LINKS ARE TOKENS AS NODES ARE: INCIDENCE-GRAPH TRANSFORMERS WITH EDITABLE WALK ENCODINGS FOR LINK PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Two gaps have kept the random-walk positional encoding (RWPE), now a standard input to graph transformers (GTs) on other tasks, out of the ones built for link prediction (LP). First, RWPE is defined at a vertex, and the object to be scored, an edge, has not been given one. Second, the walk probabilities around a pair of vertices depend on whether the pair has an edge in the graph, so a model either reads that edge's presence off its encodings, which fails to generalize, or recomputes them for every query, which fails to scale. We close both gaps with incidence RWPE and editable RWPE. We prove that on the incidence graph, whose vertices are the vertices and the edges of the original graph, the walk with each edge given its two orientations takes one step of the original walk every two steps, so every vertex and every edge around a query carries an RWPE of its own class, computed from powers of the original graph's operator alone: incidence RWPE. We also prove that the common-neighbor heuristics are closed-form functions of these encodings. Links are then tokens as nodes are, the query edge among them, scored by a transformer with no message passing, no topological attention mask, and no node identifiers. Editable RWPE, a rank-two correction, reads every encoding on the graph the query is scored under from encodings computed once per graph, at a cost per token quadratic in the walk length and independent of the graph's size. With fewer than five million parameters, Vephormer, our Vertex-Edge Graph Transformer, exceeds every published result on ogbl-ppa and matches or exceeds every GT on all six benchmarks. Ablations find the context tokens worth nearly a quarter of the model's score on ogbl-ppa, and a message-passing after every block largely redundant.

1 INTRODUCTION

A link predictor scores a pair of vertices on whether the graph should be completed with an edge between them, and the structure around the pair is its strongest evidence. The heuristics that count the pair's common neighbors with one weighting or another, such as common neighbors (CN), Adamic–Adar (AA), and resource allocation (RA), are strong predictors on their own (Liben-Nowell & Kleinberg, 2007; Adamic & Adar, 2003; Zhou et al., 2009) (definitions in Appendix A), and the neural predictors that lead the benchmarks build on those counts or on labeled subgraphs around the pair (Section 7). In each, structure reaches the model as a count, a label, or an adjacency row supplied beside the vertices. None gives a vertex or an edge the structural or positional encoding that is common in graph transformers on other tasks (Rampášek et al., 2022; Müller et al., 2024; Black et al., 2024).

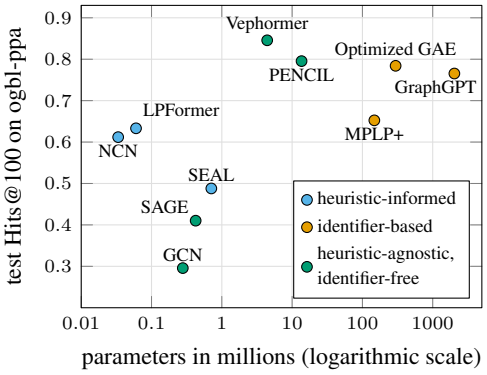


Figure 1: Test Hits@100 on ogbl-ppa against parameter count. Our model Vephormer has 4.42 million parameters. Full parameter counts can be found in Table 9.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Two gaps. Graph transformers locate their tokens by random-walk positional encodings (RWPE), the diagonal entries of successive walk matrices giving a vertex its position (Dwivedi et al., 2022; Rampášek et al., 2022) and the off-diagonal entries giving a pair of vertices their relation (Ma et al., 2023). Link prediction has adopted neither that encoding nor the edge token, for two reasons. A random-walk encoding is defined at a vertex, so an edge token has not been given one, and the object a link predictor scores, a pair of vertices, is not a vertex either. And the walk probabilities around a pair depend on whether the pair has an edge in the graph, so encodings read on the graph as it stands hand a model the presence of the very edge it is asked to predict (Appendix B.1). Encodings read correctly must be computed with the query's edge in one fixed state, on a graph that differs for every query by one edge. The remedies on record recompute per extracted subgraph or per fold of the training links (Pan et al., 2022; Dong et al., 2022; Wang et al., 2022); on a subgraph of s vertices, a depth-K random-walk encoding is $K-1$ products of its $s \times s$ walk matrix, $O(Ks^3)$ for every query. LPFormer states the conclusion the field drew: random-walk encodings "are only designed for use on the node-level and are unable to be used for link-level tasks" (Shomer et al., 2024).

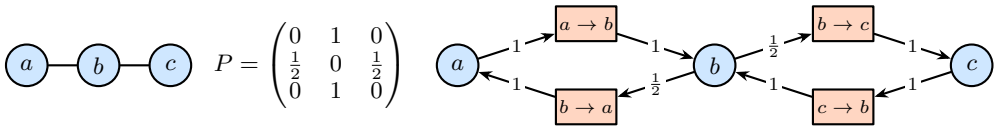
One operator for both classes. We close both gaps with the incidence graph of the base graph $G = (V, E)$, also called its Levi graph, whose vertex set is $V \sqcup E$ with an edge between a vertex and each edge it lies on. Once each edge is given its two orientations, two steps of the walk on it are one step of the walk on the base graph (Section 3.1): a vertex token carries exactly the random-walk encoding the literature already uses, an edge token its own return, a walk of the base graph between its endpoints, and every token's walk probabilities to the two endpoints of a query are its position relative to the query, SEAL's distance label (Zhang & Chen, 2018) in walk form. We call this incidence RWPE: every entry of it is an entry of a power of the base graph's walk operator, so the tables that encode the vertices encode the edges too. Links are then tokens as nodes are, the query edge among them, and the model scores that token.

The encodings hold the heuristics, and more. The common-neighbor heuristics are closed-form functions of the tokens' own rows (Section 3.3): resource allocation is a ratio of two returns of the query-edge token, and every degree-weighted common-neighbor score is a sum over the context tokens, every token but the query edge and its two endpoints, of a function of each token's row. The query edge's depth-three return separates queries on which every such score agrees.

Exact on the graph the query is scored under. Every token's encodings are evaluated on the graph in which the query edge is present, for every query, so a training edge and a held-out one are encoded identically, which closes the shift FakeEdge names (Dong et al., 2022). The query's edit of the graph is rank two, so every encoding on the edited graph is a closed-form function of the same encodings on G (Proposition 5): this is editable RWPE, whose tables are computed once per graph, so a run needs at most one walk per vertex of G and $O(K^2)$ arithmetic per token at depth K, where recomputing on each query's graph takes a walk from every token of every query (Section 4). On ogbl-ppa the encodings of a run's 68 million query pairs cost eleven minutes, against a twenty-minute epoch (Appendix E).

The contributions are the following.

- Incidence RWPE: one walk operator that gives both token classes a return encoding of their own and every token a relative encoding to the query, with the common-neighbor heuristics as closed-form functions of it that the depth-three return exceeds.
- Editable RWPE: a rank-two correction that reads every encoding on the graph the query is scored under from walks computed once on the graph, at most one per vertex, plus $O(K^2)$ arithmetic a token.
- Vephormer, for vertex–edge graph transformer: a standard transformer over these tokens with no message passing, mask, or node identifiers, level with or above every published graph transformer on the six benchmarks and above every published model on ogbl-ppa, at a small fraction of the largest models' size (Section 6, Figure 1).
- A probe of the tokens for the heuristics, and ablations that find the context worth nearly a quarter of the model's score on ogbl-ppa although the query's own tokens carry the heuristics there, and a message-passing step after every block largely redundant.



(a) G and its walk operator P (b) its directed incidence graph, with the transition probabilities of $\vec{M}$

Figure 2: The directed incidence walk on a path, each edge of G split into two arcs; two of its steps are one step of P. From b, two steps reach a with probability $P_{ba} = 1/2$; from the arc $a \to b$, four steps through b, $b \to a$, and a return to it with probability $P_{ba}/d_a = 1/2$, the edge's return at depth one (Proposition 1).

2 Background

We consider finite, simple, undirected graphs. A graph $G = (V, E)$ has n vertices and m edges, adjacency matrix A, degrees d_x, and degree matrix D. Its random-walk operator is $P = D^{-1}A$, with $D^{-1}_{xx} = 0$ at isolated vertices, and $(P^k)_{xy}$ is the probability that a k-step walk from x is at y; P is reversible with respect to the degrees, $d_x(P^k)_{xy} = d_y(P^k)_{yx}$. The random-walk positional encoding of a vertex is the sequence of its self-returns, $((P^k)_{xx})_k$ (Dwivedi et al., 2022), and the relative encoding of a pair is the sequence of its walk probabilities, $((P^k)_{xy})_k$ (Ma et al., 2023). The incidence graph $\mathcal{L}(G)$, also called the Levi graph, has vertex set $V \sqcup E$ and joins a vertex x to an edge e when $x \in e$. Its directed version replaces each edge $\{a, b\}$ by its two arcs $(a \to b)$ and $(b \to a)$, joining each arc from its tail a and to its head b, so that a walk enters an edge at one endpoint and leaves at the other (Figure 2). A random walk on either alternates between vertices and edges, so its return probabilities give an edge an encoding of the kind the random-walk encoding gives a vertex, which Section 3 develops.

3 Incidence RWPE: one walk operator for both classes

The tokens of a query are vertices and edges of G, and a transformer over them needs each token an encoding of where it stands. A vertex has one, the random-walk encoding (Dwivedi et al., 2022); an edge has none, since a walk on G is a sequence of vertices. On the incidence graph an edge is a state of the walk, with a return probability and probabilities toward the endpoints of a query as a vertex has. On the undirected incidence graph two steps return a vertex to itself with probability $1/2$, the lazy walk $(I + P)/2$ rather than P; on the directed one they are one step of P itself (Section 3.1), from which Section 3.2 reads both classes' encodings and Section 3.3 what they contain.

3.1 Two directed steps are one step of P

The undirected walk holds because an edge does not remember which endpoint the walk arrived from: from an edge it steps to either endpoint with probability $1/2$, and half the time that is the one it came from (Appendix B). The directed incidence graph removes the holding step: each edge $\{a, b\}$ becomes its two arcs $(a \to b)$ and $(b \to a)$, a step from a vertex chooses one of its outgoing arcs uniformly, and a step from an arc lands on the arc's head (Figure 2). On the state space $V \sqcup \vec{E}$ of vertices and arcs the operator is

$$\vec{M} = \begin{pmatrix} 0 & D^{-1}T \\ H^\top & 0 \end{pmatrix}, \qquad T_{a,(a\to b)} = H_{b,(a\to b)} = 1, \tag{1}$$

with T and H the $n \times 2m$ tail and head incidence matrices of the arcs and every other entry zero. Two steps from a vertex a leave along some $(a \to b)$ with probability $1/d_a$ and land on b, one step of P; two steps from an arc land on its head and leave along one of the head's arcs, the same step read at arcs. The pattern holds at every depth.

Proposition 1 (The incidence walk is the walk on G)**.** *For vertices x, y, arcs $(a \to b)$ and $(a' \to b')$, and $k \geq 0$,*

$$\begin{aligned} (\vec{M}^{2k})_{xy} &= (P^k)_{xy}, & (\vec{M}^{2k+1})_{x,(a'\to b')} &= \frac{(P^k)_{xa'}}{d_{a'}}, \\ (\vec{M}^{2k+1})_{(a\to b),y} &= (P^k)_{by}, & (\vec{M}^{2k+2})_{(a\to b),(a'\to b')} &= \frac{(P^k)_{ba'}}{d_{a'}}. \end{aligned} \tag{2}$$

In words, a walk leaves an arc from its head and enters an arc at its tail, paying $1/d$ there (proof in Appendix B). Every entry on the right is an entry of a power of P, so the incidence walk is never run: the tables of P^k that encode the vertices encode the arcs too.

Orienting loses nothing and gains three things. Removing the holding step rather than keeping the lazy walk $\tilde{P} = (I + P)/2$ loses no information in the tables, since the lazy powers are binomial mixtures of the plain ones, and gains three things. A vertex token carries exactly the encoding of Dwivedi et al. (2022), so the model without its edge tokens is the standard RWPE model and not a variant of it. An edge's return is a walk between its endpoints alone, where the undirected return adds a term in the endpoints' own returns (Appendix B). And each coordinate is a walk of one length, so every formula of Section 3.3 uses one coordinate of one token's encoding.

3.2 Every encoding is an entry of a power of P

A token's encoding has two kinds of coordinate, both read from its row of Equation (2) over a window of depths k: its return probability $r_k(\tau)$, the probability that the walk is back at the token τ, and its relative walk probability $\rho_k(\tau, t)$ to each endpoint t of the query, the probability that the walk is at t after k steps of P. We call them the token's return and its relative encoding for short; the returns, the literature's RWPE at a vertex and its counterpart at an edge, are incidence RWPE. An edge's two arcs have the same return, and their relative encodings are averaged. Throughout, (u, v) is the query, $t \in \{u, v\}$ one of its endpoints and $\bar{t}$ the other, and $[\cdot]$ the indicator of a condition.

Corollary 2 (Both classes' encodings)**.** *(i) A vertex x has*

$$r_k(x) = (\vec{M}^{2k})_{xx} = (P^k)_{xx}, \qquad \rho_k(x, t) = (\vec{M}^{2k})_{xt} = (P^k)_{xt},$$

the return being standard RWPE.

(ii) An edge $e = \{a, b\}$ has

$$r_k(e) = (\vec{M}^{2k+2})_{(a\to b),(a\to b)} = \frac{(P^k)_{ba}}{d_a} = \frac{(P^k)_{ab}}{d_b},$$
$$\rho_k(e, t) = \tfrac{1}{2}\big((\vec{M}^{2k+1})_{(a\to b),t} + (\vec{M}^{2k+1})_{(b\to a),t}\big) = \tfrac{1}{2}\big((P^k)_{at} + (P^k)_{bt}\big),$$

the return being the same in either orientation, since $d_a(P^k)_{ab} = d_b(P^k)_{ba}$.

An edge's return is a walk of G between its endpoints divided by a degree: at $k = 1$ it is $1/(d_a d_b)$, at $k = 2$ it is $\mathrm{RA}(a, b)/(d_a d_b)$, and at $k = 3$ it also weighs the three-step paths between the endpoints. The relative walk probabilities extend distance encoding's landing probabilities (Li et al., 2020) to both classes; the first depth at which one is nonzero is the distance in G to that endpoint, from an edge's nearer end. A token's input is its returns over one window of depths and its two relative sequences over another (Appendix D). The two sequences are ordered lexicographically, the larger first, so the input is invariant to swapping u and v (Appendix A).

3.3 The encodings hold the heuristics, and more

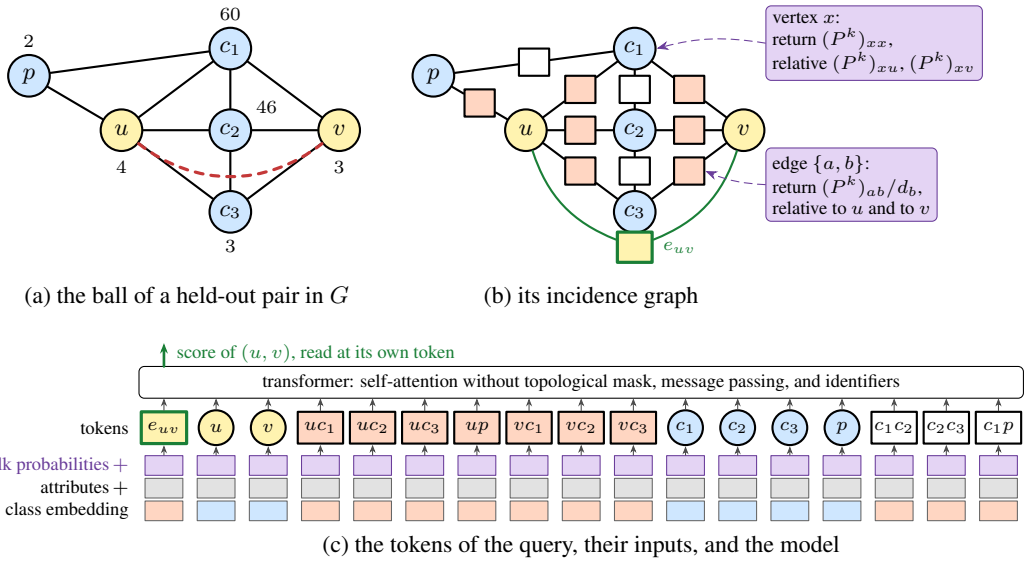
Every probability below is on G^+, the graph with the query edge present (Section 4), with degrees d^+ and operator P^+; a common neighbor's degree is unchanged by the edge. Let $N_{uv} = N(u) \cap N(v)$ and $h_f(u, v) = \sum_{c \in N_{uv}} f(d_c)$, with $f(d) = 1$ the common-neighbor count (CN), $1/\log d$ Adamic–Adar (AA), and $1/d$ resource allocation (RA), the three heuristics as Appendix A defines them. Proposition 3 divides them between the two kinds of token: RA is read from the query-edge token alone, and every h_f, CN and AA among them, is a sum over the context tokens.

Proposition 3 (The heuristics are functions of the encodings)**.** *Write $r_k(e_{uv}) = ((P^+)^k)_{uv}/d^+_v$ for the query edge's return.*

(i) The query-edge token itself gives resource allocation, with $\mathrm{RA}(u, v) = r_2(e_{uv})/r_1(e_{uv})$.

(ii) Suppose the token set keeps either (a) the vertex token of every common neighbor c of u and v, as extents 6 and 8 do, or (b) both edges $\{u, c\}$ and $\{c, v\}$ of every common neighbor, as every extent from 4 does. Then for every weight f,

$$h_f(u, v) = \sum_{\tau \text{ context}} \phi_f(\tau),$$



(a) the ball of a held-out pair in G (b) its incidence graph

(c) the tokens of the query, their inputs, and the model

Figure 3: A held-out test pair of ogbl-ppa and the model over it. (a) Its one-hop ball, each vertex marked with its degree in the whole graph; the dashed query edge is absent from the training graph. (b) The incidence graph of the ball: each edge becomes a box joined to its two endpoints, and every vertex and box carries walk probabilities (Corollary 2) with the query edge present. (c) Each token sums projections of its walk probabilities and its attributes, an edge's lifted from its endpoints, and a class embedding; attention is unmasked.

where ϕ_f is a function of a token's class and its own encodings alone, the same function for every graph and every query. Under (a) it reads a vertex token's depth-one relative encoding; under (b) an edge token's depth-zero and depth-one relative encodings and its depth-one return.

The proof is in Appendix A. An unmasked transformer of the form in Section 5.2 realizes every h_f through uniform attention, with no identifiers, mask, or supplied token count (Theorem 6). The walk identities are classical (Appendix A.1); what is particular here is that each term is a function of one token's own row, so it survives whatever cut retains the common neighbors.

Proposition 4 (Depth three exceeds the family)**.** *Two queries with equal endpoint degrees and equal h_f for every f can differ in $r_3(e_{uv})$.*

On an eight-cycle with a vertex joined to 0 and to j, the queries $(0, 3)$ and $(0, 4)$ agree on every h_f and have $r_3(e_{uv})$ of $17/256$ and $13/256$ (Appendix A.5), so the inventory is not the heuristics under another name. What this depth-three surplus is worth to prediction is not measured here: no arm varies the depth window; Section 6.4 measures a different surplus, what the context tokens add beyond the heuristics the query tokens hold.

4 Editable RWPE: exact on the graph the query is scored under

On the training graph $P_{uv} = [(u, v) \in E]/d_u$ is nonzero exactly for the pairs that are edges, and the deeper walk probabilities change with the edge's state too, so a model reads the edge off them: an arm whose context rows were read on G trained to a loss an order of magnitude below the same arm's without those rows while its validation score fell by three quarters (Appendix B.1). We therefore evaluate every query on $G^+ = G^- + (u, v)$, its edge forced present whether or not the graph has it, WalkPool's fixed state for a subgraph (Pan et al., 2022) taken for the whole graph, and carry the tables of G onto it by a rank-two edit.

Proposition 5 (Rank-two correction)**.** *Let $(u, v) \notin E$, so that $G = G^-$ and $G^+ = G + (u, v)$, and let $\Delta = P^+ - P$. Then Δ vanishes outside the rows u and v, where, with $\mathbf{1}_{\bar{t}}$ the indicator vector of*

the other endpoint, $\Delta_{t\cdot} = (\mathbf{1}_{\bar{t}}^\top - P_{t\cdot})/(d_t + 1)$*, and for all* $x, y \in V$ *and* $k \geq 1$

$$\left((P^+)^k\right)_{xy} = (P^k)_{xy} + \sum_{j=0}^{k-1} \sum_{t \in \{u,v\}} \left((P^+)^{k-1-j}\right)_{xt} (\Delta P^j)_{ty}, \quad (\Delta P^j)_{ty} = \frac{(P^j)_{\bar{t}y} - (P^{j+1})_{ty}}{d_t + 1}. \tag{3}$$

Hence the corrected entry at depth k *is a function of* $(P^k)_{xy}$ *and the uncorrected entries* $(P^j)_{xt}$*,* $(P^j)_{ty}$*, and* $(P^j)_{st}$ *for* $s, t \in \{u, v\}$ *and* $j \leq k$ *and of* d_u, d_v *alone, and all corrected entries of a pair through depth* K *cost* $O(K^2)$ *operations given them.*

The proof is the discrete Duhamel formula, the telescoping identity for matrix powers (Appendix B). This is editable RWPE. Over a run of Q queries of s tokens each, on a graph of n vertices and m edges, recomputing the encodings on each query's G^+ takes a walk of length K from every token of every query, Qs walks, since a token's return is its own walk's and no two queries share a graph; editing reads every entry it needs from walks on G itself, at most one from each vertex, n walks that give the tables and the endpoints' rows alike, plus $O(K^2)$ arithmetic a token. A walk costing $O(Km)$ sparse and $O(Kn^2)$ dense, the run costs

over Q queries	sparse	dense
recomputed on each G^+	$O(QsKm)$	$O(QsKn^2)$
edited from G (ours)	$O(nKm + QsK^2)$	$O(Kn^3 + QsK^2)$

Editing trades each token's walk, $O(Km)$ sparse, for $O(K^2)$ arithmetic, and the Qs walks for at most n, so it divides the sparse cost by about $\min(Qs/n,\, m/K)$ and the dense by $\min(Qs/n,\, n^2/K)$; usually $K \ll s \ll n \ll Q$ and $K \ll m$. On ogbl-ppa, with $K = 3$, $m = 21$ million edges, $n = 576{,}289$, $Q = 68$ million pairs, and a median s of about 166 for a training edge and 108 for a random pair, the smaller factor, Qs/n, is over ten thousand; the run's encodings took eleven minutes (Appendix E).

5 VEPHORMER: THE TOKENS AND THE MODEL

5.1 THE QUERY'S SUBGRAPH AND ITS TOKENS

The tokens of a query are the vertices and edges of the incidence graph around the query edge, and the *extent* says how far around: the tokens are those within distance r of the query edge, itself first, on the incidence graph of the query's one-hop ball, the subgraph induced by u, v, and their neighbors, and the extent is $2r$, the length of a walk from the query edge to the farthest token and back (Appendix B.2, Equation (13)). The *query tokens* are u, v, and the query edge, whose own token is the *query-edge token*; every other token is a *context token*. Extent 2 keeps the query tokens alone, extent 4 adds the edges incident to u or v, extent 6 the neighbors themselves, and extent 8 the edges among them, so that every vertex and edge of the ball is a token (Figure 3). Where a budget binds, the neighbors are ranked by their walk mass, the mass of the short closed walks through the query edge that pass through them, which favors common neighbors, low-degree neighbors, and neighbors joining the two sides of the ball; evaluation keeps the top and training samples by it (Appendix B.2, Algorithm 1). The ball is read on G with the other endpoint excluded from each neighborhood and every encoding on G^+, so a training positive and the same pair held out are one input (Proposition 7).

5.2 A STANDARD TRANSFORMER, WITH STRUCTURE ONLY AT ITS INPUT

Each token's input is the sum of a class embedding, one for vertices and one for edges, and a linear projection of its class's inventory, one projection per class shared by the query tokens and the context tokens. Where a dataset's vertices carry attributes, a vertex token adds a projection of its row, layer-normalized then linear, and an edge token adds a projection of the symmetric terms of its two endpoints' attribute embeddings, their sum, product, and absolute difference, so an edge reads its endpoints' attributes in a form that does not depend on their order. The trunk is a stack of pre-norm transformer blocks with RMS normalization and a gated feed-forward layer, attention

Table 1: Test results in percent, mean and standard deviation over seeds; best per column in bold, second best underlined; the rows grouped by whether a method reads pairwise heuristics, a learned embedding per vertex identifier, or neither of the two.

	Planetoid (HeaRT protocol)			OGB (original protocol)		
method	cora MRR	citeseer MRR	pubmed MRR	ogbl-collab Hits@50	ogbl-ppa Hits@100	ogbl-citation2 MRR
heuristic-informed						
SEAL	10.67 ± 3.46	13.16 ± 1.66	5.88 ± 0.53	63.37 ± 0.69	48.80 ± 5.61	86.93 ± 0.43
BUDDY	13.71 ± 0.59	22.84 ± 0.36	7.56 ± 0.18	65.94 ± 0.58	49.85 ± 0.20	87.56 ± 0.11
NCN	14.66 ± 0.95	**28.65** ± 1.21	5.84 ± 0.22	64.76 ± 0.87	61.19 ± 0.85	88.64 ± 0.14
NCNC	14.98 ± 1.00	24.10 ± 0.65	8.58 ± 0.59	66.61 ± 0.71	61.42 ± 0.73	89.12 ± 0.40
LPFormer	16.80 ± 0.52	26.34 ± 0.67	9.99 ± 0.52	68.14 ± 0.51	63.32 ± 0.63	89.81 ± 0.13
Link-MoE				**71.32** ± 0.99	69.39 ± 0.61	91.25 ± 0.02
identifier-based						
MPLP+				66.99 ± 0.40	65.24 ± 1.50	90.72 ± 0.12
Optimized GAE				66.11 ± 0.35	78.41 ± 0.83	88.74 ± 0.06
GraphGPT					76.55 ± 0.67	**93.05** ± 0.20
heuristic-agnostic, identifier-free						
GCN	16.61 ± 0.30	21.09 ± 0.88	7.13 ± 0.27	54.96 ± 3.18	29.57 ± 2.90	84.85 ± 0.07
SAGE	14.74 ± 0.69	21.09 ± 1.15	9.40 ± 0.70	59.44 ± 1.37	41.02 ± 1.94	83.06 ± 0.09
GAE	**18.32** ± 0.41	25.25 ± 0.82	5.27 ± 0.25			
PENCIL	14.63 ± 0.52	16.80 ± 1.43	8.88 ± 0.49	66.88 ± 0.34	79.54 ± 0.07	86.86 ± 0.20
Vephormer (ours)	17.20 ± 0.15	26.33 ± 0.53	**11.90** ± 0.23	69.72 ± 0.34	**84.57** ± 0.23	92.98 ± 0.05

unmasked within each query's segment on the variable-length kernel. No block carries a message-passing step, a mask, or an attention bias; every structural quantity enters at the input. The score is a linear head on the normalized representation of the query-edge token, so the link is scored as the token it is. Training minimizes binary cross-entropy over the training positives and as many uniformly drawn non-edges (Appendix D).

6 Experiments

6.1 Setup

We evaluate on the three Planetoid citation graphs (Yang et al., 2016), cora, citeseer, and pubmed, under the split of HeaRT (Li et al., 2023), and on ogbl-collab, ogbl-ppa, and ogbl-citation2 under the official OGB splits (Hu et al., 2020) (Appendix C). The extent is 8 on the Planetoid graphs, where the ball is uncut, and 4 on the OGB graphs, where a budget cuts it by walk mass (Section 5.1). The model is that of Section 5, and Appendix D gives its modeling and training hyperparameters. Every result in the main text on the Planetoid graphs is under HeaRT's per-positive hard negatives (Li et al., 2023), and every result on the OGB graphs under their original protocol, the evaluator and negatives OGB releases; each protocol selects its own checkpoint on validation and scores the whole pool (Table 4), by mean reciprocal rank (MRR) on the Planetoid graphs and by each OGB graph's metric, Hits@50 on ogbl-collab, Hits@100 on ogbl-ppa, and MRR on ogbl-citation2.

Table 2: Held-out R^2 of MLP probes reading each heuristic from the query tokens' walk probabilities, and CN also from the context tokens averaged beside them, over the test queries with a common neighbor (Appendix F).

	query tokens			+context
graph	CN	AA	RA	CN
cora	0.777	0.990	0.999	0.923
citeseer	0.799	0.974	0.998	0.965
pubmed	0.710	0.932	1.000	0.989
ppa	0.968	0.982	0.999	0.980

6.2 The probe recovers RA from the query tokens and CN with the context

Proposition 3 places RA in the query-edge token alone and every degree-weighted score in the context tokens; small MLP probes, following PENCIL's heuristic regression (Truong et al., 2026), ask whether that placement holds in the standardized walk inputs the model reads, from the query tokens alone or with the context tokens pooled beside them (Table 2; Appendix F). It does. RA is

read from the query tokens almost exactly on every graph. CN is not, on the Planetoid graphs, since the query edge's depth-two return weights each common neighbor by its inverse degree, which only that neighbor's own token carries, and the context closes most of the gap. AA from the query tokens, and CN on ogbl-ppa, are recovered beyond the placement, though neither is a function of the query tokens alone: on these graphs they correlate with what the tokens do hold, RA above all, and a high R^2 cannot tell such a correlation from an exact function of the encodings.

6.3 Comparison with published methods

Table 1 reports results in percent; prose keeps each metric's own scale. Each published method shows its better variant with or without attributes, read from its source's own table (Appendix G). Among the methods that read neither heuristics nor vertex identifiers, the last group of the table, Vephormer leads every column but cora, where GAE is ahead, and is above PENCIL on all six.

The lead is widest where the structure around the pair decides. On ogbl-ppa Vephormer is above every published row, the identifier-based ones included, whose embeddings per vertex run to hundreds of millions of parameters against its 4.4 million (Figure 1). On pubmed it leads LPFormer by 0.019, and the lead is the context's: without its context tokens the same model reads 0.0998 (Table 3), level with LPFormer's 0.0999. Where it is not first, it is second to GAE on cora and level with LPFormer on citeseer; on ogbl-citation2 it is within the spread of GraphGPT, a transformer pretrained on the graph's own subgraphs by masked-token prediction before fine-tuning (Zhao et al., 2025), at over twenty times its size, and on ogbl-collab below only Link-MoE, an ensemble of eight predictors whose gate is fitted on validation pairs (Ma et al., 2024). The attribute lift and the seed counts behind every difference are in Appendix D.1.

6.4 Ablations

Table 3 removes one input of the full model at a time, read against the seed spread; its first row drops the vertex attributes, on ogbl-collab the dated rows, as PENCIL's "w/o Features" (Truong et al., 2026).

The encodings carry the structure. Without the walk encodings every score falls, least on cora and citeseer and most on ogbl-ppa, which halves. The attributes alone outscore the encodings alone on the Planetoid graphs, and on ogbl-ppa the encodings alone are above every published row.

The context is worth most on ogbl-ppa. Without the context tokens ogbl-ppa loses nearly a quarter of the model's score, pubmed and ogbl-collab less, and cora and citeseer no more than the spread. The model without context still has the query tokens, and on ogbl-ppa they carry the heuristics (Table 2) and the depth-three return of Proposition 4, so that quarter is likely what the context adds beyond both. That is an inference: the probe shows the heuristics are readable from the query tokens, not that this model reads them.

A message-passing step adds a marginal gain. A learned projection of each token's average with its neighbors over the incidence graph, after every block, moves no score by more than the seed spread, where removing PENCIL's message-passing branch costs it 0.18 to 0.44 of its score (Truong et al., 2026, Table 9); with exact walk probabilities at the input, the adjacency product appears largely redundant, if marginally beneficial.

7 Related work

Link predictors, by what they describe. The heuristics score a pair by its common neighbors, weighted uniformly, by inverse log degree, or by inverse degree, and their walk forms are classical (Zhou et al., 2009; Liu & Lü, 2010). The strongest neural predictors build on those scores, NCN by aggregating the common neighbors' representations (Wang et al., 2024), BUDDY by sketching the endpoints' neighborhoods (Chamberlain et al., 2023), MPLP by propagating quasi-orthogonal vectors (Dong et al., 2024), HL-GNN by products of normalized adjacency matrices (Zhang et al., 2024), and PENCIL by random token vectors propagated through its message-passing branch (Truong et al., 2026, Proposition 4.5); here they are closed-form functions of each token's

Table 3: Ablations with seed standard deviation as a subscript

	HeaRT			original	
	cora MRR	citeseer MRR	pubmed MRR	collab Hits@50	ppa Hits@100
structure only	$14.12_{0.37}$	$15.30_{0.80}$	$7.50_{0.25}$	$67.96_{0.20}$	$82.83_{0.39}$
attributed					
no walk encoding	$15.25_{0.86}$	$25.36_{0.48}$	$8.39_{0.30}$	$52.07_{0.10}$	$41.26_{0.48}$
no context tokens	$16.98_{0.26}$	$25.27_{1.07}$	$9.98_{0.13}$	$66.52_{0.22}$	$65.12_{0.91}$
Vephormer (full)	$17.20_{0.15}$	$26.33_{0.53}$	$11.90_{0.23}$	$69.72_{0.34}$	$84.57_{0.23}$
+ Message-passing	$17.19_{0.97}$	$26.35_{1.00}$	$12.21_{0.69}$	$70.00_{0.28}$	$84.74_{0.08}$

own row (Proposition 3). The subgraph methods describe the pair by an extracted neighborhood: SEAL labels each vertex by its distances to the two endpoints (Zhang & Chen, 2018), distance encoding generalizes the label to landing probabilities of walks from the target set (Li et al., 2020), and WalkPool computes walk profiles on each enclosing subgraph with the target link forced present and forced absent (Pan et al., 2022). LGLP made the link a vertex, of the line graph of SEAL's subgraph, with a distance label and no encoding of its own (Cai et al., 2022). Of the transformers, LPFormer attends over vertices with a pairwise encoding built from personalized PageRank and the heuristic counts (Shomer et al., 2024), and PENCIL attends over adjacency rows with no encoding, arguing from cost (Truong et al., 2026); neither has a token for the link. GraphGPT serializes sampled subgraphs along Eulerian paths, spelling each vertex's identity in tokens, and pretrains on the graph itself by masked-token prediction (Zhao et al., 2025); it locates a vertex by its identity, not by an encoding of the structure around it.

Tokens and encodings on graph transformers. TokenGT made every vertex and every edge a token and recovered the connectivity through identifiers (Kim et al., 2022), and ESA tokenizes edges through the line graph (Buterez et al., 2025); in both an edge token's position is learned, not read from a walk. ESA's structure enters as a topological attention mask, as GAT's (Veličković et al., 2018) and Exphormer's (Shirzad et al., 2023) do; Vephormer carries no mask, since every token's position is in its input. Random-walk encodings give a vertex its self-returns (Dwivedi et al., 2022; Rampášek et al., 2022), and GRIT gives a pair of vertices the off-diagonal entries as relative walk probabilities (Ma et al., 2023). Proposition 1 joins the two lines: the operator that returns a vertex's encoding returns an edge's and every token's position relative to the query, so the edge class needs neither an identifier nor a line graph.

The query edge, and the protocols. FakeEdge names the shift between a training edge, present in the graph, and a held-out one, absent, and closes it by editing each extracted subgraph (Dong et al., 2022); WalkPool's forced-present subgraph is the fixed state used here, per query and on the subgraph's own walks, where Proposition 5 carries the whole graph's walks onto it in closed form. PEG, the one positional-encoding paper to have faced it, recomputes its encodings with each of ten folds of the training links removed (Wang et al., 2022); Propositions 5 and 7 close the shift for whole-graph encodings at $O(K^2)$ a token. On evaluation, HeaRT showed that a shared random pool and a per-positive hard pool rank the same methods differently and released the split and negatives used here (Li et al., 2023).

8 Conclusion

Vephormer is a graph transformer before it is a link predictor: its tokens are a graph's vertices and edges, all of its structure is the incidence walk's encodings at the input, and link prediction only chooses which token to score. The same model can take tasks on vertices, edges, or whole graphs, including graphs whose edges carry features, such as molecules, and the rank-two edit serves any input a few edges away from a tabulated graph, from a link to be predicted to an edge arriving in a temporal graph. That on ogbl-ppa it outscores the models built around the heuristics suggests that much of what graph models engineer by hand can be left to the walk encodings.

AI USE STATEMENT

We used generative AI tools, Claude and Codex, in the following ways, under the authors' direction and review. Of the tasks with required disclosure: the tools implemented methods, writing much of the code base, its tests, and its experiment tooling to the authors' design; they assisted in formulating the theoretical statements and their proofs; they gave feedback on the design of the experiments; and they assisted in refining hypotheses and in interpreting the results. They were not used to generate synthetic data, to clean or reformat a dataset, for translation, or for qualitative analysis. Of the tasks with recommended disclosure: they drafted and edited parts of the text, produced the figures and the tables from the run records, wrote and edited software, and searched and summarized the literature. The authors are responsible for checking every proof, testing the generated code, verifying published numbers against their sources, and reviewing all AI-assisted work before submission. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The heuristic identities and their proofs are in Section 3.3 and Appendix A, with an exact-arithmetic finite verification script. Every encoding is defined by Propositions 1 and 5 and computed by Algorithm 1; the model and its objective are Section 5.2, and the protocols and their checkpoint selection Section 6. Appendix D gives the modeling and training hyperparameters and the hardware; Appendix F the probe; Appendix G the source of every published row. The anonymized code is uploaded as supplementary material, and the full code will be released in a public repository upon publication; the datasets are the public Planetoid and OGB releases, under the splits of Li et al. (2023) and of OGB.

REFERENCES

Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3): 211–230, 2003.

Mitchell Black, Zhengchao Wan, Gal Mishne, Amir Nayyeri, and Yusu Wang. Comparing graph transformers via positional encodings. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

David Buterez, Jon Paul Janet, Dino Oglic, and Pietro Liò. An end-to-end attention-based approach for learning on graphs. *Nature Communications*, 16:5244, 2025. doi: 10.1038/s41467-025-60252-z. URL https://doi.org/10.1038/s41467-025-60252-z.

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5103–5113, 2022. doi: 10.1109/TPAMI.2021.3080635. URL https://arxiv.org/abs/2010.10046.

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=m1oqEOAozQU.

Kaiwen Dong, Yijun Tian, Zhichun Guo, Yang Yang, and Nitesh Chawla. FakeEdge: Alleviate dataset shift in link prediction. In Bastian Rieck and Razvan Pascanu (eds.), *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, pp. 56:1–56:19. PMLR, 2022. URL https://proceedings.mlr.press/v198/dong22a.html.

Kaiwen Dong, Zhichun Guo, and Nitesh V. Chawla. Pure message passing can estimate common neighbor for link prediction. In *Advances in Neural Information Processing Systems*, volume 37, pp. 73000–73035, 2024. doi: 10.52202/079017-2324. URL https://arxiv.org/abs/2309.00976.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International*

Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=wTTjnvGphYj.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020. URL https://arxiv.org/abs/2005.00687.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2207.02505.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. doi: 10.1016/S0893-6080(05)80131-5. URL https://doi.org/10.1016/S0893-6080(05)80131-5.

Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 36, pp. 3853–3866, 2023. doi: 10.52202/075280-0169. URL https://arxiv.org/abs/2306.10453v3.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://arxiv.org/abs/2009.00142v4.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007. doi: 10.1002/asi.20591.

Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, 2010. doi: 10.1209/0295-5075/89/58007. URL https://arxiv.org/abs/1001.2467.

Li Ma, Haoyu Han, Juanhui Li, Harry Shomer, Hui Liu, Xiaofeng Gao, and Jiliang Tang. Mixture of link predictors on graphs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 16043–16070, 2024. doi: 10.52202/079017-0511. URL https://arxiv.org/abs/2402.08583v2.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23321–23337. PMLR, 2023. URL https://proceedings.mlr.press/v202/ma23c.html.

Weishuo Ma, Yanbo Wang, Xiyuan Wang, and Muhan Zhang. Reconsidering the performance of GAE in link prediction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 2025. doi: 10.1145/3746252.3761108. URL https://doi.org/10.1145/3746252.3761108.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *Transactions on Machine Learning Research*, 2024.

Liming Pan, Cheng Shi, and Ivan Dokmanić. Neural link prediction with walk pooling. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.04375.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14501–14515, 2022. URL https://arxiv.org/abs/2205.12454.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Harry Shomer, Yao Ma, Haitao Mao, Juanhui Li, Bo Wu, and Jiliang Tang. LPFormer: An adaptive graph transformer for link prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2686–2698, 2024. doi: 10.1145/3637528.3672025. URL https://arxiv.org/abs/2310.11009v4.

Quang Truong, Yu Song, Donald Loveland, Mingxuan Ju, Tong Zhao, Neil Shah, and Jiliang Tang. Plain transformers are surprisingly powerful link predictors. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. URL https://arxiv.org/abs/2602.01553.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2203.00199.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sNFLN3itAd.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 40–48, 2016. URL https://arxiv.org/abs/1603.08861.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByxRM0Ntvr.

Juzheng Zhang, Lanning Wei, Zhen Xu, and Quanming Yao. Heuristic learning with graph neural networks: A unified framework for link prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4223–4231, 2024. doi: 10.1145/3637528.3671946. URL https://arxiv.org/abs/2406.07979v2.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1802.09691.

Qifang Zhao, Weidong Ren, Tianyu Li, Hong Liu, Xingsheng He, and Xiaoxiao Xu. GraphGPT: Generative pre-trained graph Eulerian transformer. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77630–77656. PMLR, 2025. URL https://arxiv.org/abs/2401.00529v3.

Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009. doi: 10.1140/epjb/e2009-00335-8. URL https://arxiv.org/abs/0901.0553.

A Heuristic identities and transformer realization

The heuristics. For a pair (u, v) with common neighbors $N_{uv} = N(u) \cap N(v)$, the common-neighbor count is $\mathrm{CN}(u, v) = |N_{uv}|$, Adamic–Adar is $\mathrm{AA}(u, v) = \sum_{c \in N_{uv}} 1/\log d_c$ (Adamic & Adar, 2003), and resource allocation is $\mathrm{RA}(u, v) = \sum_{c \in N_{uv}} 1/d_c$ (Zhou et al., 2009); each is

$h_f(u,v) = \sum_{c \in N_{uv}} f(d_c)$ for $f(d) = 1$, $1/\log d$, and $1/d$, the degree-weighted family of Section 3.3. Each scores a pair by the paths of length two between its endpoints, weighted by the degree of the vertex in the middle, and each is nonzero exactly when the pair has a common neighbor.

All identities in this appendix use the simple undirected graph $G^+ = G^- + (u,v)$ and its walk operator P^+. A context vertex's degree is unchanged by the insertion; an endpoint's degree increases by one. The feature coordinates are the raw walk probabilities before the fixed injective preprocessing, which is this: every coordinate but the vertex return is passed through $\rho \mapsto \log(1 + n\rho)$, which reads ρ in multiples of the uniform mass $1/n$; the vertex return enters raw, as the literature's RWPE does; and every coordinate is then standardized by the training split's mean and standard deviation. For fixed graph size, both maps are invertible wherever the standardization's scale is nonzero. Lexicographic ordering exchanges whole endpoint sequences and therefore preserves the alignment of depths within each sequence. The formulas below respect that exchange.

Proof of Proposition 3. A context vertex c has $\rho_1(c,u) = [c \sim u]/d_c$ and $\rho_1(c,v) = [c \sim v]/d_c$, both nonzero exactly when c is a common neighbor, and their reciprocal is its degree, which gives (ii)(a). Inserting the query edge changes neither the common neighbors nor their degrees, so the classical identity $\mathrm{RA}(u,v) = d_u^+((P^+)^2)_{uv}$ (Liu & Lü, 2010, Eq. 8) with $r_1(e_{uv}) = 1/(d_u^+ d_v^+)$ gives the ratio. The incident-edge formula (ii)(b), which uses the edge's own return to recover its endpoint's degree, is below. □

A.1 Relation to earlier heuristic formulations

Classical identities. The connection between walk probabilities and local heuristics predates their use as positional encodings. Zhou et al. (2009, Sections 5–6) introduce the resource-allocation score and use the local-path score $A^2 + \epsilon A^3$ to distinguish pairs tied by common-neighbor count. Liu & Lü (2010, Eq. (8)) explicitly identify RA with a degree-scaled two-step random walk. For our simple undirected graph, the same algebra gives

$$\mathrm{CN}(u,v) = (A^2)_{uv}, \qquad \mathrm{RA}(u,v) = (AD^{-1}A)_{uv} = d_u(P^2)_{uv}. \tag{4}$$

More generally, extending f by zero at degrees zero and one, $h_f(u,v) = (A\,\mathrm{diag}(f(d_x))A)_{uv}$ for distinct endpoints. These matrix identities are background to Proposition 3. On G^+, inserting the query leaves the common neighbors and their degrees unchanged, so the RA identity becomes the ratio of the query-edge token's two return probabilities. The proposition also supplies vertex and incident-edge formulas under the actual coverage and endpoint-ordering rules.

Earlier learned formulations. WalkPool learns transition probabilities and reads their powers as node-, link-, and graph-level features (Pan et al., 2022, Section 3.2). HL-GNN gives a formulation through products of differently normalized adjacency matrices, including CN and RA as special cases (Zhang et al., 2024, Table 2 and Proposition 3.1). MPLP derives unbiased CN and higher-order connectivity estimators from random node signatures, and uses degree-dependent norm rescaling for weighted counts (Dong et al., 2024, Theorems 3.2–3.3 and Section 4.1). PENCIL proves a corresponding estimation result for its adjacency-row input and evaluates learned heuristic regression (Truong et al., 2026, Section 3.3 and Proposition 4.5). Thus neither neural heuristic recovery nor the use of walks to generalize heuristics is a priority claim here. Our analysis establishes sufficient token coverage, closed-form functions of the exact corrected incidence inventory, and their realization by a variable-length unmasked transformer without supplied node identifiers or token counts.

Which random-walk representation? Vertex self-returns alone do not imply these guarantees. Every vertex of an eight-cycle has the same self-return sequence at every depth, yet pairs at distance two have one common neighbor and pairs at distance three have none. Our input includes relative walk probabilities and edge returns. Distance Encoding and GRIT already establish uses and expressivity of relative walk probabilities (Li et al., 2020; Ma et al., 2023); Theorem 6 addresses computing the heuristics from our incidence tokens, rather than claiming heuristic recovery from arbitrary encodings called RWPE. Likewise, Proposition 4 gives a degree-matched separation from the whole family h_f on the forced-present graph, specializing the established distinction between two- and three-step structure.

A.2 THE INCIDENT-EDGE FORMULA

Consider an incident edge $e = \{t, c\}$, where $t \in \{u, v\}$, $c \notin \{u, v\}$, and $\bar{t}$ is the other query endpoint. Its depth-zero relative walk probabilities are $\rho_0(e, t) = 1/2$ and $\rho_0(e, \bar{t}) = 0$, so its column toward t sorts first. This identifies the incident column locally, without matching an endpoint's identity across tokens. Write

$$a_e = 2\rho_1(e, t) = \frac{1}{d_c}, \qquad b_e = 2\rho_1(e, \bar{t}) = \frac{1}{d_t^+} + \frac{[c \sim \bar{t}]}{d_c}, \qquad r_1(e) = \frac{(P^+)_{tc}}{d_c} = \frac{1}{d_t^+ d_c}. \quad (5)$$

The return $r_1(e)$ belongs to this edge token, not to the query-edge token. Consequently

$$\chi_e = \frac{b_e - r_1(e)/a_e}{a_e} = [c \sim \bar{t}], \qquad \phi_f(e) = \frac{1}{2}\chi_e f(1/a_e). \quad (6)$$

When $\chi_e = 0$ the contribution is defined to be zero without evaluating f at degree one. Every common neighbor supplies two such edges, each contributing $f(d_c)/2$; a private neighbor supplies one edge with zero contribution. Set the contribution of all other token types to zero. This proves Proposition 3(ii)(b) from each incident edge's own inventory, even at the smallest context extent and after the endpoint sequences have been sorted. The edge return is essential to this particular local decoder: it supplies the incident endpoint's reciprocal degree without an ambiguous cross-token alignment.

For case (a), use only context vertex tokens: their relative walk probabilities are both zero at depth zero, and both positive at depth one exactly for common neighbors. The formula in the main text counts each once. For either construction, deleting all tokens associated with a context vertex deletes its contribution; retaining every common neighbor, even if private neighbors are omitted, suffices for exact recovery.

A.3 DEGREE-NORMALIZED SCORES

An endpoint token t has relative walk probabilities $(1, 0)$ at depth zero after ordering and $(0, 1/d_t^+)$ at depth one in the same columns, since $\rho_0(t, t) = 1$, $\rho_1(t, t) = 0$, $\rho_0(t, \bar{t}) = 0$ and $\rho_1(t, \bar{t}) = 1/d_t^+$. Thus the pair of endpoint tokens recovers the unordered degrees $d_u^- = d_u^+ - 1$ and $d_v^- = d_v^+ - 1$ on G^-. With $\mathrm{CN} = \mathrm{CN}(u, v)$, the following scores are fixed functions of the endpoint tokens' encodings:

$$\text{Jaccard} = \frac{\mathrm{CN}}{d_u^- + d_v^- - \mathrm{CN}}, \quad \text{Salton} = \frac{\mathrm{CN}}{\sqrt{d_u^- d_v^-}}, \quad \text{Sørensen} = \frac{2\,\mathrm{CN}}{d_u^- + d_v^-}, \quad \mathrm{PA} = d_u^- d_v^-, \quad (7)$$

with zero assigned when a denominator vanishes. On a fixed finite graph domain these conventions have continuous extensions from the admissible inputs, just as the common-neighbor indicators do. The transformer can collect endpoint-degree functions along with the heuristic contributions; for example, their sum and product are determined by the sums of the degrees and of their squares. This extends Theorem 6 to the displayed scores.

A.4 REALIZATION BY UNMASKED ATTENTION

Theorem 6 (Transformer realization)**.** *Fix n, the invertible preprocessing of the walk probabilities, and a finite set $\mathcal{F}$ of weight functions. On simple undirected n-vertex graphs, retain the query edge, its endpoints, and either token coverage in Proposition 3. For every $\epsilon > 0$, a sufficiently wide unmasked transformer of the form in Section 5.2, in real arithmetic, with $|\mathcal{F}|$ output coordinates at the query-edge token approximates every h_f, $f \in \mathcal{F}$, uniformly to error at most ϵ; a scalar head selects any one score. No node identifiers, attention bias, or supplied token count are required. RA alone requires no context tokens.*

The construction is short. A tokenwise network computes its heuristic contribution ϕ_f and the query indicator q; the query edge is identifiable from its class and depth-zero relative walk probabilities $(1/2, 1/2)$, unique among the edge tokens. With the attention logits set to zero, uniform attention over the ℓ tokens reads

$$(\overline{\phi}_f, \overline{q}) = \left(\frac{1}{\ell}\sum_{\tau} \phi_f(\tau), \frac{1}{\ell}\right), \qquad h_f = \frac{\overline{\phi}_f}{\overline{q}}, \quad (8)$$

and a subsequent tokenwise network recovers the sum. The theorem guarantees representability, not successful optimization at the experimental width or in finite precision; with a truncated context the sums recover only the retained contributions, whereas RA and the depth-three separation survive in the query-edge token.

Proof. The theorem quantifies over every simple undirected graph on a fixed number n of vertices and every allowed query and retained token set satisfying the coverage assumption. The preprocessing is fixed across this domain; it is not refitted separately for each graph. Ignore attributes by setting their projections to zero. In a sufficiently wide input space, the class embeddings and the two class-specific projections preserve the class and all of its input coordinates, together with an additional constant coordinate.

Tokenwise maps. The admissible structural rows form a finite set, so any well-defined map on them has a continuous extension to compact neighborhoods of those rows. The vertex and incident-edge formulas above are such maps: identical rows give identical contributions. The query indicator is the class-conditioned predicate $q(\tau) = [\tau \text{ is an edge token and } \rho_0(\tau, u) = \rho_0(\tau, v) = 1/2]$. Only the query edge satisfies it in a simple graph, so $\sum_\tau q(\tau) = 1$. A tokenwise feed-forward network can approximate the vector $((\phi_f)_{f\in\mathcal{F}}, q)$ to arbitrary accuracy. A first transformer block with its attention output set to zero computes this vector into reserved coordinates.

Aggregation and output. In a second block, setting query and key projections to zero makes softmax attention uniform over the actual segment. Value and output projections write the means of these coordinates into a fresh group of coordinates; their residual input is zero. There are at most $\ell_{\max} = n + \binom{n}{2}$ distinct vertex and edge tokens, so the exact query-indicator mean is at least $1/\ell_{\max}$. Division by that mean is continuous on a compact neighborhood bounded away from zero. The second block's feed-forward network approximates the vector of ratios in (8) there. For example, errors smaller than $1/(2\ell_{\max})$ in the indicator mean keep the approximate denominator positive. Uniform continuity and the finite number of coordinates let the preceding approximation errors be chosen so that the final error is at most ϵ for every admissible query. A scalar scoring head can select any coordinate; parallel heads state the simultaneous version of the theorem. Additional blocks may be identities. For RA without context, a tokenwise network approximates $r_2(e_{uv})/r_1(e_{uv})$ directly at the query-edge token, where $r_1(e_{uv}) \geq 1/(n-1)^2$.

RMS normalization and gated feed-forward layers. The construction also applies to the pre-norm architecture in Section 5.2. Reserve a coordinate with value $\kappa > 0$, supplied identically by both class embeddings, and set every residual update to that coordinate to zero. For a hidden vector (κ, z) of width d_{model} with bounded z, its RMS-normalized nonconstant coordinates, multiplied by $\kappa/\sqrt{d_{\text{model}}}$, converge uniformly to z as κ grows; its constant coordinate converges to $\sqrt{d_{\text{model}}}$. Linear weights can absorb this fixed scale, and the nearly constant coordinate supplies the biases needed by a tokenwise network. The same argument handles the final RMS normalization and linear scoring head. A gated feed-forward layer can approximate an ordinary SiLU network by using its linear branch as a constant and its activated branch for the hidden units; SiLU is continuous and nonpolynomial, so its networks approximate continuous functions on compact sets (Leshno et al., 1993). Choose the tokenwise approximants first and then take κ large enough that the normalization perturbations fall within their error margins. Thus normalization does not impose a loss of token-count information in this construction. This is an existence argument in real arithmetic, not a conditioning or finite-precision guarantee, and gives no bound matching the experimental width.

General transformer approximation results provide broader context (Yun et al., 2020); the explicit construction here supplies the variable-cardinality sum and the query marker needed for this particular claim.

A.5 THE STRICT-SEPARATION PAIR

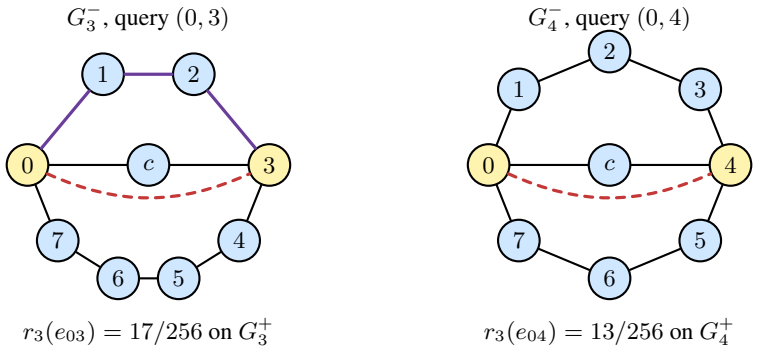


Figure 4: Strict separation beyond degree-weighted common-neighbor scores (Proposition 4). Both queries have endpoint degrees three and the same unique common neighbor c, of degree two, so CN is 1, AA is $1/\log 2$, RA is $1/2$, and every h_f equals $f(2)$. The dashed query edge is inserted before computing the walk encodings. The purple three-step path contributes an additional $1/64$ to the left query's depth-three return; the backtracking contribution is identical in both graphs.

Let G_j^- have vertices $\{0, \dots, 7, c\}$, cycle edges $\{i, i+1 \bmod 8\}$, and edges $\{0, c\}, \{j, c\}$, with query $(0, j)$, for $j \in \{3, 4\}$. Both graphs have nine vertices and ten edges, identical degree multisets, endpoint degrees three, and one common neighbor of degree two. They therefore agree on every h_f, including nonzero CN, AA, and RA, and on the degree-normalized scores above.

In G_j^+, both endpoints have degree four and each of their three other neighbors has degree two. Every length-three walk from 0 to j that uses the inserted edge is either $0, x, 0, j$ with $x \neq j$, or $0, j, y, j$ with $y \neq 0$, or $0, j, 0, j$. There are three walks of each of the first two types and one of the last. After dividing by $d_j^+ = 4$, their total is

$$3\frac{1}{4 \cdot 2 \cdot 4 \cdot 4} + 3\frac{1}{4 \cdot 4 \cdot 2 \cdot 4} + \frac{1}{4 \cdot 4 \cdot 4 \cdot 4} = \frac{13}{256}. \tag{9}$$

The only additional length-three walk for $j = 3$ is $0, 1, 2, 3$, of normalized weight $1/(4 \cdot 2 \cdot 2 \cdot 4) = 1/64$. For $j = 4$ there is no additional walk of that length. This gives the two values in Proposition 4. The distinction is in the query-edge token itself, and both graphs have the same n, so neither graph size, token count, nor a different preprocessing scale explains it. Taking the disjoint union of G_3^- and G_4^- gives the same separation for two queries within one fixed graph: inserting either query leaves its component's walk probabilities unchanged from the calculation above.

What depth three measures. The return $r_3(e_{uv})$ includes backtracking walks as well as simple three-step paths. A simple three-step path between the endpoints closes a four-cycle when the query edge is inserted, but $r_3(e_{uv})$ is not a pure four-cycle count. The example matches the backtracking contribution, exposing precisely the additional path contribution. It proves strict refinement of the stated heuristic family, without asserting that walk probabilities recover arbitrary motifs or all path-based heuristics.

Finite verification. The accompanying `theory_checks.py` checks the identities using exact rational arithmetic on all simple graphs through five vertices, including both query-edge states, isolates, both token extents, and endpoint-sequence ordering. It also verifies the separation pair and the query-marker averaging identity. These finite checks support the algebra; the proofs above establish the general statements.

B OPERATOR IDENTITIES AND QUERY-EDGE CORRECTION

Proof of Proposition 1. Since $TH^\top = A$, the diagonal blocks of $\vec{M}^2$ are $D^{-1}TH^\top = P$ and $H^\top D^{-1}T$, so for $k \geq 1$ $\vec{M}^{2k}$ has the blocks P^k and $H^\top P^{k-1} D^{-1} T$, and one more factor of $\vec{M}$

gives the odd powers. A factor $D^{-1}T$ on the right reads a matrix at column a' and divides by $d_{a'}$; a factor $H^\top$ on the left reads it at row b. □

The directed incidence graph is the subdivision of the symmetric digraph of G, and its two-step chain on the arcs is the walk on that digraph's line digraph, the backtracking counterpart of the non-backtracking walk.

With R the $n \times m$ incidence matrix of G, the undirected incidence graph has the adjacency and walk operators

$$A_{\mathcal{L}} = \begin{pmatrix} 0 & R \\ R^\top & 0 \end{pmatrix}, \qquad P_{\mathcal{L}} = \begin{pmatrix} 0 & D^{-1}R \\ R^\top/2 & 0 \end{pmatrix}, \qquad P_{\mathcal{L}}^2 = \begin{pmatrix} D^{-1}RR^\top/2 & 0 \\ 0 & R^\top D^{-1}R/2 \end{pmatrix}, \tag{10}$$

Here $D_{xx}^{-1} = 0$ at an isolate. Since $RR^\top = D + A$, the vertex block of two steps is $(J + P)/2$, where $J_{xx} = [d_x > 0]$, and hence is the lazy walk $(I + P)/2$ on the non-isolated support: from a vertex, one step goes to an incident edge and the next returns or moves on with equal probability. Orienting each edge removes this forced holding step and recovers the non-lazy walk P. The edge block iterates the same way: since $RR^\top D^{-1} = D(I + P)D^{-1}$,

$$\left(R^\top D^{-1}R/2\right)^k = \tfrac{1}{2}\, R^\top \tilde{P}^{k-1} D^{-1} R, \qquad \tilde{P} = (I + P)/2, \tag{11}$$

whose diagonal entry at $e = \{a, b\}$, the probability that the undirected walk returns to the edge in $2k$ steps, is

$$\tfrac{1}{2}\big((\tilde{P}^{k-1})_{aa}/d_a + (\tilde{P}^{k-1})_{bb}/d_b\big) + (\tilde{P}^{k-1})_{ab}/d_b, \tag{12}$$

the two crossing terms $(\tilde{P}^{k-1})_{ab}/d_b$ and $(\tilde{P}^{k-1})_{ba}/d_a$ merged by reversibility. The undirected edge return thus mixes the endpoints' own returns with the crossing term, where the oriented walk's arc return in the same $2k$ steps, $(P^{k-1})_{ba}/d_a$, has a crossing term alone. The choice of walk loses no information in the tables of powers: the lazy powers are binomial mixtures of the plain ones, $\tilde{P}^k = 2^{-k} \sum_{j \le k} \binom{k}{j} P^j$, so the plain powers up to a depth K and the lazy ones up to the same depth determine each other. What changes is what one coordinate means: a plain entry is a walk of length exactly k, and a lazy entry averages the walks of every length up to k, so each quantity a plain coordinate holds becomes a combination of several lazy ones.

Proof of Proposition 5. Row t of P^+ is $(A_{t\cdot} + \mathbf{1}_t^\top)/(d_t + 1)$ and every other row is unchanged, which gives Δ. The telescoping sum $\sum_{j=0}^{k-1} (P^+)^{k-1-j}(P^+ - P)P^j = (P^+)^k - P^k$ is (3) once ΔP^j is written row by row. The dependence follows by induction on k, each $((P^+)^{k-1-j})_{xt}$ being the identity at a smaller depth. □

For an isolated vertex on G, set its positive-depth row to zero and its depth-zero row to the identity rather than dividing by its degree in the reversibility step. For a query absent from G, gather its base pair probability from the endpoint rows; there is no existing-edge table row to gather. Correct the constituent probabilities before averaging or dividing by the degrees on G^+.

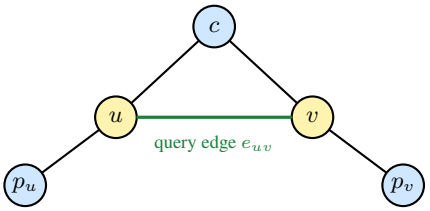
B.1 Why the correction, and not the label

Figure 5 makes the leak concrete on a graph of five vertices drawn for the purpose. For a training positive the entry $(P)_{uv}$ is $1/d_u$ and for a negative it is zero, so the length-one column alone separates the training set perfectly, and a model reaches near-zero training loss by reading it; at test time every positive is absent from G, the column reads zero for all of them, and the model ranks them as negatives. The deeper probabilities leak the same way, through the endpoints' degrees and through every walk that crosses the edge. Deleting the edge from each extracted subgraph, as SEAL and PENCIL do, closes the leak for what is recomputed inside the subgraph, and is why those methods recompute per query; a table evaluated once on the whole graph cannot be edited that way, and Proposition 7 replaces the deletion at the same exactness, for the whole graph's walk probabilities. The effect is not small. On pubmed, an arm whose context rows carried deep walk probabilities read on G trained to a loss of 0.008 by the twentieth epoch against 0.067 for the same arm without those rows, while its validation MRR fell from 0.103 to 0.026; the vertex table of the same runs carried a leak of the same kind one order deeper, and closing it moved citeseer from 0.134 to 0.166 under

Algorithm 1 The tokens of a query (u, v) and their encodings, every probability exact on G^+

Require: G with walk operator P and degrees d; largest power K; context budget β; the extent; tables $\mathrm{ret}_V[x, k] = (P^k)_{xx}$ and $\mathrm{ret}_E[\{a, b\}, k] = (P^k)_{ab}/d_b$ for existing edges and $k \in [1, K]$

$\mathrm{row}_t \leftarrow (\mathbf{1}_t^\top P^j)_{j\in[0,K]}$ for $t \in \{u, v\}$ ▷ batched over query endpoints
$B \leftarrow (N(u) \cup N(v)) \setminus \{u, v\}$; compute its incidence bits and mass μ from (14)
retain up to β vertices: highest mass at evaluation, weighted sampling in training
tokens $\leftarrow e_{uv}, u, v$, then retained ball vertices and induced edges within the extent
$d_t^+ \leftarrow d_t + [(u, v) \notin E]$ for $t \in \{u, v\}$; other degrees unchanged
for all tokens τ **do**
    gather base vertex returns and pair probabilities from $\mathrm{ret}_V, \mathrm{ret}_E$, or from $\mathrm{row}_u, \mathrm{row}_v$ for the query
    gather each constituent vertex's walk probabilities to the query endpoints by reversibility
    if $(u, v) \notin E$, carry these vertex-to-vertex probabilities onto G^+ using (3)
    form the token's relative walk probabilities and its class return using Corollary 2, with degrees d^+
    order whole endpoint sequences lexicographically; apply $\log(1 + n\rho)$, except to a vertex return, and standardize
return the token list with its inventories, one segment of the batch's flat sequence



entry	positive, on G	negative, on G	either, on G^+
d_u	3	2	3
$(P)_{uv}$	$1/3$	0	$1/3$
$(P^2)_{uv}$	$1/6$	$1/4$	$1/6$
$(P^2)_{uu}$	$11/18$	$3/4$	$11/18$
$(P^2)_{up_v}$	$1/9$	0	$1/9$
$(P)_{uv}/d_v$	$1/9$	0	$1/9$

Figure 5: The leak on a constructed graph of five vertices. A query (u, v) with one common neighbor c and one private neighbor each, p_u and p_v, drawn with the query edge e_{uv} present, as on G^+; on G the edge is present for a training positive and absent for a negative or a held-out positive. The entries are those of P on the drawn graph, which by Proposition 1 are the incidence walk's own: the returns of the vertex token u and of the query-edge token, and the relative walk probabilities between the tokens. Read on G as it stands, every entry above separates the two cases, and the length-one probability between the endpoints is the label itself: the degree, the return at u, the two-step walk $u \to v \to p_v$ through the edge, and the edge's own return all move with it. Read on G^+, the negative's entries are the positive's, and the two cases differ only through the rest of the neighborhood.

HeaRT. Every column a token reads here is its value on G^+, checked cell by cell against the kernels run on the edited graph.

B.2 The token set, its extent, and its budget

The extent. For a query (u, v) write $B = (N(u) \cup N(v)) \setminus \{u, v\}$ for the neighbors of its endpoints, each $x \in B$ with its two incidence bits $[x \sim u]$ and $[x \sim v]$, and $H = G^+[\{u, v\} \cup B]$ for the one-hop ball, the subgraph of G^+ they induce with the query edge present. The token set at extent $2r$, for $r \in \{1, 2, 3, 4\}$, is

$$T_{2r}(u, v) = \big\{\tau \in V(H) \sqcup E(H) : \mathrm{dist}_{\mathcal{L}(H)}(e_{uv}, \tau) \le r\big\}, \tag{13}$$

the query edge first, where $\mathcal{L}(H)$ is the incidence graph of H. In $\mathcal{L}(H)$ the query edge is at distance one from u and v, two from an edge joining an endpoint to B, three from a vertex of B, and four from an edge within B, so T_2 is the query tokens, T_4 adds the edges incident to u or v, T_6 the vertices of B, and $T_8 = V(H) \sqcup E(H)$ the whole of $\mathcal{L}(H)$. Where a budget binds, B is first cut to the vertices it keeps, and an edge is kept only with both of its endpoints.

Proposition 7 (Invariance to the state of the query edge)**.** *Let $G^- = G - (u, v)$. The token set of a query (u, v), every encoding its tokens carry, and every other input of the model, conditional on fixed attributes, preprocessing and sampling randomness, are functions of (G^-, u, v), whether or not (u, v) is an edge of G; a training positive on G and the same pair held out of G^- are one input.*

Proof. The ball, its bits, its degrees, and the mass (14) exclude the other endpoint from each neighborhood and so read G^-; the query edge is a token in either case; every encoding is read on $G^+ = G^- + (u, v)$; the attributes are the dataset's. □

Where a budget binds, B is ranked by the mass of the closed walks through the query that each vertex lies on,

$$\mu(x) = \frac{[x \sim u] + [x \sim v]}{d_x} + \frac{[x \sim u]\,[x \sim v]}{d_x} + \frac{1}{d_x} \sum_{y \in B} [x \sim y]\, \frac{[x \sim u]\,[y \sim v] + [x \sim v]\,[y \sim u]}{d_y}, \tag{14}$$

the three terms being the backtracking walk, the triangle u, x, v, and the walks through an edge inside the ball, each with its first step's factor $1/d_u$ or $1/d_v$ dropped. A common neighbor receives the extra triangle term, and edges joining the two sides contribute the final term. These weights favor such structures at fixed degree; they do not guarantee that every common neighbor outranks every private one. At evaluation the top of the list is kept; in training the same weights draw the set without replacement, so the cut is a sample rather than a rule. An edge token survives only while both its endpoints do. Where no budget binds, as on the Planetoid graphs, nothing is cut and the token set is a function of the graph and the query alone. The tokens of a batch are one flat list with segment offsets, and attention runs per segment on a variable-length flash kernel, so a batch costs the sum of its balls and not a rectangle padded to the largest. Algorithm 1 in Appendix B lists the steps from the query to its encoded tokens.

C Datasets

Table 4: The six graphs. Vertices, distinct undirected edges of the training graph, its mean degree, the width of the vertex attributes, the held-out positives of each split, and the negatives each protocol ranks a positive against: HeaRT's 500 hard negatives per positive on the Planetoid graphs, under its split, and on the OGB graphs the original protocol's one shared pool on ogbl-collab and ogbl-ppa and a pool per positive on ogbl-citation2.

		training	mean		validation	test	negatives per positive	
graph	vertices	edges	degree	attributes	positives	positives	original	HeaRT
cora	2,708	4,488	3.3	1,433	263	527		500
citeseer	3,327	3,870	2.3	3,703	227	455		500
pubmed	19,717	37,676	3.8	500	2,216	4,432		500
ogbl-collab	235,868	967,632	8.2	128	60,084	46,329	100,000	
ogbl-ppa	576,289	21,231,931	73.7	58	6,062,562	3,031,780	3,000,000	
ogbl-citation2	2,927,963	30,351,880	20.7	128	86,596	86,596	1,000	

Table 4 counts each graph as loaded. On ogbl-collab the arms keep the edges dated 2007 or later, as the leaderboard's entries do, and score test queries on the graph with the validation edges added, under the two data fields the record carries. The arm of Table 1 trains on the 46,684 pairs of 2017 between authors of the 2007 to 2016 rows, over the graph of those rows, as OGB drew the validation and test pairs from the years after the training span; on the graph before them they split 0.49 repeat collaborations, 0.17 at distance two, 0.11 at distance three, and 0.23 farther, against the test pairs' 0.46, 0.20, 0.14, and 0.21. Validation reads every training row and test every row and the validation edges, as for the other arms, and the rows a query's pair has and the years since an endpoint's last row are read on the graph the query reads. On ogbl-citation2 the query split is directed and each edge is stored once as (source, target), while the structural walk graph is symmetrized, as assumed by the theory.

D Settings and hardware on the OGB graphs

Every token carries its class's returns over the depths $[1, 4)$ and its relative walk probabilities over $[0, 4)$. The trunk is four unmasked blocks of width 256 with four heads. The token budget keeps

150 neighboring vertices on ogbl-ppa and 300 on ogbl-collab and ogbl-citation2. Each dataset has an attributed arm, with the symmetric edge lift of Section 5.2, and a structure-only arm, the setting PENCIL reports as "w/o Features" (Truong et al., 2026); on ogbl-collab the arm also reads the query pair's dated rows and trains on the latest year of them, as OGB drew its held-out pairs (Appendix C).

Each run trained on four GPUs of one node, one process per GPU with the batch per process: A100-SXM4-40GB for ogbl-collab and ogbl-citation2, and RTX PRO 6000 Blackwell for ogbl-ppa, whose node is the one Table 5 times. The ogbl-ppa and ogbl-citation2 arms train at dropout 0.1 and weight decay 0.01, against as many negatives as positives, drawn uniformly among the non-edges of the training graph once per seed. The ogbl-collab arm trains at dropout 0.5 and the same weight decay under the logistic pairwise loss, at a batch of 512 per process without accumulation and with each epoch's negatives drawn from a pool of sixteen times its positives; the dropout, the loss, and the inputs were chosen on validation in screens of the unsliced arm, and the slice and recency on validation of three of the eight seeds reported. Seed counts are eight on ogbl-collab, five on ogbl-citation2, and two on ogbl-ppa, against three to ten for the published rows; the two ogbl-ppa seeds of each arm ran under different revisions of the code. Selection fell at epochs 21 to 23 on ogbl-ppa and 38 to 48 on ogbl-citation2, against PENCIL's 15 epochs on ogbl-ppa and half an epoch on ogbl-citation2 (Truong et al., 2026, Table 5).

On ogbl-citation2 a run ticks a fixed sample of the pool of a thousand negatives per positive, drawn once under a seed of its own so every seed reads the same sample, stops and selects on that sample, and scores the selected checkpoint over the whole pool afterwards.

D.1 Attribute effects

The attribute lift improves the score by 0.0290, 0.1011, and 0.0458 on the three Planetoid graphs. It adds 0.0174 on ogbl-ppa and 0.0094 on ogbl-citation2, and reduces the score on ogbl-collab, by 0.0326 on the inputs of the other graphs and by 0.0445 with the pair's rows, dropout 0.5, and the pairwise loss, where the main table therefore reports the structure-only arm. The limited seed counts, three for Planetoid and two for ogbl-ppa, constrain conclusions about small differences. These effects measure the contribution of attributes conditional on the walk inventory; the heuristic-inclusion theorem concerns the structural input alone.

E Size and cost

Figure 1 places the model beside the published ones on ogbl-ppa by size: 4,420,213 parameters against PENCIL's estimated 13.6 million and GraphGPT's 2.03 billion (Appendix H); no parameter here grows with the number of vertices. Everything the encodings cost is the first three rows of Table 5: 641 seconds on the run that trained the ogbl-ppa structure-only arm, half of one training epoch and 0.007 of the run, memoized on disk under the graph, edge set, feature specification, and kernel source, so that a second run on the same graph pays none of it. PENCIL times the per-subgraph alternative on a synthetic graph of ten thousand vertices of mean degree 15 (Truong et al., 2026, Table 14): a two-hop subgraph of up to 1,862 vertices with its depth-8 random-walk encoding takes 60 ms, the encoding's time growing 8.6-fold from 7 ms at 842 vertices, against the 10.8-fold that $O(Ks^3)$ predicts for that growth in size. Once per query, at PENCIL's one negative per positive, the 42 million training queries of one ogbl-ppa epoch would take a month at depth 8 and ten days at the depth 3 used here, on a graph five times as dense. Parameter savings are not compute savings: the model trains for more epochs than PENCIL on the largest graphs, with estimated compute to the selected checkpoint about equal on ogbl-ppa and higher on ogbl-citation2 (Appendix D).

Table 5: What the encodings cost, from the phase timer of the run that trained the ogbl-ppa structure-only arm at seed 0 on four RTX PRO 6000. The first three rows are every pass that reads a walk probability; the remaining setup, loading the data and building the model, is 120 seconds. The propagation from the endpoints inside each of the first and third rows is 152 seconds.

phase	seconds
exact walk probabilities of the 46.4 M evaluation and training-positive pairs, to depth 3	248
tables of the whole graph	167
exact walk probabilities of the 21.2 M training negatives, to depth 3	226
one training epoch, median of 54	1221
one evaluation tick, median	467
the run	91410

F THE PROBE

This appendix gives the probe of Section 6.2 in full: every token set, every query, the context tokens with and without the query tokens beside them, and how the probes are fitted.

Token sets. The probe reads the walk encodings of Section 3.2, without attributes, on both token sets of Section 5.1: extent 4, the query tokens and the edges incident to them, which is what the ogbl-ppa arms use, and extent 8, every vertex and edge of the one-hop ball, which is what the Planetoid arms use. Table 6 reports each graph on its arm's set; Table 7 reports the Planetoid graphs on both.

Queries. The queries are the test positives, up to 5000 of them, each with twenty negatives: from the positive's own row of HeaRT negatives on the Planetoid graphs, and from the original protocol's shared random pool on ogbl-ppa. That is 11,067 queries on cora, 9,555 on citeseer, 93,072 on pubmed, and 105,000 on ogbl-ppa. The heuristics are nonzero only for a query with a common neighbor, and on the Planetoid graphs these are few: 0.125, 0.082, and 0.292 of the queries, and 0.43, 0.35, and 0.28 of the positives. On ogbl-ppa they are 0.95 of the positives but 0.052 of the queries, whose negatives are random pairs.

Inputs and probes. A token's input is its standardized row of walk probabilities, captured as the first block receives it. Five probes are fitted to $\log(1 + h)$ for each heuristic h computed on the training graph. The *query tokens* probe is a two-layer MLP on the rows of the query edge and its two endpoints. The four set probes pass each context token's row through a shared two-layer MLP, pool over the context tokens by summing or by averaging, and read the pool with a second two-layer MLP; *with query* gives the shared MLP the three query tokens' rows beside each context row, and *alone* gives it the context row only. Every MLP has width 64, and each probe is trained full-batch with Adam at a learning rate of 0.003 for 3000 steps on 0.8 of the queries and scored on the other 0.2; each entry of the tables is the mean over three such splits.

Table 6: Held-out R^2 of probes fitted to $\log(1 + h)$ for each heuristic h, over the queries with a common neighbor, each graph on the token set its arm in Table 1 uses, extent 8 on the Planetoid graphs and extent 4 on ogbl-ppa. *Query tokens*: an MLP on the rows of the query edge and its two endpoints. *Context*: a shared MLP on each context token's row beside the query tokens' rows, summed or averaged over the context tokens, then a second MLP. Each entry is the mean over three held-out splits of up to 5000 test positives with twenty negatives each; Table 7 has every query, both token sets, and the context alone.

	query tokens			context, summed			context, averaged		
graph	CN	AA	RA	CN	AA	RA	CN	AA	RA
cora	0.777	0.990	0.999	0.848	0.982	0.993	0.923	0.995	0.999
citeseer	0.799	0.974	0.998	0.948	0.987	0.995	0.965	0.994	1.000
pubmed	0.710	0.932	1.000	0.973	0.987	0.996	0.989	0.993	1.000
ogbl-ppa	0.968	0.982	0.999	0.967	0.980	0.988	0.980	0.990	1.000

What the table shows. Table 6 gives each graph on its arm's token set, over the queries with a common neighbor. The query tokens recover RA at 0.998 to 1.000 on every graph, the ratio of return probabilities in Proposition 3. The common-neighbor count is not read from them on the Planetoid graphs, at 0.71 to 0.80, since the query-edge token's depth-two return weights each common neighbor by its inverse degree, which only that neighbor's context token carries; with the context the count rises to 0.92 to 0.99. AA lies between the two and is nearly recovered from the query tokens, at 0.93 to 0.99. On ogbl-ppa the query tokens give the count at 0.968, and the context adds at most 0.012. Averaging the context is at or above summing it everywhere.

Table 7: Held-out R^2 of every probe, over all queries and over those with a common neighbor, on both token sets, extent 4 and extent 8 (Section 5.1). *Query tokens* reads the query edge and its two endpoints; *context with query* reads each context token beside them through a shared network, then sums or averages over the context tokens; *context alone* does the same without the query tokens beside each context token. The target is $\log(1 + h)$ on the training graph; each entry is the mean over three held-out splits.

	all queries			with a common neighbor		
probe	CN	AA	RA	CN	AA	RA
cora, extent 4						
query tokens	0.983	0.996	1.000	0.798	0.990	0.999
context with query, summed	0.985	0.995	0.996	0.834	0.985	0.994
context with query, averaged	0.992	0.998	1.000	0.908	0.994	1.000
context alone, summed	0.961	0.988	0.993	0.716	0.975	0.990
context alone, averaged	0.944	0.981	0.995	0.652	0.969	0.994
cora, extent 8						
query tokens	0.981	0.996	0.999	0.777	0.990	0.999
context with query, summed	0.986	0.994	0.995	0.848	0.982	0.993
context with query, averaged	0.993	0.998	1.000	0.923	0.995	0.999
context alone, summed	0.996	0.992	0.992	0.962	0.980	0.988
context alone, averaged	0.993	0.996	0.997	0.946	0.992	0.997
citeseer, extent 4						
query tokens	0.974	0.992	0.999	0.795	0.974	0.999
context with query, summed	0.982	0.990	0.997	0.858	0.968	0.996
context with query, averaged	0.992	0.997	1.000	0.931	0.988	1.000
context alone, summed	0.978	0.989	0.995	0.870	0.973	0.993
context alone, averaged	0.961	0.985	0.995	0.813	0.969	0.993
citeseer, extent 8						
query tokens	0.975	0.992	0.999	0.799	0.974	0.998
context with query, summed	0.993	0.996	0.997	0.948	0.987	0.995
context with query, averaged	0.995	0.998	1.000	0.965	0.994	1.000
context alone, summed	0.997	0.995	0.992	0.983	0.984	0.986
context alone, averaged	0.996	0.997	0.998	0.978	0.991	0.997
pubmed, extent 4						
query tokens	0.955	0.978	1.000	0.716	0.934	1.000
context with query, summed	0.986	0.992	0.999	0.915	0.977	0.999
context with query, averaged	0.988	0.993	1.000	0.925	0.980	1.000
context alone, summed	0.971	0.987	0.995	0.876	0.970	0.994
context alone, averaged	0.960	0.979	0.994	0.844	0.952	0.993
pubmed, extent 8						
query tokens	0.954	0.977	1.000	0.710	0.932	1.000
context with query, summed	0.995	0.995	0.997	0.973	0.987	0.996
context with query, averaged	0.998	0.998	1.000	0.989	0.993	1.000
context alone, summed	0.999	0.997	0.996	0.993	0.992	0.994
context alone, averaged	0.997	0.997	0.999	0.986	0.992	0.998
ogbl-ppa, extent 4						
query tokens	0.993	0.994	0.999	0.968	0.982	0.999
context with query, summed	0.992	0.993	0.993	0.967	0.980	0.988
context with query, averaged	0.996	0.997	1.000	0.980	0.990	1.000
context alone, summed	0.987	0.989	0.983	0.950	0.969	0.968
context alone, averaged	0.989	0.991	0.992	0.956	0.975	0.985

What the full table adds. Over all queries the query tokens read the common-neighbor count at 0.95 to 0.98 on the Planetoid graphs, well above their 0.71 to 0.80 on the queries with a common neighbor. They tell the two kinds of query apart exactly, since resource allocation is zero exactly when the count is, and on these graphs most queries have none; the harder subset is therefore the one Table 6 reports. The context alone, summed, reads the count at 0.96 to 0.99 at extent 8, which is the form of Proposition 3(ii)(a) at the largest token set: a sum over the context tokens of a function of each token's own walk probabilities. At extent 4 the same probe reads 0.72 to 0.88. The decoder in Appendix A shows that their own return and relative walk probabilities suffice in principle; the lower fitted accuracy measures the difficulty of learning that decoder, not missing information in the complete incident-edge set. The probe recovers the count more accurately at extent 8 than at extent 4: with the query tokens beside each context token and averaged, 0.92 to 0.99 against 0.91 to 0.93.

G SOURCES OF THE PUBLISHED ROWS

Table 8: The source of every published row of Table 1: the document, its table and page, the seed count it reports, and the variant shown, the better of a method's attributed and structure-only results where both are published. A dash is a seed count the source does not state. NCN's ogbl-citation2 number is its Table 7's, which differs from its Table 2's 0.8809. Optimized GAE sits with the identifier-based rows, as in Truong et al. (2026, Table 1), though its ogbl-collab cell reads the attributes alone.

row	source	seeds	variant
GCN, SAGE, SEAL, BUDDY, NCN, NCNC, GAE, Planetoid	Li et al. (2023), Table 4, p. 8	10	attributed
GCN, SAGE, SEAL, OGB	Li et al. (2023), Table 2, p. 5	10	attributed; structure-only on ppa
BUDDY, OGB	Chamberlain et al. (2023), Table 2, p. 10	–	attributed; structure-only on ppa
NCN, collab, ppa	Wang et al. (2024), Table 2, p. 8	10	attributed
NCN, citation2	Wang et al. (2024), Table 7, p. 16	10	attributed
NCNC, OGB	Wang et al. (2024), Table 2, p. 8	10	attributed
MPLP+, OGB	Dong et al. (2024), Table 2, p. 8	10	attributed on collab; structure-only on ppa, citation2
LPFormer, Planetoid	Shomer et al. (2024), Table 5, p. 9	–	attributed
LPFormer, OGB	Shomer et al. (2024), Table 2, p. 7	–	attributed
Optimized GAE, OGB	Ma et al. (2025), Table 3, p. 6	5	released as Refined-GAE; attributed, with a learned embedding per vertex on ppa and citation2 and attributes alone on collab (its Table 5)
Link-MoE, OGB	Ma et al. (2024), Table 2, p. 8	10	attributed; gating trained on a split of the validation set
GraphGPT, ppa, citation2	OGB leaderboard, 2026-09-09	10	pretrained; SMTP on ppa, d1n30 on citation2
PENCIL, Planetoid	Truong et al. (2026), Table 8, p. 25	5	structure-only on cora; attributed on citeseer, pubmed
PENCIL, OGB	Truong et al. (2026), Table 7, p. 24	3	structure-only on collab; attributed on ppa, citation2

H PARAMETER COUNTS OF FIGURE 1

Table 9: The points of Figure 1, and BUDDY, which the figure leaves out since its 324 parameters would stretch the axis by two decades for one point: each model's parameter count on ogbl-ppa, its test Hits@100 at the variant shown, and where the count comes from. Every count is exact, from a run record, a paper's table, the OGB leaderboard's parameter column as retrieved on 2026-09-24, or the released code instantiated at the command that produced the number, except PENCIL's, interpolated between its own two figures, which it measures at its $(1, 75)$ configuration rather than ogbl-ppa's $(1, 150)$. The configurations of GCN and SAGE released with Li et al. (2023) are the OGB reference architectures, whose leaderboard counts they reproduce to the parameter, at the Hits@100 that paper reports under the original protocol; Optimized GAE's leaderboard entry, 295,848,449, carries no convolution weights and scores 0.7374, so the paper's command is counted instead. Link-MoE and NCNC publish no count and are omitted.

model	parameters	Hits@100	source of the count
Vephormer (ours)	4,420,213	0.8457	the run record
PENCIL	about 13,600,000	0.7954	Truong et al. (2026, Table 13): 10.2 M at three layers, 27.3 M at eight; four by interpolation
GraphGPT	2,032,859,200	0.7655	OGB leaderboard, entry SMTP
Optimized GAE	296,636,417	0.7841	the released ogbl-ppa command, counted: a 512-wide embedding per vertex, two convolutions, and a five-layer predictor
MPLP+	147,794,531	0.6524	OGB leaderboard, entry MPLP
SEAL	709,122	0.4880	OGB leaderboard
SAGE	424,449	0.4102	the ogbl-ppa command released with Li et al. (2023), counted; the OGB leaderboard's GraphSAGE entry
GCN	278,529	0.2957	the ogbl-ppa command released with Li et al. (2023), counted; the OGB leaderboard's GCN entry
LPFormer	60,188	0.6332	the released ogbl-ppa command, counted
NCN	33,538	0.6119	OGB leaderboard
BUDDY, not plotted	324	0.4985	the released ogbl-ppa model, counted