(1+\frac{\frac{R(j,i)}{d(i,j)^2}}{\text{noise}+\displaystyle\sum_{\substack{(k,l)\in E\\ k\neq j,\, s_k=1,\, s_l=0}} e_{kl}\ \text{interference}\big((j,i),(k,l)\big)}\right). \end{aligned} \tag{6}$$

For every possible link $e_{ij} \in E$, define:

- $d(i, j)$: Euclidean distance between nodes i and j,
- $R(i, j)$: transmission power, adapted to the distance between i and j (provided by an external power-control algorithm),
- interference$((i, j), (k, l))$:

$$\text{interference}((i,j),(k,l)) \;=\; \frac{R(k,l)}{d(k,j)^2+\epsilon}\,\mathbf{1}(\angle((k,l),(k,j)) \leq \theta_{\max})\,, \tag{7}$$

  where $\mathbf{1}(\cdot)$ equals 1 if the angular separation is $\leq \theta_{\max}$ and 0 otherwise; $\epsilon > 0$ prevents division by zero and bounds near-field gains,
- $\delta(i, a)$: the set of links that must use node i's antenna sector a if selected (this set can change with positions/orientations).

Exact simulation parameters remain outside the scope of this work, as they involve the physical modeling of particular, possibly proprietary antenna patterns and behaviors, as well as precise propagation environment modeling.

6.2 General Directional Density Encoding (2D/3D)

For v with position $\mathbf{x}_v \in \mathbb{R}^d$ ($d \in \{2, 3\}$), let $\Delta_{uv} = \mathbf{x}_u - \mathbf{x}_v$, $r_{uv} = \|\Delta_{uv}\|$, and $\hat{\Delta}_{uv} = \Delta_{uv}/\max(r_{uv}, \varepsilon)$ (unit direction; $\varepsilon > 0$). Let the set of axis directions be

$$\mathcal{S}_2 = \{+\mathbf{e}_x, -\mathbf{e}_x, +\mathbf{e}_y, -\mathbf{e}_y\} \quad \text{(2D)}, \qquad \mathcal{S}_3 = \mathcal{S}_2 \cup \{+\mathbf{e}_z, -\mathbf{e}_z\} \quad \text{(3D)}.$$

With a radial kernel $K(r) = \exp\big(-r^2/(2\sigma^2)\big)$, the directional density vector is

$$\mathbf{d}(v) \;=\; \sum_{u\in V\setminus\{v\}} K(r_{uv})\big[\,[\langle\hat{\Delta}_{uv}, \mathbf{s}\rangle]_+\,\big]_{\mathbf{s}\in\mathcal{S}_d}, \quad \text{where } [z]_+ = \max(0, z).$$

In 2D this yields the $(\text{right}, \text{left}, \text{up}, \text{down})$ components; in 3D it extends to $(\pm x, \pm y, \pm z)$.

We can then compute the DDEs for a 3D point cloud:

Let v have position $\mathbf{x}_v = (x_v, y_v, z_v) \in \mathbb{R}^3$ and define $\Delta_{uv} = \mathbf{x}_u - \mathbf{x}_v$, $r_{uv} = \|\Delta_{uv}\|$, $\hat{\Delta}_{uv} = \Delta_{uv} / \max(r_{uv}, \varepsilon)$ with $\varepsilon > 0$, and $K(r) = \exp\big(-r^2/(2\sigma^2)\big)$. Using $[z]_+ = \max(0, z)$:

Vector form (6 directions):

$$\mathbf{d}^{(3\mathrm{D})}(v) = \sum_{u \neq v} K(r_{uv}) \Big([\hat{\Delta}_{uv,x}]_+,\ [-\hat{\Delta}_{uv,x}]_+,\ [\hat{\Delta}_{uv,y}]_+,\ [-\hat{\Delta}_{uv,y}]_+,\ [\hat{\Delta}_{uv,z}]_+,\ [-\hat{\Delta}_{uv,z}]_+ \Big).$$

Expanded components:

$$\begin{aligned} D_{+x}(v) &= \sum_{u \neq v} K(r_{uv})\, [\hat{\Delta}_{uv,x}]_+, & D_{-x}(v) &= \sum_{u \neq v} K(r_{uv})\, [-\hat{\Delta}_{uv,x}]_+, \\ D_{+y}(v) &= \sum_{u \neq v} K(r_{uv})\, [\hat{\Delta}_{uv,y}]_+, & D_{-y}(v) &= \sum_{u \neq v} K(r_{uv})\, [-\hat{\Delta}_{uv,y}]_+, \\ D_{+z}(v) &= \sum_{u \neq v} K(r_{uv})\, [\hat{\Delta}_{uv,z}]_+, & D_{-z}(v) &= \sum_{u \neq v} K(r_{uv})\, [-\hat{\Delta}_{uv,z}]_+. \end{aligned}$$