Het-node2vec: second-order random walk sampling for heterogeneous graph embedding

Mauricio Soto-Gomez[1,*], Carlos Cano[2], Justin Reese[3], Peter N. Robinson[4], Giorgio Valentini[1,6], and Elena Casiraghi[1,3,5,6]

[1]AnacletoLab, Dipartimento di Informatica, Università degli Studi di Milano, Milan, Italy
[2]Department of Computer Science and Artificial Intelligence, Univerisidad de Granada, Granada, Spain
[3]Lawrence Berkeley National Laboratory, Berkeley, USA
[4]Berlin Institute of Health - Charité, Universitätsmedizin, Berlin, Germany
[5]Computer Science Department, Aalto University, Espoo, Finland
[6]ELLIS - European Laboratory for Learning and Intelligent Systems, Milan unit, Italy
[*]mauricio.soto@unimi.it

ABSTRACT

Many real-world problems are naturally modeled as heterogeneous graphs, where nodes and edges represent multiple types of entities and relations. Existing learning models for heterogeneous graph representation usually depend on the computation of specific, user-defined heterogeneous paths, or on the application of large, and often non-scalable, deep neural network architectures. We propose *Het-node2vec*, an extension of the node2vec algorithm, designed to embed heterogeneous graphs by capturing the topological and structural characteristics of the graph together with the type information associated with nodes and edges; this is performed by introducing a simple stochastic node-type switching strategy in second-order random walk processes. Empirical results on synthetic graphs, as well as on benchmark and real-world biomedical graphs, show that *Het-node2vec* achieves comparable performance with respect to state-of-the-art methods for heterogeneous graphs in node label prediction tasks.

Introduction

In the field of biology, medicine, social science, economics, and many other disciplines, the representation of relevant problems through complex graphs of interrelated concepts has fueled the increasing interest of the scientific community towards Graph Representation Learning (GRL[1]). Indeed, by learning low-dimensional representations of network vertices that reflect the network topology and the structural relationships between nodes, we can translate the graph representation of nodes and edges into a fully Euclidean embedding space that can be easily ingested into vector-based machine learning algorithms to efficiently carry out network analytic tasks, ranging from vertex classification and edge prediction to unsupervised clustering, node visualization, and recommendation systems[2–6].

Graphs built to model real-world problems may be homogeneous, when they use a single type of node and edge to represent the underlying knowledge, or heterogeneous, when nodes and edges belong to different types, i.e. they encode different categories of entities and relations. For example, in a biomedical knowledge graph, genes, diseases, phenotypes, drugs, and molecular functions, all represented by different node types, play different semantic and topological roles. Moreover, the downstream task may target only one of these categories, as in node-classification tasks restricted to specific node types, or link-prediction tasks between particular pairs of node types. In this setting, if all nodes and edges are treated uniformly by homogeneous embedding approaches, these type-specific roles may be blurred in the learned representation[7]; this effect can be especially relevant when the node or edge types involved in the task are under-represented or over-represented in the graph. Therefore,, to maximize the utility of graph embeddings for real-world applications, it is crucial that the learned representations are sufficiently expressive and type-aware to support a variety of downstream tasks, including tasks defined on highly imbalanced node- or edge-type distributions.

In the past decade, most research efforts have focused on homogeneous networks, with the proposal of matrix factorization-based methods[8], random walk (RW)-based methods[2,9], edge modeling methods[10], Generative Adversarial Nets[11], and deep learning methods[12,13]. Nevertheless, the highly informative representation provided by graphs that include different types of entities and relationships has motivated the development of increasingly complex heterogeneous networks[7], including Knowledge Graphs[14,15], which integrate and represent information from multiple data sources in the form of concepts (nodes characterized by multiple types) interconnected by edges representing relationships between them. Following these

advancements, Heterogeneous Graph Representation Learning (HGRL) algorithms have been recently proposed to process such complex, heterogeneous, real-world graphs[16–20].

The core issue with HGRL is to simultaneously capture the structural properties of the network and the information encoded by heterogeneous node and edge types; in other words, we need node and edge type-aware embeddings that can preserve both the topology of the graph and the type-specific roles of its nodes and edges.

In this context, from an algorithmic point of view, three main lines of research have recently emerged, all inspired by homogeneous network representation learning[7] (more details in Supplementary Section 1). The first leverages homogeneous RW-based approaches; these are based on the "distributional hypothesis"[1], first exploited to capture the semantic similarity of words[23], and then extended to capture the similarity between graph nodes[2]. The second exploits neural networks specifically designed for graph-structured data, using, e.g., convolutional filters[24], and more generally supervised feature learning through *Graph Neural Networks (GNNs)*[13,24]. The third research line[25–28] proposes *relation-learning methods* that view heterogeneous graphs as knowledge bases expressing relationships (edges) between source and destination nodes, and then learn the latent space where all such relationships are "optimally" represented.

These three families of methods have complementary roles (more details in Supplementary Section 1). In particular, RW-based methods are appropriate for generalist and scalable node embeddings and node-centric downstream tasks; GNN-based methods are well suited when rich node or edge attributes, end-to-end supervised training, and task-specific heterogeneous neighborhood aggregation are required; translational and relation-learning models are often preferable for explicit relation modeling and knowledge-graph completion.

Despite the impressive advancements achieved in recent years by the aforementioned methods (distributional RW-based approaches, GNN-based, and relation learning approaches), they show drawbacks and limitations. Indeed, methods based on the distributional hypothesis, which base the embeddings on RW sampling of neighborhoods, usually rely on the manual exploration of heterogeneous structures, i.e., they require human-designed meta-paths to capture structural and type-specific dependencies between the nodes and edges of the graph. This requires human intervention and non-automatic pre-processing steps for designing the meta-paths and the overall network scheme. Moreover, similarly to Heterogeneous GNN, in most cases, they treat separately each type of homogeneous network extracted from the original heterogeneous one and are not able to focus on specific types of nodes or edges that constitute the objective of the underlying prediction task (e.g., prediction of a specific edge type).

The successful application of GNNs to large heterogeneous graphs can be challenging because heterogeneous architectures often introduce type-specific parameters, relation-specific aggregation functions, or attention mechanisms that increase computational and memory requirements. Although scalable GNN variants, parameter-efficient architectures, and sampling-based approaches have been proposed[29], computational efficiency remains an important practical consideration when dealing with very large heterogeneous graphs. Moreover, some GNN methods augment graphs by leveraging human-designed meta-paths, thus showing the same limitation of distributional approaches, i.e., the need for human intervention and non-automatic pre-processing steps.

On the other hand, relation-learning approaches handle each triple with equal probability, leading to a bias towards the most represented relationships. Furthermore, while models like TransH attempt to induce better separability between relationships, they struggle with 1-to-N or N-to-N relationships, where two node types are linked by semantically different relationships[30]. More complex triple-learning techniques, such as ComplEx[28], suffer from computational inefficiency, hindering their practical application. To overcome these drawbacks, we propose a general framework to deal with complex heterogeneous networks, in the context of the previously discussed "distributional hypothesis" RW based research line. The proposed approach, which we named Het-node2vec to remark its derivation from the classical *node2vec* algorithm[2], can process heterogeneous multi-graphs characterized by multiple types of nodes and edges and can scale up with big networks, due to its intrinsic parallel nature.

Differently from classic meta-path-based RW approaches, *Het-node2vec* does not require the manual specification of complete heterogeneous meta-path schemas of fixed length or structure. Instead, it directly models the heterogeneous graph as a whole, without splitting it into homogeneous components. The proposed switching bias still performs type-guided path exploration, since it controls which node-type transitions are promoted or demoted during the RW process. However, this control is implemented through a local and stochastic transition rule, rather than through explicit user-defined meta-paths that must be enumerated in advance.

Het-node2vec can focus on specific nodes of the heterogeneous graph by biasing the walk toward node types of interest. Our proposed approach is particularly appropriate when we need to embed edge or node types that are underrepresented in the heterogeneous network, since the algorithm can focus on specific types of edges or nodes, even when they are largely outnumbered by the other types. At the same time, the proposed algorithms learn embeddings that are aware of the different types of nodes and edges of the heterogeneous network, while preserving the topology of the overall network.

[1]The distributional hypothesis was originally proposed in linguistics[21,22]. It assumes that "linguistic items with similar distributions have similar meanings", from which it follows that words (elements) used and occurring in the same contexts tend to convey similar meanings[22].

The main contributions of our work are the following:

- We introduce *Het-node2vec* that extends the node2vec[2] algorithm to handle heterogeneous graphs. *Het-node2vec* enables the generation of node embeddings that capture both the graph structural topology as well as node- and edge-type diversity.
- We design a novel type-aware RW sampling scheme that incorporates node and edge type information, allowing the algorithm to focus on specific types of nodes in heterogeneous graphs; by exploiting node- and edge-type information, our strategy improves the quality of the computed embeddings.
- We propose a special node-type switching mechanism that prioritizes transitions toward specific node types, allowing the algorithm to address problems where specific node/edge types are crucial for the prediction task under investigation.
- We provide a computationally efficient implementation of *Het-node2vec* that retains the scalability of the original node2vec algorithm.
- Experiments on synthetic graphs, benchmark heterogeneous graphs, and a real-world biomedical graph show that *Het-node2vec* improves performance in node classification and link prediction tasks with respect to homogeneous methods, achieving competitive results with state-of-the-art heterogeneous methods and maintaining the same overall complexity of scalable random-walk-based methods for homogeneous graphs.
- To ensure a F.A.I.R. (Findable, Accessible, Interoperable, and Reusable) description and encourage further research and applications involving heterogeneous graphs, we release an open-source implementation of *Het-node2vec*, fully integrated within the efficient GRAPE library[31].

Random walk-based methods

RW-based methods for graph representation learning generate node embeddings by leveraging the structural information encoded in sequences of nodes obtained through stochastic traversals of the graph. The fundamental principle underlying these approaches is that nodes appearing in similar network contexts should have similar vector representations, mirroring the distributional hypothesis from natural language processing. The methodology comprises two principal stages. First, random walks are sampled from each node in the graph, creating a corpus of node sequences that capture both local neighborhood structures and higher-order connectivity patterns[2]. These walks proceed by iteratively selecting neighboring nodes according to predefined transition probabilities, effectively exploring the topology of the graph in a manner that encodes structural proximities. Specifically, let X_t represent the node visited at step t in a RW on graph $G = (V,E)$, where V and E denote, respectively, the set of nodes and the set of edges. Each RW-generation model defines, at step t, the transition probability from node $X_t = v$ to one of its neighbors $X_{t+1} = x \in \mathcal{N}(v)$, being $\mathcal{N}(v)$ the set of one-hop neighbors of v, that is $P(X_{t+1} = x | X_t = v)$.

Second, dense vector embeddings of graph components are constructed by leveraging the word2vec paradigm[23], a well-known algorithm from natural language processing (NLP). Word2vec generates word embeddings from text corpora with the goal of capturing the semantic similarity of words. The core idea behind word2vec is to use the internal representation of a shallow neural network trained to predict either the context words from a target word (Skip-gram) or a target word from its context words (CBOW), where context words are defined as the neighboring words within a fixed window in the text.

The Skip-gram optimization framework, which we adopt throughout this work, aims to find an embedding $f : V \to \mathbb{R}^d$ mapping nodes to d-dimensional vectors which maximizes the log-probability of observing context nodes given each node in the random walk sequences:

$$\max_f \sum_{v \in V} \sum_{c \in \mathcal{N}_w(v)} \log P(c|f(v)), \tag{1}$$

where $\mathcal{N}_w(v)$ represents the multi-set of context nodes appearing within a window of size w around node v. By assuming conditional independence and symmetry in the embedding space, the conditional probability is computed using the softmax function as

$$P(c|f(v)) = \exp(f(c)^T f(v)) / \sum_{u \in V} \exp(f(u)^T f(v))$$

.

[2]Higher-order connectivity refers to relationships between nodes that are not necessarily directly connected, but are repeatedly co-visited along multi-step paths.

However, computing the term in the denominator of the softmax function is computationally prohibitive for large graphs. To address this, a standard choice is to use negative sampling[23], which replaces the softmax probability definition with a binary classification objective distinguishing observed context pairs from randomly sampled negative pairs, formulated as:

$$\log \sigma(f(c)^T f(v)) + \sum_{i=1}^{k} \mathbb{E}_{v_i \sim P_n(v)} [\log \sigma(-f(v_i)^T f(v))] \tag{2}$$

where σ is the sigmoid function, k is the number of negative samples, and $P_n(v)$ is the negative sampling distribution.

Recalling that the dot product measures similarity between vectors, the objective function maximizes the similarity of embedded representations for neighboring nodes, while minimizing it for non-neighboring nodes. This preserves the graph's topological relationships in the embedding space.

Homogeneous random walks sampling

Two of the most popular methods are DeepWalk[9] and node2vec[2], which employ first-order and second-order transition probabilities, respectively.

In DeepWalk, the transition probability at step t from node $X_t = v$ to one of its neighbors $X_{t+1} = x \in \mathcal{N}(v)$, over the edge vx depends solely on the weight of the edge vx between v and x, that is $P(X_{t+1} = x | X_t = v) \propto w_{vx}$. Node2vec extends DeepWalk by biasing the walk via a second-order transition probability $P(X_{t+1} = x | X_t = v, X_{t-1} = r)$ that also depends on the node visited in the previous step, $X_{t-1} = r$. In more detail (see also Figure 1a):

$$\pi_{rvx} = \alpha_{pq}(r,v,x) \cdot w_{vx} \tag{3}$$

$$\alpha_{p,q}(r,v,x) = \begin{cases} \frac{1}{p} & \text{if } d_{rx} = 0 \\ 1 & \text{if } d_{rx} = 1 \\ \frac{1}{q} & \text{if } d_{rx} = 2 \end{cases} \tag{4}$$

$$P(X_{t+1} = x | X_t = v, X_{t-1} = r) = \pi_{rvx} / \sum_{z \in \mathcal{N}(v)} \pi_{rvz} \tag{5}$$

where π_{rvx} denotes the unnormalized transition probability and α_{pq} is a parametric function depending on the hop-distance d_{rx} between nodes r and x. The transition probability is finally obtained as the normalization of the function π_{rvx} with respect to the set of neighbors of v (Equation 5).

Hyperparameters p (the *return* or *inward* hyperparameter) and q (the *in-out* or *explore*) bias the walk to either favor a depth-first (DFS)-like or breadth-first (BFS)-like search, thereby controlling the tendency of the RW to explore new regions or to revisit nodes in the graph, respectively. These dynamics are illustrated in Figure 1a, which shows a step of a second-order RW. Supplementary Section 2 reports experimental examples illustrating the effect of parameters p and q on the resulting embeddings for graphs with simple topological structures.

Het-node2vec: Heterogeneous random walk sampling

Heterogeneous graph contains nodes belonging to different types, such as genes, diseases, papers, authors, or businesses, and edges representing relations between them. The downstream task may involve only a subset of these types, for example classifying nodes of a given type, predicting links between two selected node types, or focusing the embedding on under-represented node or edge types. In such settings, the type of the node reached by a RW step influences which structural and type-specific contexts are sampled and therefore obtain a richer representation in the computed embeddings. The goal of *Het-node2vec* is therefore to modify the node2vec transition probabilities so that the RW can either preserve, increase, or target type heterogeneity, depending on the embedding objective.

Het-node2vec generalizes the node2vec algorithm by sampling type-aware RWs. This can be accomplished by introducing a "switch" function that biases the way the RW moves between different node types, providing a mechanism to incorporate node-type information into the RW generation process.

Let $G = (V, E)$ be a heterogeneous graph where $\phi : V \to \Sigma_\phi$ denotes the function defining the type of the graph nodes.

Consider a RW currently residing at node $X_t = v$, coming from node $X_{t-1} = r$; if x is a neighbor of v, *Het-node2vec* defines the second-order transition probability of stepping to $X_{t+1} = x$ as:

$$\hat{\pi}_{rvx} = \beta_s(v,x) \cdot \pi_{rvx} = \beta_s(v,x) \cdot \alpha_{pq}(r,v,x) \cdot w_{vx} \tag{6}$$

$$P\Big(X_{t+1} = x | X_t = v, X_{t-1} = r\Big) = \hat{\pi}_{rvx} / \sum_{z \in \mathcal{N}(v)} \hat{\pi}_{rvz}, \tag{7}$$

where $\hat{\pi}_{rvx}$ denotes the unnormalized transition probability, α_{pq} is defined as in node2vec, and w_{vx} is the weight of the edge vx connecting v and x. With respect to node2vec (Equation 3), the transition probability in Equation 6 incorporates a new *node-type switching function* $\beta_s(v,x)$. It depends on the types of nodes v and x involved in the transition and on a user-set parameter s (the *node-type switching* weight), whose value allows biasing the transition probabilities according to node types[3].

In this work we provide two definitions of the node-type switching function β_s, described in the following subsections, to enable flexible random-walk sampling strategies that exploit node- and edge-type information to suit specific embedding objectives. The first definition of β_s, which we call "generic node-type switching," promotes/demotes type heterogeneity; alternatively, we define a "special node-type switching" function to preferentially traverse (or avoid) particular node and edge types. Such strategies prove particularly valuable for tasks targeting specific node types or addressing under-represented node types in the graph.

Generic node-type switching. The basic *Het-node2vec* algorithm, that we refer to as **generic switching**, leverages parameter s to control switching between nodes of different types in order to promote or demote type heterogeneity. In this approach, the switching probability in a transition from node v to node x depends solely on whether the types differ, i.e., on $\phi(x) \neq \phi(v)$. Specifically, function β_s is defined as follows:

$$\beta_s(v,x) = \frac{1}{s} \text{ if } \phi(v) \neq \phi(x), \text{ and 1 otherwise} \tag{8}$$

The theoretical effect of the switching function can be interpreted in terms of relative transition probabilities. Indeed, for two candidate next nodes $x, z \in \mathcal{N}(v)$, the ratio between their transition probabilities is:

$$\frac{P(X_{t+1} = x \mid X_t = v, X_{t-1} = r)}{P(X_{t+1} = z \mid X_t = v, X_{t-1} = r)} = \frac{\beta_s(v,x)}{\beta_s(v,z)} \cdot \frac{\alpha_{pq}(r,v,x) w_{vx}}{\alpha_{pq}(r,v,z) w_{vz}} = \frac{\beta_s(v,x)}{\beta_s(v,z)} \cdot \frac{\pi_{rvx}}{\pi_{rvz}}. \tag{9}$$

This probability ratio corresponds to the standard node2vec probability ratio, i.e., the second factor on the right-hand side of Equation 9, further rescaled by the type-switching ratio $\beta_s(v,x)/\beta_s(v,z)$. In practice, Equation 9 highlights that β_s does not define an absolute transition probability by itself, but locally rescales the relative probability of selecting neighbors of particular types before the final normalization over $\mathcal{N}(v)$.

For example, in the generic switching strategy, suppose that x has a different type from v ($\phi(x) \neq \phi(v)$), whereas z has the same type as v ($\phi(z) = \phi(v)$). If x and z have the same distance from r, so that the node2vec weight $\alpha_{pq}(r,v,x) = \alpha_{pq}(r,v,z)$, and if the corresponding edge weights are equal $w_{vx} = w_{vz}$, then the node2vec probability ratio equals 1, and the overall probability ratio of selecting x instead of z equals $1/s$. More generally, when the node2vec ratio differs from 1, the relative transition probability is multiplied by 1/s.

Thus, in the generic node-type switching strategy, values $1/s > 1$ (i.e., $s < 1$) favor type-switching transitions, whereas values $1/s < 1$ (i.e., $s > 1$) demote them.

The realized effect of a given value of s is therefore local and depends on the type composition of the neighborhood of v, the edge weights, the node2vec second-order bias α_{pq}, and the homophily or heterophily of the graph around the current node. Consequently, type switching changes the transition probabilities locally at each random-walk step, rather than imposing a fixed global proportion of visits to each node type. This local nature of the switching mechanism also explains why there is no universal closed-form threshold of s at which embeddings of nodes with different types are expected to form clearly separated clusters. Such separation depends on the transition-probability ratio in Equation 9, as well as on local type composition, edge weights, degree distribution, graph topology, and the downstream dimensionality-reduction method used for visualization.

Besides being illustrated experimentally in Figure 5 and further discussed in the subsection "Experimental Results – Grid topology", this behavior is schematized in Figure 1, where we show how *Het-node2vec* differs from node2vec for the same local neighborhood.

The RW is currently at node v_{ϕ_1} and arrived there from node r_{ϕ_2}. At the next step, the walk can either return to r_{ϕ_2} or move to one of the candidate neighbors x_*. In node2vec (Figure 1a), the unnormalized transition score depends only on the edge weight and on the second-order topological bias $\alpha_{pq}(r,v,x)$, which is determined by the distance between the previous node r and the candidate next node x. Thus, returning to r receives score $1/p$, moving to a neighbor also adjacent to r receives score 1, and moving farther away from r receives score $1/q$.

In *Het-node2vec* (Figure 1b), the same node2vec topological score is multiplied by the additional type-switching factor $\beta_s(v,x)$. In the generic switching case shown in the figure, this factor is equal to $1/s$ when the candidate next node has a type

[3]For the sake of simplicity, we will omit the arguments of functions β_s whenever the context is clear.

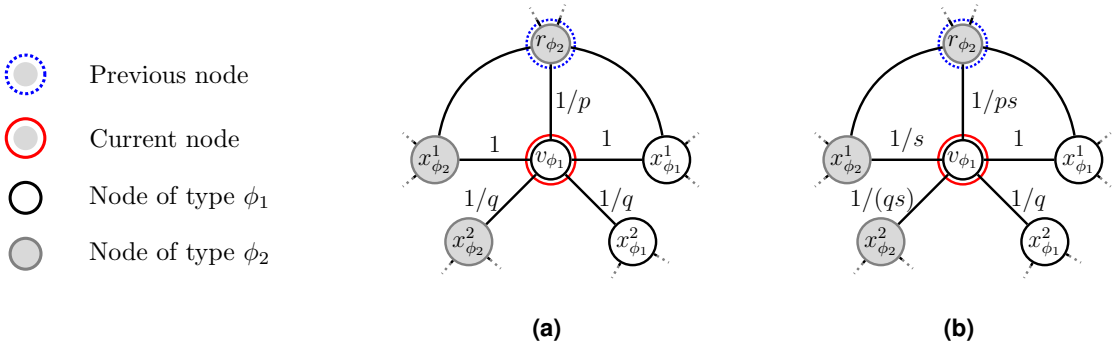


Figure 1. Comparison between unnormalized transition probabilities for node2vec (a) and *Het-node2vec* (b) in an unweighted heterogeneous network with heterogeneous nodes. The figure depicts one local step of a second-order RW, where the current and previous nodes have different types. The color of nodes represents their type. To simplify the notation, edges are unweighted and nodes x with type ϕ_* are denoted as x_{ϕ_*}. Edge labels depict the value of the function π_{rvx} (a) and $\hat{\pi}_{rvx}$ (b) without considering edge weights for a second-order RW transitioning from v_{ϕ_1}, i.e. $X_t = v_{\phi_1}$ and coming from node $X_{t-1} = r_{\phi_2}$.

different from the current node v_{ϕ_1}, and is equal to 1 otherwise. Consequently, transitions from v_{ϕ_1} to nodes of type ϕ_2 receive an additional multiplicative factor $1/s$, whereas transitions to nodes of type ϕ_1 keep the original node2vec score. For example, the return transition from v_{ϕ_1} to r_{ϕ_2} changes from $1/p$ in node2vec to $1/(ps)$ in *Het-node2vec*, while a transition to a same-type neighbor with topological score $1/q$ remains $1/q$.

The figure therefore shows that *Het-node2vec* preserves the second-order exploration behavior controlled by p and q, while locally rescaling candidate transitions according to node-type changes.

Special node-type switching. Many real-world applications involve downstream tasks that focus on a specific subset of node types, which we refer to as "special node types". For example, in a biomedical heterogeneous graph, nodes may represent diseases, genes, miRNAs, drugs, phenotypes, and biological processes, while edges may represent associations or interactions among these entities. If the downstream task is, for example, disease-label prediction, disease-gene association prediction, or miRNA-gene interaction prediction (as in the case-study we present i the "Experimental results" section), then the node types directly involved in the task may constitute only a small fraction of the graph. In this case, standard RWs may under-sample the neighborhoods of the target types, producing embeddings that are dominated by more frequent node or edge categories. Promoting transitions toward special node types can therefore be useful when the target types are rare, under-connected, or under-sampled, whereas demoting transitions toward a dominant type can be useful when that type overwhelms the sampled contexts and reduces the diversity of the generated walks. To address these scenarios, the "special node-type switching" strategy partitions the node types into special Σ_{ϕ_S} and non-special $\Sigma_{\phi_{NS}}$ node sets. A node x is "special" if $\phi(x) \in \Sigma_{\phi_S}$, otherwise it is "non-special." By adjusting the switching parameter, the algorithm can promote or demote transitions involving these special types according to the downstream task and the type distribution of the graph. The simplest bias function for special node-type switching is:

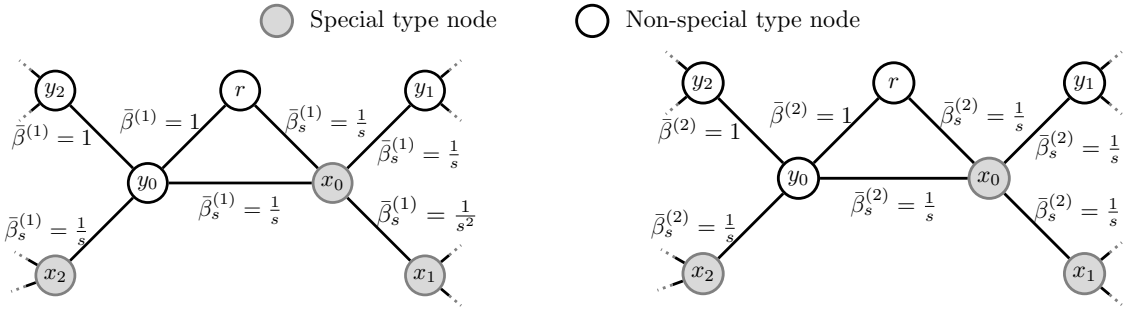
$$\beta_s^{(1)}(v,x) = 1/s \text{ if } \phi(x) \in \Sigma_{\phi_S}, \text{ and 1 otherwise} \tag{10}$$

which uniformly promotes or demotes jumps toward special nodes regardless of the type of the source node. This means that the bias affects equally both transitions from non-special to special nodes and transitions between special nodes. However, in some scenarios, this uniform approach may be limiting. When $s \ll 1$ or $s \gg 1$, the model could generate RWs to either concentrate almost exclusively on special nodes or avoid them altogether, thus failing to capture the heterogeneity of node neighborhoods. To overcome this limitation, we propose a refined bias function:

$$\beta_s^{(2)}(v,x) = 1/s \text{ if } (\phi(v) \notin \Sigma_{\phi_S}) \text{ and } (\phi(x) \in \Sigma_{\phi_S}), \text{ and 1 otherwise} \tag{11}$$

which applies bias *only* when transitioning from a non-special node to a special node. When the RW is at a special node, no bias is applied, allowing unbiased exploration among special and non-special nodes alike. This approach encourages broader, more heterogeneous walks that better reflect the graph's structural and type diversity.

Both strategies use parameter s to promote or demote transitions toward special nodes when the walk is at a non-special node. The key difference is that the first strategy continues to bias transitions *within* special nodes, while the second avoids such bias to preserve heterogeneity. Empirical results reported in Supplementary Section 9 (Subsection "Link



$$\bar{\beta}^{(1)}_s(v,x) = \begin{cases} 1/s & \text{if } \phi(v) \in \Sigma_{\phi_S} \oplus \phi(x) \in \Sigma_{\phi_S} \\ 1/s^2 & \text{if } \phi(v) \in \Sigma_{\phi_S} \wedge \phi(x) \in \Sigma_{\phi_S} \\ 1 & \text{otherwise,} \end{cases}$$

(a) Implementation of $\beta^{(1)}$

$$\bar{\beta}^{(2)}_s(v,x) = \begin{cases} 1/s & \text{if } \phi(v) \in \Sigma_{\phi_S} \text{ or } \phi(x) \in \Sigma_{\phi_S} \\ 1 & \text{otherwise.} \end{cases}$$

(b) Implementation of $\beta^{(2)}$

Figure 2. Implementation of special node-type switching strategies. (a) In strategy $\beta^{(1)}_s$, the bias is 1 for edges connecting two non-special nodes, $1/s$ if the edge connects a non-special node to a special node, and $1/s^2$ if the edge connects two special node types. (b) In strategy $\beta^{(2)}_s$, biases are applied to all edges having a special node at one (or both) endpoints. Edges connecting two non-special nodes are not biased.

prediction performance comparison between node-type switching strategies", and Supplementary Table 3) demonstrate that $\beta^{(2)}_s$ consistently outperforms $\beta^{(1)}_s$. Therefore, $\beta^{(2)}_s$ is adopted for all experiments presented in the "Experimental Results" Section.

As for the generic node-type switching strategy, the theoretical effect of the special node-type switching strategy can be interpreted through the relative transition probabilities in Equation 9.

Unlike the generic switching strategy, which can be used to generate task-independent type-aware embeddings, the special switching strategy is partially task-oriented because the special node types must be selected according to the downstream prediction task.

Implementation, complexity and scalability of *Het-node2vec*. The introduction of a type switching strategy could potentially increase the type complexity of the random walk generation process by introducing further conditions for the computation of transition probabilities. Nevertheless, our implementation of *Het-node2vec* retains the same space and time complexity as node2vec. Indeed, in the case of the generic node-type switching strategy, the values of the switching bias s depends solely on the types of the nodes involved in the edge-transition. They can be therefore precomputed by simply scaling the weights of edges connecting nodes of different types by a factor $\beta_s = 1/s$. This avoids increasing the time complexity of the random walk sampling process.

In the case of the special node-type switching, we reformulated the $\beta^{(1)}_s$ and $\beta^{(2)}_s$ functions to make them independent of the transition direction. Therefore, these switching biases can also be precomputed and integrated into the edge weights prior to RW execution.

In more detail, Figure 2 depicts the edge-weight redefinitions implementing equivalent versions of the two special node-type switching strategies. This equivalence between the *Het-node2vec* formulation and its implementation is possible since transition probabilities are obtained from the normalization of biases to sum up to 1. For the sake of clarity, a detailed proof of this equivalence is provided in Supplementary Section 6.

Regarding the method complexity, in its simplest form, our RW implementation only stores the node neighborhood, which requires $\mathcal{O}(|E|)$ space. In the case of second-order walk, transition probabilities can be computed efficiently by storing the two-hop neighborhood of graph nodes with a $\mathcal{O}(\bar{d}^2|V|)$ space requirement, where $\bar{d}$ denotes the mean degree of a node, which is small in most applications[2]. Regarding time complexity, as a Markovian generation process, RW generation enables the reuse of RW samples. Namely, by constructing a walk of length $L > l$, it is possible to generate $(L-l)$ RWs of length l with time complexity $\mathcal{O}\left(\frac{L}{l(L-l)}\right)$ per sample[2]. Furthermore, the sampling procedure can be optimized via parallelization and the use of succinct data structures[31,32].

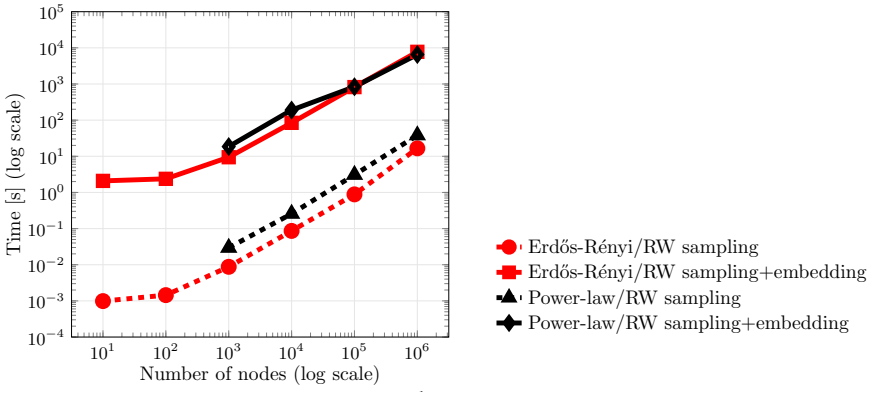


Figure 3. Computation time evolution of the *Het-node2vec* representation in Erdős-Rényi graphs with an average degree of 10 and random power-law graphs with exponent $\alpha = 2.2$. For each family, graphs nodes and edges were randomly assigned to one of ten classes.

Finally, as aforementioned, in our node-type switching strategies, the bias function β_s can be precomputed and incorporated as a factor in the edge weights. This one-time pre-processing has $\mathcal{O}(|E|)$ time complexity. The previous discussion ensures that using *Het-node2vec* with a special switching strategy does not increase complexity in either space or time compared to node2vec.

We explored the scalability of the model in a way similar to Grover et al.[2]; we conducted an empirical analysis of computation time for the generic node-type switching strategy in *Het-node2vec*, measuring both RW sampling and complete embedding construction (Figure 3). We tested two random graph models: Erdős-Rényi graphs with edge probability yielding average degree 10, and power-law degree distribution characterized by an exponent $\alpha = 2.2$[33], which is a common graph topology in many practical applications. For each family, graphs nodes and edges were randomly assigned to one of ten classes. Starting from each node, 10 RWs of length 100 were generated using the parameters $p = 0.25, q = 4$, and $s = 0.5$. The set of RW samples was used as the input of a Skip-gram model to produce a vectorial embedding of size 100. Computations were performed on a system equipped with an AMD EPYC ROME 7452 CPU (32 cores at 2.3 GHz) and a RAM of 1024 GB. Under this experimental setting, Random walk sampling for graphs up to 1M nodes takes less than one minute, while sampling process plus the embedding takes less than 2.8 hours. As can be seen in Figure 3, for graph sizes in the range $[10^3, 10^6]$ the entire embedding construction exhibits a linear scaling.

Experimental results

To assess the capabilities of *Het-node2vec* in enhancing the representation of heterogeneous graphs by incorporating node-type information, we conduct four sets of experiments, progressively evaluating its components and overall performance.

First, we present a qualitative analysis to investigate the behavior of the generic node-type switch implemented in *Het-node2vec*. To this aim, we compare representations obtained by varying model parameters (p, q, and s) on synthetic networks with simple topologies, allowing us to isolate and analyze the structural effects of the generic switching mechanism.

Second, we analyze the effect of the special node-type switch strategy on real-world data. In particular, we perform prediction tasks on under- and over-represented node types in the benchmark Cora graph[4], evaluating how this targeted switching mechanism improves representation quality for specific node categories.

Third, we evaluate the complete *Het-node2vec* framework using the HNE benchmark framework[16], which includes heterogeneous benchmark graphs and state-of-the-art HGRL methods for node label prediction. Within this setting, we first analyze the impact of the node-switching weight on performance metrics, assessing the contribution of the node-type information. We then compare *Het-node2vec* against popular state-of-the-art HGRL methods, demonstrating that its representations are competitive and often outperform alternative approaches.

Finally, we conduct a case study on a biological Knowledge Graph representing relationships between different biological entities. In this graph, we perform an edge prediction task between microRNA (miRNA) and genes. This experiment further illustrates how incorporating node-type information associated with the target relation through *Het-node2vec* can significantly improve prediction metrics in a real-world biomedical application.

In addition to comparing *Het-node2vec* with existing HGRL methods, the experimental design includes several ablation-like analyses aimed at isolating the contribution of each component of the proposed sampling strategy. First, the homogeneous

[4] https://graphsandnetworks.com/the-cora-dataset/

node2vec setting is recovered by setting $1/s = 1$ for all transitions; this provides a no-type-switching baseline while retaining the second-order node2vec bias controlled by p and q. Second, when comparing with existing HGRL methods, we also include DeepWalk[9], which represents a more reduced homogeneous RW baseline in which both the second-order node2vec biases and the type-switching bias are removed, corresponding to $p = 1$, $q = 1$, and $s = 1$. Third, the generic node-type switching strategy isolates the effect of promoting or demoting transitions between different node types, without using task-specific information. Fourth, the special node-type switching strategy isolates the effect of biasing the RWs toward task-relevant node types. Finally, in the RNA-KG case study, the edge-type-oriented switching experiment evaluates whether focusing the RWs on the endpoint types of the target relation improves link prediction. Together, these comparisons separately assess the effect of homogeneous structural sampling, unbiased homogeneous RW sampling, generic type-aware sampling, task-oriented node-type sampling, and target-relation-oriented sampling.

Analysis of the generic node-type switch in *Het-node2vec* on synthetic graphs

This section compares the embeddings obtained from *Het-node2vec* using different parameter configurations for simple graph topologies. In this section, edges are considered unweighted and homogeneous, while nodes have heterogeneous types. For each parameter configuration, embeddings are computed by generating 10 RWs of length 128 from each node. These paths are used as input to a Skip-gram architecture with a window size equal to five to generate an embedding of the nodes into a 10D space. Finally, 2D representations for visual analysis of results are obtained using t-SNE mapping[34].

Random graph. Figure 4 shows embeddings of a random graph generated using an Erdös-Rényi model with 2,000 nodes, edge probability 0.01, and three node types randomly assigned via a truncated exponential distribution. Since this model yields uniform connection probabilities between nodes, the generated graphs lack specific topological structure. As expected, under the generic node-type switching strategy, large values of s (i.e., $1/s \ll 1$) promote type-based node clustering because RWs are discouraged from switching between node types and therefore preferentially generate paths within the same type. Conversely, varying parameters p produces uniformly distributed representations that fail to capture type-based graph graph ; a similar behavior is obtained by varying exploring parameter q. This behavior together with additional experiments are presented in the Supplementary Section 3 (Supplementary Figures 2, 3, and 4).

Grid topology. Figure 5 depicts *Het-node2vec* applied to a synthetic grid topology where each row corresponds to a different node type, shown with different colors in the figure. The embeddings obtained for different values of the switching parameter s ($p = 1$, $q = 1$) under the generic node-type switching strategy of *Het-node2vec* are shown in Figure 5a; Figure 5b shows the influence of varying the return parameter p ($q = 1$) in node2vec.

As can be seen in Figure 5a, proper tuning of the switching parameter s yields type-aware embeddings. Small values of the node-type switching probability β_s ($1/s < 1$, e.g., $1/s = 10^{-3}$) tend to cluster nodes by type; large node-type switching probabilities β_s ($1/s = 10^3$) group together nodes belonging to the same grid columns, regardless of type.

Setting intermediate values of s modulates the switching probability, while permitting both inter-type and intra-type walks. This effect can be observed in Figure 5a for values of $1/s = 0.1$ and $1/s = 10$. When $1/s = 0.1$, type switching is discouraged, leading to nodes of the same type being embedded closely together and forming type-specific clusters. However, a few type switches are still permitted, resulting in clusters that group adjacent rows in the grid.

On the other hand, when $1/s = 10$, the probability of moving between node types (rows) increases, leaving room for intra-type visits; we observe clusters containing nodes with completely different types (each cluster contains nodes from one column of the grid), while clusters that are close to each other contain nodes from neighboring columns. When $1/s = 10^3$ the type switching transition probability is close to 1 and clusters are composed by nodes of different types according to the columns of the grid; in this case the topological closeness of different columns is lost, and, in practice jumps between columns are impeded.

Note that the transition between type-driven and topology-driven clusters is gradual. Additional intermediate values of $1/s$ between 0.1 and 1 are reported in Supplementary Information Figure 2 and Figure 4. It is important to consider that, consistently with the transition-probability ratio in Equation 9, these values should be interpreted as inducing intermediate local preferences for type switching or type preservation, rather than as defining a universal global threshold for cluster separation.

Conversely, node2vec ($1/s = 1$) effectively captures the topological structure of the grid; it is however unable to cluster together nodes of the same type, even when the values of parameters p and q vary in all their range (Figure 5b). Note that the representations obtained in node2vec by varying the inward parameter p ($q = s = 1$) mirror those obtained by varying q ($p = s = 1$). This can be explained by considering the fact that the grid contains no triangles (a complete subgraph of three vertices); in this case, random walk transition probabilities can only take the values $1/p$ or $1/q$. Therefore, the ratio q/p fully characterizes the parameter space and quantifies the relation between the local/global type of the generated paths.

A broader comparison of parameter sensitivity can be found in the Supplementary Sections 2, 3, 4, 5 where different embeddings are constructed by varying parameters p, q, and s over synthetic and real-world graphs.

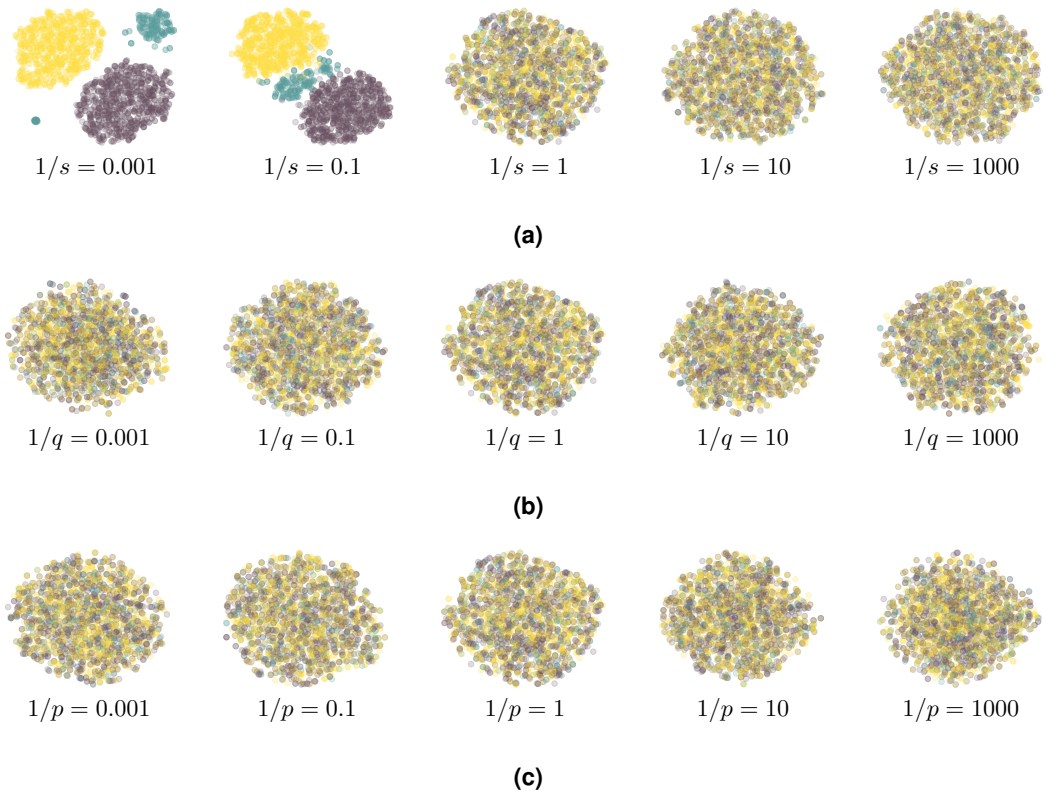


Figure 4. t-SNE visualization comparing *Het-node2vec* embeddings computed with different settings of the s and p parameters. The heterogeneous random (Erdös-Rényi) graph has three node types, distributed according to a truncated exponential function. (a) *Het-node2vec* embeddings of the random graph for varying node-type switching values s with $p = q = 1$. (b) *Het-node2vec* embeddings of the random graph from different inward switching values q and $p = s = 1$. (c) *Het-node2vec* embeddings of the random graph for varying inward-bias values p with $q = s = 1$.

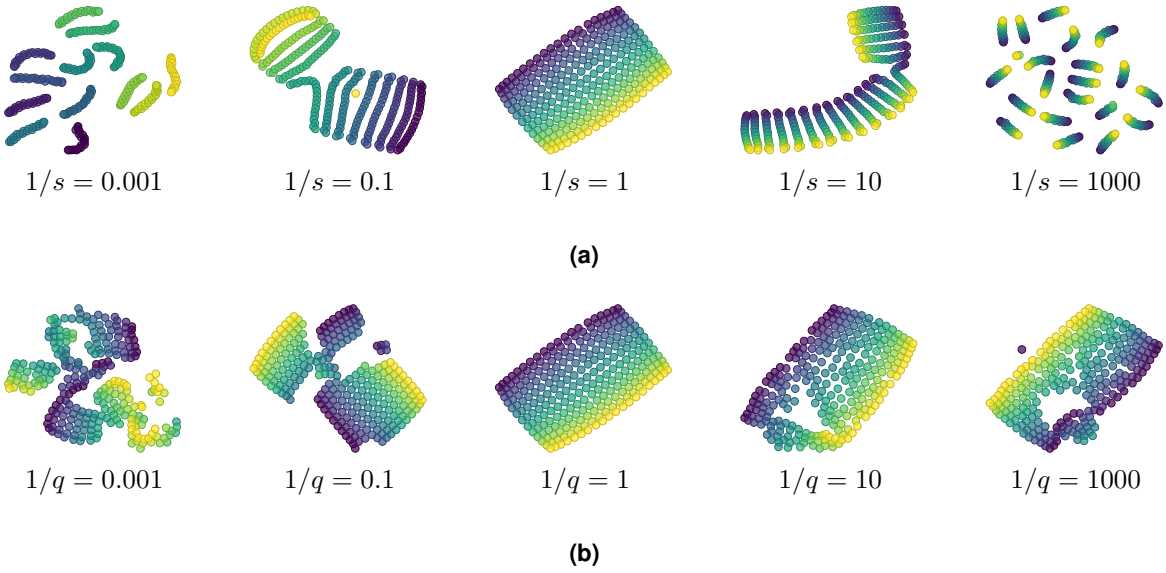


Figure 5. Comparison of *Het-node2vec* and node2vec embeddings on a synthetic heterogeneous graph having a grid topology. Node types are represented through different colors, and nodes in the same row belong to the same type. (a) *Het-node2vec* embeddings of the grid for different node type switching values s and $p = q = 1$. (b) node2vec embeddings of the grid when different values of the explore parameter q are used ($p = s = 1$).

Analysis of the special node-type switch in *Het-node2vec* on over- and under-represented node types in the Cora benchmark graph

The predictive performance of node embeddings depends on several factors, among which the prevalence of different node types in the network plays a crucial role. Rare node types are less frequently visited by random walks; consequently, the relationships between these nodes and the rest of the graph are often underrepresented in the learned embeddings.

The experiments reported in this section, as well as those conducted on real-world graphs (see next section), demonstrate that type-aware and prevalence-aware transition strategies, such as the special node-type switch implemented in *Het-node2vec*, can improve the quality of the learned representations, particularly for under-represented node types.

Our first experimentation was conducted using the Cora dataset. Cora is a citation network consisting of scientific publications used as a standard benchmark dataset for semi-supervised node classification tasks. The graph consists of 2,708 nodes representing scientific publications, connected by 5,278 edges representing citation relations between (node) paper. Each node is assigned one of seven classes describing its main academic research topic. Cora exhibits real-world complex networks properties; such as varying degrees of node connectivity and class imbalance, where some types are significantly more frequent than others (e.g., “Rule Learning” is the least frequent class, comprising 6.65% of the nodes, whereas “Neural Networks” is the most frequent, comprising 30.21%). Moreover, because it maps actual citation patterns, the graph exhibits “homophily", the tendency of nodes to connect to other nodes with similar labels. This makes it an ideal environment to explore type biases effect in the representation of real graphs. Type distributions for nodes ans edges are reported in Section 5 of the Supplementary Material. In this experiment, we use the paper-class labels as node types to study the behavior of the special node-type switching strategy under controlled type imbalance. The purpose of this analysis is not to use Cora as a general heterogeneous-graph benchmark, but to isolate how biasing RWs toward a selected node type affects the representation of over- and under-represented node categories.

We perform intra-type edge prediction tasks, where each task predicts citation edges whose endpoints belong to the same paper class, using a five-fold cross-validation setting with an 85:15 train:test split performed on edges. For each class-specific task, positive examples are citation edges connecting two papers of the selected class, while negative examples are sampled from non-existing edges between papers of the same class. This design allows us to evaluate each node type separately: each paper class is selected in turn as the special node type, and we assess whether biasing the walks toward nodes of that type improves the prediction of citation edges connecting papers within that class.

Node-embeddings were generated by processing RWs with a Skip-gram model. Default GRAPE parameters were used for

both RW generation and Skip-gram (Supplementary Table 2 details all of them). The values of $p = 4$ and $q = 0.25$ were fixed based on preliminary experiments detailed in Supplementary section 9. A Random Forest classifier was trained on the resulting node representations using the parameters described in the Supplementary Table 2 but with a Hadamard edge representation and trees of maximum depth equal to ten. Figure 6 summarizes the achieved performance, where setting $1/s = 1$ corresponds to the neutral no-switching case and therefore recovers the homogeneous node2vec baseline, while $1/s = 0.1$ and $1/s = 2$ respectively demote or promote transitions toward the selected special node type.

As shown in Figure 6a, increasing the bias toward the least frequent node types tends to improve ROC-AUC for those types, with the largest gains observed for the rarest classes. However, the trend is not strictly monotonic across all classes, indicating that node-type prevalence alone does not fully determine the effect of the switching parameter. Local density, degree distribution, and homophily/heterophily also affect how often target-type neighborhoods are reached by the RWs.

In contrast, for well-represented node types, a strong positive bias ($1/s = 2$) may degrade performance, whereas reducing their visitation probability ($1/s = 0.1$) tends to yield improvements.

To further illustrate this behavior, Figure 6b depicts the relative improvement over the baseline (expressed as a percentage gain) in terms of ROC-AUC when $1/s = 2$. We report both the total improvement and the gap improvement with respect to the perfect score; the latter is computed as the relative reduction in distance to the ideal value (i.e., ROC-AUC = 1.0).

Overall, the largest improvements are observed for the least frequent classes, suggesting that special node-type switching can help mitigate the sampling bias of standard RWs for under-represented node types. This supports the use of type-aware transition strategies when the downstream task focuses on categories that are poorly represented in the sampled random-walk contexts. However, as also indicated by the probability-ratio interpretation and the grid-topology analysis, this effect should not be interpreted as depending only on node-type prevalence: local edge density, degree distribution, and homophily/heterophily also influence the realized benefit of the switching strategy. Consequently, the choice of the most appropriate value of s should not be based solely on node-type prevalence.

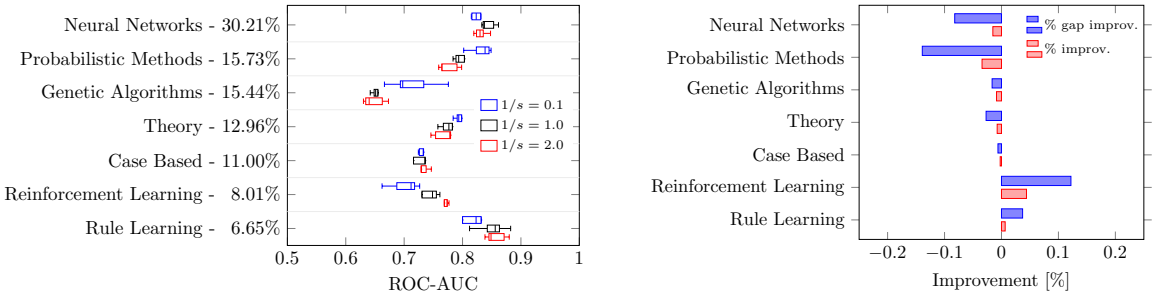


Figure 6. Edge prediction performance on the Cora dataset. We compare the standard node2vec random-walk (RW) baseline with specialized transition strategies that bias walk transitions toward the endpoint node types of the target edge type. In both panels, node types are ordered by prevalence in Cora, from least frequent (bottom: Rule Learning) to most frequent (top: Neural Networks). (a) ROC-AUC scores comparing node2vec and *Het-node2vec* (node-type switching) on intra-type edge prediction tasks (each task predicts edges whose endpoints share the same node type); node-type prevalence is reported in the y-axis labels. (b) Percentage improvement of the specialized switching strategy with $1/s = 2$ over node2vec, showing larger gains for underrepresented node types.

Analysis of *Het-node2vec* performance on real benchmark data sets

In this section, we evaluate *Het-node2vec* on real-world benchmark datasets and study how different switching strategies and parameters affect node-label prediction performance. To ensure a fair and robust comparison, we adopt the benchmark graphs and node-label prediction protocol from the Heterogeneous Network Embedding (HNE) evaluation framework[16] (more details are provided in the following paragraphs and in Supplementary Sections 7 and 8), which enables both a sensitivity analysis of switching configurations (subsection "Evaluation of *Het-node2vec* node-type switching on real-graphs") and an unbiased comparison with state-of-the-art HGRL methods (subsection "Comparison of *Het-node2vec* with state-of-the-art HGRL methods").

Datasets/Graphs. The four heterogeneous benchmark graphs proposed in HNE[16] are Freebase, DBLP, Yelp, and PubMed.

The differences among these benchmark graphs extend beyond node and edge cardinality and include node-type/and edge-type distribution, node-degree distribution, and graph connectivity. Consequently, we believe that evaluating *Het-node2vec* on these datasets allows us to assess its robustness and generalization capabilities across different application scenarios.

Table 1 summarizes the macroscopic features of the graphs. As can be observed, the graph dimensions and the number of different node/and edge types span a large range of values. In Supplementary Section 7, we provide additional details about the characteristics of the four graphs, including the node-type distribution and basic degree statistics (Supplementary Table 1), the edge-type distribution (Supplementary Figure 9), and the complementary cumulative distribution function (CCDF) of node degrees (Supplementary Figure 10).

Dataset	# nodes	# edges	mean degree	# types		
				nodes	edges	labels
Freebase	12,164,758	62,982,566	10.35	8	36	8
DBLP	1,989,077	258,850,593	277.46	4	6	13
Yelp	82,465	30,542,675	740.74	4	4	16
PubMed	63,109	244,986	7.76	4	10	8

Table 1. Macroscopic description of the four heterogeneous network datasets.

Experimental settings. Following the experimental setting proposed in HNE[16], we applied *Het-node2vec* in an unsupervised and unattributed setting to assess the quality of the computed representations. Following the evaluation protocol provided by HNE, and using the training and test splits provided by the authors, we embedded each of the four benchmark graphs and used the computed embeddings to perform node-label prediction on specific node types within each graph (the node type under consideration and the labels are provided by the HNE framework). Specifically, we predict labels for `book` nodes in Freebase, `author` nodes in DBLP, `business` nodes in Yelp, and `disease` nodes in PubMed. Further details about the characteristics of the prediction tasks considered in the experiments are available in Supplementary Section 8.

Evaluation workflow. The workflow of the evaluation pipeline is depicted in Figure 7. The process starts with the removal of 20% of existing edges at random to obtain an independent test set for unbiased evaluation. The remaining dataset is used to compute a vectorial representation of the graph nodes according to the different embedding methods.

According to the experimental setting used in HNE[16], the embedding is computed in an unsupervised setting, with the objective of minimizing a loss function that preserves the graph-topological structure. When used under this specific setting, GNNs are trained to maximize the likelihood of observing edges in the heterogeneous network. To this end, they use negative sampling and minimize the cross-entropy loss (see the code provided by the authors of[16]).

The embeddings obtained from the previous step serve as the resulting representations for the subsequent predictive tasks. The HNE framework[16] fixes the embedding size to 50, while other crucial parameters are optimized on all datasets through standard five-fold cross-validation on the training graph[5]. To guarantee a fair and repeatable comparison, we perform the predictive tasks using the training/test splits provided by the HNE framework as well as their implementation of the evaluation pipeline.

Node-label classification is performed by training and evaluating a linear support vector machine[35] on a subset of labeled nodes using five stratified holdouts (80:20 train:test ratio). The predictive performance measures are the macro-F1 (across all labels) and micro-F1 (across all nodes), averaged across the five holdouts.

Let TP_c, FP_c, and FN_c denote, respectively, the number of true positives, false positives, and false negatives for class c, and let C be the set of classes. For each class c, precision, P_c, recall, R_c, and F1, $F1_c$, are computed and averaged:

$$\text{Macro-}F1 = \frac{1}{|C|}\sum_{c\in C} F1_c \qquad \text{where} \qquad P_c = \frac{TP_c}{TP_c+FP_c}, \qquad R_c = \frac{TP_c}{TP_c+FN_c}, \qquad F1_c = \frac{2P_cR_c}{P_c+R_c}.$$

The micro-F1 score is computed by first aggregating true positives, false positives, and false negatives over all classes and then applying the harmonic mean:

$$\text{Micro-}F1 = \frac{2P_{\text{micro}}R_{\text{micro}}}{P_{\text{micro}}+R_{\text{micro}}} \qquad \text{where:} \qquad P_{\text{micro}} = \frac{\sum_{c\in C}TP_c}{\sum_{c\in C}(TP_c+FP_c)}, \qquad R_{\text{micro}} = \frac{\sum_{c\in C}TP_c}{\sum_{c\in C}(TP_c+FN_c)}.$$

Practically, Macro-F1 gives equal weight to each class, independently of class prevalence, and therefore makes poor performance on minority classes more visible. Micro-F1 aggregates true positives, false positives, and false negatives over all classes and is therefore more influenced by the performance on majority classes.

[5]To apply the benchmarked algorithms, Yang et al.[16] adapted the code provided by the authors of the algorithms themselves. The parameters that are optimized therefore depend on the code itself, as chosen by the authors of the code.

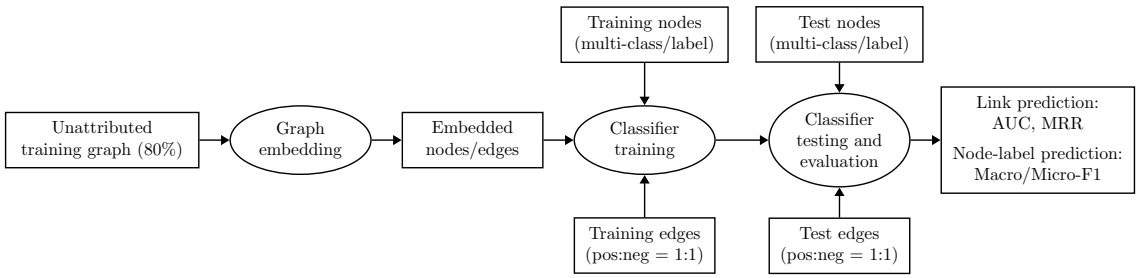


Figure 7. HNE experimental evaluation workflow.

Evaluation of *Het-node2vec* node-type switching on real-graphs. To isolate the contribution of *Het-node2vec* node-type switching and assess whether the use of node-type information could impact the quality of the obtained graph representation, we compared *Het-node2vec* against its homogeneous counterpart, node2vec, while keeping the node2vec hyperparameters p and q fixed so that both methods employed the same second-order random-walk bias (BFS-like vs. DFS-like) across all graphs. We selected p and q via preliminary experiments on the two smallest benchmarks (PubMed and Yelp, more details in Supplementary section 9) using an internal five-fold cross-validation, sweeping values in $\{0.1, 0.25, 0.5, 4, 10, 100\}$; the most robust performance was obtained with $1/p = 0.25$ and $1/q = 4$ (i.e., $p = 4$ and $q = 0.25$). These values were kept fixed in all subsequent experiments, biasing random walks toward a DFS-like exploration that tends to reach new regions of the graph.

The remaining hyperparameters were selected to ensure comparability with the HNE benchmark and to isolate the effect of the proposed type-switching mechanism. The embedding dimension was fixed to 50, following the HNE evaluation protocol.

For each graph, the number of RWs starting from each node was chosen between 10 and 50, based on internal three-fold cross-validation. These values were identified by balancing computational time against the representativeness of the sampled node contexts. A high number of RWs yields embeddings that better represent the topological relationships in the graph but increases computational time, while a low number of RWs ensures faster computation at the expense of embedding accuracy. The walk length was fixed to $L = 100$ across the benchmark experiments.

We used a Skip-gram model as the shallow neural network for embedding, with the network hyperparameter values set to their default values according to the implementation provided by GRAPE (see Supplementary Table 2 in Supplementary Section 9). This procedure produced the vector representations of the network nodes.

Using this setting, we empirically evaluated the results obtained from *Het-node2vec* using either the generic node-type-switching strategy or the special node-type-switching strategy, where the special node type was set as the node targeted by the node-label prediction task (see Supplementary Table 1), and we varied the switching parameter $1/s$ in the range $[10^{-1}, 10^{2}]$.

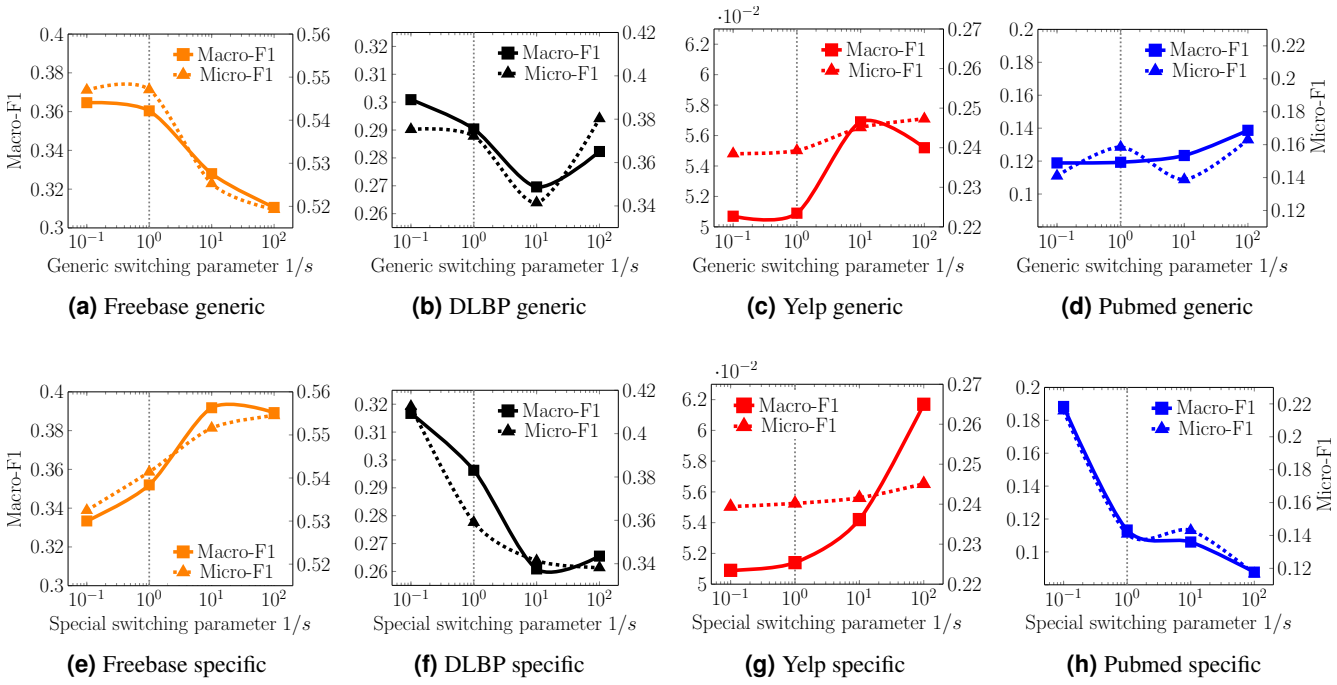
Figure 8 demonstrates how varying the value of the switching parameter s affects the Macro- and Micro-F1 scores achieved. This effect is observed when both the generic (Figure 8a–d) and the special (Figure 8e–h) node-type switching strategies are employed. We recall that by setting $1/s = 10^{0}$ (marked with a dotted vertical line in each plot, *Het-node2vec* reduces to the classical node2vec algorithm.

With the generic switching strategy (Figure 8a–d), Freebase shows a clear decrease in performance as the switching probability $1/s$ increases, whereas DBLP shows a more variable pattern with an overall tendency toward lower performance at larger $1/s$ values in the considered grid. In contrast, Yelp and PubMed display the opposite trend. The special switching strategy behaves differently, showing a positive correlation between $1/s$ and the F1 scores for Freebase and Yelp, and a negative correlation for DBLP and PubMed (Figure 8 e–h).

With the generic switching strategy (Figure 8, first row), Freebase shows a clear decrease in performance as the switching probability $1/s$ increases, whereas DBLP shows a more variable pattern, with lower performance for most larger values of $1/s$ in the considered grid. In contrast, Yelp and PubMed display the opposite tendency, with higher F1 scores for larger values of $1/s$. The special switching strategy behaves differently, showing higher F1 scores for larger values of $1/s$ in Freebase and Yelp, and lower F1 scores for larger values of $1/s$ in DBLP and PubMed (Figure 8 e–h).

Note that these observations should be interpreted as a coarse sensitivity analysis of the switching parameter over the tested grid, rather than as evidence of strictly monotonic trends.

This behavior can be explained by the fact that in both Freebase and DBLP, labeled nodes (of type `book` in Freebase and of type `author` in DBLP) have lower degrees compared to other node types (see Supplementary Figures 11a and 11b). In these cases, generic node-type switching could cause the RW to move away from the nodes of interest, thus failing to adequately capture their topological neighborhoods (as shown by the decreasing trends in Figures 8a and 8b). This is particularly true in Freebase, where only approximately 23% of nodes are of type `book` (Supplementary Table 1); using the special node-type switching strategy to bias the walk in favor of `book` nodes (i.e., using high values of β_s, see Figure 8e) yields better results, as



(a) Freebase generic **(b)** DLBP generic **(c)** Yelp generic **(d)** Pubmed generic

(e) Freebase specific **(f)** DLBP specific **(g)** Yelp specific **(h)** Pubmed specific

Figure 8. Node-label prediction: Performance metrics showing macro-F1 (left axis - continuous line) and micro-F1 (right-axis - dashed line) across varying β_s parameter values. The first row (subplots a,b,c,d) shows the performances values using the generic switching strategy, while the second column (subplots e,f,g,h) shows the results for the special switching strategy. The scales for macro-F1 and micro-F1 vary between datasets but are consistent for the same metric across the generic and special *Het-node2vec* strategies to enable direct comparison.

indicated by the increasing trend. Conversely, in DBLP, `author` nodes, which account for approximately 89% of all nodes, are the most prevalent node type. Here, using the special node-type switching strategy with high values of β_s would confine the walk to `author` nodes, neglecting other node types and resulting in a loss of information in the computed embedding. This is reflected in Figure 8f, where performance is generally lower for larger values of $1/s$ in the tested grid, although the pattern should not be interpreted as strictly monotonic.

In the Yelp graph (Figures 8c and 8g), the node-label prediction task focuses on `business` nodes, which are much less represented (approximately 9% of all nodes) compared to `phrase` nodes, which dominate the graph, constituting approximately 91% of all nodes (Supplementary Table 1). In this scenario, using a generic type-switching strategy that enforces heterogeneity (high values of β_s) has roughly the same positive effect as a special node-type switching strategy where the special node type is `business`; in both cases, high values of β_s promote visits to business nodes. Both settings yield better performance for high values of $1/s$ (Figures 8c and 8g) because they compel the walk to move away from phrase nodes and capture as much heterogeneous information as possible.

PubMed is characterized by the most balanced node-type distribution and edge-degree distribution (Supplementary Table 1). In other words, with the exception of `species` nodes, the types of nodes in the PubMed network are more evenly distributed than in the other benchmark graphs, with no overwhelming majority of any node type. Additionally, the degree distributions are nearly identical for all node types (see Supplementary Figure 11d). In this case, favoring the heterogeneity of the walk has only a minor positive effect on performance (slightly increasing trend in Figure 8d), and using a special node-type switch that forces the walk toward `disease` nodes (the nodes used in the node-label prediction task) leads to walks that focus predominantly on `disease` nodes (Figure 8h).

The results presented in this section highlight that the relative distribution of node types and their degrees must be carefully considered when setting the switching parameter. However, as already discussed for the Cora experiment, these patterns should not be attributed to node-type prevalence alone: local edge density, node degree distribution, edge-type distribution, and homophily/heterophily also influence how the switching parameter changes the sampled RW contexts. Moreover, careful tuning of this parameter can improve the prediction performance of classical homogeneous RW-based embedding techniques by allowing the model to focus on node types of interest, which are often underrepresented.

Comparison of *Het-node2vec* with state-of-the-art HGRL methods. To assess the effectiveness of *Het-node2vec* representations, we further compared the results obtained in the node-label prediction task with various HGRL methods. Similar to the previous section, we applied the HNE evaluation framework to compare the performance of the node-label prediction task against:

- five RW-based embedding methods (metapath2vec[36], PTE[37], Aspem[38], HIN2Vec[39], HEER[40]);
- four GCN-based embedding methods (R-GCN[41], HAN[42], HGT[43], and MAGNN[44]);
- four relation-learning neural methods (TransE[25], DistMult[26], ConvE[45], and ComplEx[28]).

The results depicted in Table 2 demonstrate that *Het-node2vec* is highly competitive in node-label prediction across all graphs. These results indicate that the proposed heterogeneous RW approach effectively captures the underlying structure while incorporating node- and edge-type information. In the case of DBLP, *Het-node2vec* is not the best-performing method, but it ranks among the top four. Nevertheless, it is worth noting that the target node type in DBLP (`author`) accounts for more than 88% of the graph nodes (Supplementary Table 1), thus exhibiting a predominantly homogeneous character. Therefore, it is not surprising that heterogeneous sampling techniques cannot fully exploit the available node- and edge-type information information.

We emphasize that the special node-type switching strategy achieves the best results, while the generic type-switching strategy also achieves, on average, better results than the other competing HGRL methods. By tuning the $1/s$ parameter, both strategies lead to results that are better than or comparable to state-of-the-art competing methods.

		Node label prediction (Macro-F1/Micro-F1)							
Model	Parameter	Freebase		DBLP		Yelp		Pubmed	
HetNode2vec generic	$1/s = 0.1$	36.46	54.70	30.09	37.53	5.07	23.85	11.89	14.10
	$1/s = 100$	31.06	51.93	28.23	38.03	5.52	**24.73**	13.87	16.10
HetNode2vec special	$1/s = 0.1$	33.34	53.25	31.67	41.26	5.09	23.96	**18.84**	**21.58**
	$1/s = 100$	**38.93**	**55.45**	26.55	33.82	**6.11**	**24.73**	8.77	11.67
node2vec		36.04	54.71	29.04	37.22	5.09	23.96	11.93	15.85
metapath2vec		20.55	46.43	**43.85**	**55.07**	5.16	23.32	12.90	15.51
PTE		10.25	39.87	43.34	54.53	5.10	23.24	9.74	12.27
HIN2Vec		17.40	41.92	12.17	25.88	5.12	23.25	10.93	15.31
AspEm		23.26	45.42	33.07	43.85	5.40	23.82	11.19	14.44
HEER		12.96	37.51	9.72	27.72	5.03	22.92	11.73	15.29
R-GCN		6.89	38.02	9.38	13.39	5.10	23.24	10.75	12.73
HAN		6.90	38.01	7.91	16.98	5.10	23.24	9.54	12.18
MAGNN		6.89	38.02	6.74	10.35	5.10	23.24	10.30	12.60
HGT		23.06	46.51	15.17	32.05	5.07	23.12	11.24	18.72
TransE		31.83	52.04	22.76	37.18	5.05	23.03	11.40	15.16
DistMult		23.82	45.50	11.42	25.07	5.04	23.00	11.27	15.79
ComplEx		35.26	52.03	20.48	37.34	5.05	23.03	9.84	18.51
ConvE		24.57	47.61	12.42	26.42	5.09	23.02	13.00	14.49

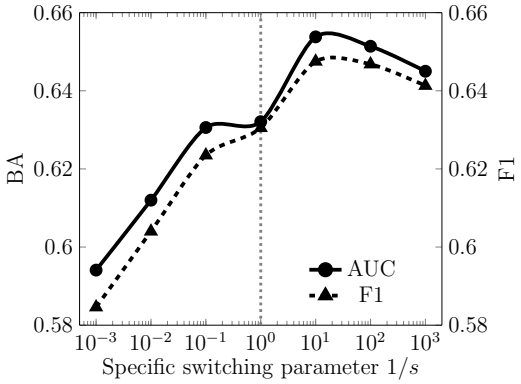
Table 2. Performance metric for the node-label prediction task on the benchmark graphs. Het-node2vec-*special* refers to the special node-type switching strategy, where the special node type is the one targeted by the node-label prediction task. In the table, for both *Het-node2vec* settings, we report only the values obtained using the two extreme values of the node-type switching parameter $1/s = 0.1, 100$.

In Supplementary Section 10, we provide the experimental results on the benchmark graphs for the link-prediction tasks provided by HNE.

In this task, *Het-node2vec* does not outperform the other methods but consistently ranks among the top five (Supplementary Table 4). These results indicate that *Het-node2vec* can still produce valuable embeddings for link prediction, even if its primary strength lies in node-label prediction.

Source	Destination	Count	Proportion
lncRNA	mRNA	2,425,740	0.3192
mRNA	miRNA	1,065,227	0.1402
Gene	**miRNA**	841,583	0.1107
circRNA	miRNA	416,884	0.0549
Chemical	Chemical	358,045	0.0471
Protein	lncRNA	200,568	0.0264
Protein	Pseudogene	190,416	0.0251
Protein	Protein	165,300	0.0218
miRNA	tsRNA	119,965	0.0158
tRF	tRNA	114,804	0.0151

(a)



(b)

Figure 9. (a) RNA-KG node/edge type distribution. (b) Edge prediction performance metric for the prediction of Gene-miRNA interactions.

Nevertheless, prediction performance also strongly depends on the supervised method applied to the computed embeddings. Indeed, by repeating the same experiments using a nonlinear classifier (a random forest), we significantly improved the edge-prediction results obtained with *Het-node2vec* embeddings (Supplementary Table 5).

A case study: RNA-KG Knowledge Graph

We conclude the experimental section by using *Het-node2vec* to conduct a link prediction task on a knowledge graph representing relationships between biomedical entities. The dataset, which we refer to as RNA-KG[15], integrates coding and non-coding RNA data from over 60 public databases and contains interactions between RNA molecules and various biomolecular entities, including chemicals, diseases, abnormal phenotypes, and proteins.

The RNA-KG graph contains 673825 nodes and nearly 17 million edges, with a mean degree of 25.23. Graph nodes belong to 74 different node types, and their distribution is depicted in Supplementary Section 11. This biological knowledge graph provides an effective testing bed for type-aware sampling strategies due to its heterogeneous structure and the critical importance of relationship types in biological systems.

In the RNA-KG graph, we perform an edge prediction task for a specific pair of node types: `miRNA-Gene`. The nodes of RNA-KG represent heterogeneous biomedical entities, including RNA molecules, genes, proteins, chemicals, diseases, phenotypes, and other biological concepts. Edges represent curated biological associations or interactions between these entities, such as molecular interactions, regulatory relationships, or associations extracted from the integrated RNA-related resources. The downstream task considered here is a binary edge-prediction problem: given the embeddings of nodes of type miRNA and Gene, we train a classifier to distinguish existing miRNA–Gene interactions from sampled non-existing pairs.

Gene and miRNA node types account for the 3.086% and 0.688% f the nodes, respectively, and this relation accounts for 841,583 edges, representing 11.07% of the total edges in RNA-KG after `lncRNA-mRNA` (2425740, 31.92%) and `mRNA-miRNA` (1065227, 14.02). Figure 9a shows the edge types, defined by their endpoint node types, accounting for 77.6% of the graph edges. MicroRNAs (miRNA's) are small regulatory RNA molecules that bind to messenger RNAs (mRNAs) from genes, regulating their expression. One miRNA can target multiple genes, and one gene can be targeted by multiple miRNAs, creating a regulatory network that fine-tunes gene expression.

The predictive task is performed following a workflow similar to the one described in the "Experimental Settings" paragraph (see Figure 7). The RNA-KG graph was partitioned into a 70:30 training/test split, where the partition ensures connectivity of the training set. RWs were generated according to a special edge switching strategy similar to strategy $\hat{\beta}_s^{(1)}$ described in the "Implementation" Section and Figure 2a for node type switching. Specifically, edges incident to a node of type either `miRNA` or `Gene` are weighted by $1/s$, while edges connecting nodes of type `Gene` and `miRNA` are weighted by $1/s^2$. From each node in the training set, we generate 128 random walks of length 16, which we use to create a 25-dimensional embedding of graph nodes. Edge embeddings are generated as the concatenation of the embedding of their endpoints. This embedding is used to train a Random Forest model to predict edges of type `miRNA-Gene` using a balanced positive:negative dataset. Hyperparameter values used in *Het-node2vec* and the RF training are described in the Discussion Section. Figure 9b depicts the Balanced Accuracy (BA) and F1 values for the edge prediction task on the targeted edge types (`miRNA-Gene`) for different values of the edge-type switching weight (s). As can be seen, the best results are obtained with $1/s = 10$, i.e by promoting the visit of the target edge type; this choice obtains a significant gain of the 3.7% with respect to node2vec (Wilcoxon signed

rank test, $alpha = 0,001$). Moreover, other preliminary experiments we run on this RNA-KG graph (reported in[46]) show that *Het-node2vec* significantly improves `miRNA-gene` interaction predictions results obtained through first-order LINE embeddings, achieving 66% balanced accuracy compared to 58%. This improvement demonstrates that a proper choice of switching weights can leverage the node- and edge-type information, ultimately enhancing the graph representation.

Discussion and Conclusions

The results obtained through the proposed *Het-node2vec* algorithm indicate the advantages of embedding techniques that account for node and edge heterogeneity in graphs. Methods like node2vec, originally conceived for homogeneous graphs may fail to capture the diverse types of nodes and relationships, often leading to sub-optimal performance in real-world applications. On the other hand, many state-of-the-art HGRL methods depend on the specific characteristics of the input graph, and/or rely on large, non-scalable neural network architectures, or are trained to produce embeddings that are tailored to a specific predictive task. By introducing type-aware second-order RWs, *Het-node2vec* provides a flexible and scalable alternative that captures the topological graph structure and and incorporates node- and edge-type information into the sampled contexts.

This advantage should be interpreted as a tradeoff between scalability, simplicity, and expressiveness. Attention-based heterogeneous GNNs such as HAN and MAGNN are indeed generally more expressive than *Het-node2vec* because they can learn task-specific node-feature transformations, attention weights, and meta-path-specific or relation-aware aggregation functions. This expressiveness, however, comes at the cost of increased complexity and does not necessarily lead to better performance in unattributed, task-independent embedding settings such as the one considered in our benchmark experiments (Table 2); nevertheless, such mechanisms can be particularly beneficial when rich node or edge attributes are available, when informative meta-paths can be defined, or when the target task requires hierarchical or complex relational reasoning. By contrast, *Het-node2vec* does not learn type-specific aggregation functions and does not directly exploit node attributes or richer relation attributes beyond node and edge types. Its strength lies instead in providing an unsupervised, scalable, and type-aware structural embedding method that does not require manual meta-path engineering and can be applied to large heterogeneous graphs even when node attributes are absent, incomplete, or difficult to harmonize across node types. Therefore, *Het-node2vec* should be regarded as complementary to attention-based heterogeneous GNNs: it favors automation, scalability, and reusable structural embeddings, whereas methods such as HAN and MAGNN favor higher task-specific expressiveness when the required features, meta-paths, and computational resources are available. Our experiments on multiple benchmark datasets demonstrate that our model enables the design of simple strategies to balance the exploration of graph heterogeneity. For example, datasets like Freebase and PubMed benefit from increasing the heterogeneity in the RW, promoting the representation of the relationships between different node types. Conversely, for datasets where nodes are more evenly distributed, such as Yelp, promoting homogeneity in the walks can improve performance by focusing the walk on specific target node types. This can also be beneficial when we focus on underrepresented nodes or edges in the underlying heterogeneous graph. Indeed, our results suggest that, in cases where certain node types are relatively rare, focusing on special node types can yield significant improvements in the resulting representation. On the other hand, excessive focus on these special nodes can result in overly homogeneous embeddings that neglect the broader network structure, thereby limiting the representation ability of the methods.

We further show that a proper experimental tuning the s parameter can significantly improve results with respect to homogeneous methods. Moreover, extensive experimental results outline that *Het-node2vec* is competitive with state-of-the-art methods for heterogeneous graphs and exhibits a temporal complexity of the same order of its homogeneous counterpart node2vec, which allows for scalable applications with large heterogeneous networks.

Overall, our experiments show that the choice of the switching parameter s should be guided by both the target task and the structure of the heterogeneous graph. In particular, node- or edge-type prevalence alone does not fully determine the effect of s: local density, degree distribution, and homophily/heterophily patterns also influence the realized transition frequencies induced by a given global value of s. Thus, the same value of s may have different effects in different regions of a heterogeneous graph. From a practical perspective, a useful strategy is to first tune or select the standard node2vec parameters p and q using the homogeneous baseline, and then tune s on a logarithmic grid around the neutral value $1/s = 1$.

Regarding the choice of the generic versus special switching strategy, the generic switching strategy is appropriate when the goal is to globally promote or demote transitions across node types and to obtain reusable, task-independent type-aware embeddings. The special switching strategy is instead preferable when the downstream task focuses on known target node types or edge endpoint types, especially when these types are under-represented or under-sampled by standard RWs; in this case, the embeddings are partially task-oriented because the sampling process is biased toward the (under-represented) types involved in the target task. Conversely, when the target type is already dominant in the graph, the special switching parameter should be tuned carefully; in such cases, lower values of $1/s$ may be preferable because they demote transitions toward that type and encourage the RWs to explore more heterogeneous neighborhoods.

For the generic switching strategy, very large or very small values of $1/s$ can similarly distort the sampled contexts by excessively promoting or suppressing transitions across node types.

The proposed switching functions are fixed and user-controlled rather than learned end-to-end. This design makes the effect of s directly interpretable from the transition probabilities, since it explicitly promotes or demotes transitions involving selected node types. Therefore, s should not be interpreted as a universal parameter, but as a task- and graph-dependent control of local type-aware sampling.

Summarizing, results show that the proposed method can match or outperform alternative approaches in several settings; however, its representation potential is far from being completely exploited and should be further explored in future work. Indeed, while the synthetic experiments, the Cora analysis, the HNE benchmark graphs, and the RNA-KG case study consistently show that type-aware switching can affect the quality of the learned embeddings, they also indicate that node-type prevalence, local density, degree distribution, edge-type distribution, and homophily/heterophily are intertwined in real heterogeneous graphs. This makes the choice of appropriate values for p, q, and s more complex. A limitation of the present experimental analysis is that it does not fully disentangle the individual contribution of all these factors to the observed effect of the switching parameter. A dedicated controlled study based on synthetic graphs or systematic subsampling would be required to isolate their individual effects. Besides this more detailed analysis of the relationship between node and edge type distributions and model parametrization, future work will focus on the design of switching techniques to improve edge prediction in a heterogeneous graph framework.

A final important note is that, in the present *Het-node2vec* formulation, edge-type information is mainly considered through the types of the source and destination nodes involved in an edge. Nevertheless, node- and edge-level representations may still differ in relational graphs where relations with different semantics connect the same pair of node types. The edge-prediction experiments on the DBLP graph, reported in the Supplementary Material, indicate that *Het-node2vec* can still capture differences between such relationships because embeddings are learned from random-walk neighborhoods and therefore reflect the broader structural and type-aware context of each endpoint. Proper tuning of the switching parameter can support this task: the generic switching parameter can be tuned to collect more heterogeneous walks, while the special switching parameter can bias the walks toward rare node types that may provide crucial contextual information. Although this already provides a preliminary way to distinguish relations with the same endpoint node types, future work will investigate explicit relation-subtype-aware switching functions, in which different edge labels connecting the same node-type pair can directly influence the transition probabilities.

References

1. Zhang, D., Yin, J., Zhu, X. & Zhang, C. Network representation learning: A survey. *IEEE Transactions on Big Data* **6**, 3–28, DOI: 10.1109/TBDATA.2018.2850013 (2020).

2. Grover, A. & Leskovec, J. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 855–864, DOI: 10.1145/2939672.2939754 (Association for Computing Machinery, New York, NY, USA, 2016).

3. Zhang, D., Yin, J., Zhu, X. & Zhang, C. Homophily, structure, and content augmented network representation learning. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 609–618, DOI: 10.1109/ICDM.2016.0072 (2016).

4. Martinez, V., Berzal, F. & Cubero, J. A survey of link prediction in complex networks. *ACM Comput. Surv.* **49**, DOI: 10.1145/3012704 (2017).

5. Wang, X. *et al.* Community preserving network embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 203–209 (AAAI Press, 2017).

6. Wang, P., Zhang, J., Liu, G., Fu, Y. & Aggarwal, C. Ensemble-spotting: Ranking urban vibrancy via poi embedding with multi-view spatial graphs. In *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM)*, 351–359, DOI: 10.1137/1.9781611975321.40 (2018). https://epubs.siam.org/doi/pdf/10.1137/1.9781611975321.40.

7. Dong, Y., Hu, Z., Wang, K., Sun, Y. & Tang, J. Heterogeneous network representation learning. In Bessiere, C. (ed.) *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 4861–4867, DOI: 10.24963/ijcai.2020/677 (International Joint Conferences on Artificial Intelligence Organization, 2020). Survey track.

8. Natarajan, N. & Dhillon, I. S. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics* **30**, i60–i68, DOI: 10.1093/bioinformatics/btu269 (2014). https://academic.oup.com/bioinformatics/article-pdf/30/12/i60/1090926/btu269.pdf.

9. Perozzi, B., Al-Rfou, R. & Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, 701–710, DOI: 10.1145/2623330.2623732 (Association for Computing Machinery, New York, NY, USA, 2014).

10. Tang, J. *et al.* Line: Large-scale information network embedding. In *WWW '15: Proceedings of the 24th International Conference on World Wide Web*, WWW '15, 1067–1077, DOI: 10.1145/2736277.2741093 (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2015).

11. Wang, H. *et al.* Learning graph representation with generative adversarial nets. *IEEE Transactions on Knowl. Data Eng.* **33**, 3090–3103, DOI: 10.1109/TKDE.2019.2961882 (2021).

12. Cao, S., Lu, W. & Xu, Q. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 1145–1152 (AAAI Press, 2016).

13. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 1025–1035 (Curran Associates Inc., Red Hook, NY, USA, 2017).

14. Dai, Y., Wang, S., Xiong, N. N. & Guo, W. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics* **9**, DOI: 10.3390/electronics9050750 (2020).

15. Cavalleri, E. *et al.* An ontology-based knowledge graph for representing interactions involving RNA molecules. *Sci. Data* **11**, DOI: 10.1038/s41597-024-03673-7 (2024).

16. Yang, C., Xiao, Y., Zhang, Y., Sun, Y. & Han, J. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowl. Data Eng.* **34**, 4854–4873 (2020).

17. Xie, Y. *et al.* A survey on heterogeneous network representation learning. *Pattern Recognit.* **116**, DOI: https://doi.org/10.1016/j.patcog.2021.107936 (2021).

18. Bing, R. *et al.* Heterogeneous graph neural networks analysis: a survey of techniques, evaluations and applications. *Artif. Intell. Rev.* **56**, 8003–8042 (2023).

19. Sefer, E. Comparison of biological graph alignment algorithms with hyperbolic heterophilic deep graph learning-based approach. *IEEE Transactions on Mol. Biol. Multi-Scale Commun.* (2026).

20. Gezici, A. H. B. & Sefer, E. Pagerank-based unsupervised deep vertex representations for anti-money laundering detection. *IEEE Access* **13**, 196935–196950 (2025).

21. Fries, C. C. Meaning and linguistic analysis. *Language* **30**, 57–68 (1954).

22. Harris, Z. S. Distributional structure. *Word* **10**, 146–162 (1954).

23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 3111–3119 (Curran Associates Inc., Red Hook, NY, USA, 2013).

24. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17 (2017).

25. Bordes, A., Weston, J., Collobert, R. & Bengio, Y. Learning structured embeddings of knowledge bases. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 25, 301–306 (AAAI Press, 2011).

26. Yang, B., tau Yih, W., He, X., Gao, J. & Deng, L. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations* (2014).

27. Wang, Z., Zhang, J., Feng, J. & Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 28, 1112–1119 (AAAI Press, 2014).

28. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É. & Bouchard, G. Complex embeddings for simple link prediction. In *International conference on machine learning*, 2071–2080 (PMLR, 2016).

29. Hu, J., Hooi, B. & He, B. Efficient heterogeneous graph learning via random projection. *IEEE Transactions on Knowl. Data Eng.* (2024).

30. Wang, Y. *et al.* Kg2vec: A node2vec-based vectorization model for knowledge graph. *Plos one* **16**, e0248552 (2021).

31. Cappelletti, L. *et al.* Grape for fast and scalable graph processing and random-walk-based embedding. *Nat. Comput. Sci.* **3**, 552–568, DOI: 10.1038/s43588-023-00465-8 (2023).

32. Lombardo, G. & Poggi, A. Actornode2vec: An actor-based solution for node embedding over large networks. *Intelligenza Artif.* **14**, 103–114, DOI: 10.3233/IA-190038 (2020).

33. Viger, F. & Latapy, M. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. *J. Complex Networks* **4**, 15–37, DOI: 10.1093/comnet/cnv013 (2015). https://academic.oup.com/comnet/article-pdf/4/1/15/6999929/cnv013.pdf.

34. Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research* **9**, 2579–2605 (2008).

35. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. & Lin, C.-J. Liblinear: A library for large linear classification. *J. machine Learn. research* **9**, 1871–1874 (2008).

36. Dong, Y., Chawla, N. V. & Swami, A. Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, 135–144, DOI: 10.1145/3097983.3098036 (Association for Computing Machinery, New York, NY, USA, 2017).

37. Tang, J., Qu, M. & Mei, Q. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, 1165–1174, DOI: 10.1145/2783258.2783307 (Association for Computing Machinery, New York, NY, USA, 2015).

38. Shi, Y., Gui, H., Zhu, Q., Kaplan, L. & Han, J. Aspem: Embedding learning by aspects in heterogeneous information networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, 144–152 (SIAM, 2018).

39. Fu, T.-y., Lee, W.-C. & Lei, Z. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1797–1806 (2017).

40. Shi, Y., Zhu, Q., Guo, F., Zhang, C. & Han, J. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2190–2199 (2018).

41. Schlichtkrull, M. *et al.* Modeling relational data with graph convolutional networks. In *The Semantic Web. ESWC 2018*, vol. 10843 of *Lecture Notes in Computer Science* (Springer, 2018).

42. Wang, X. *et al.* Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19, 2022–2032, DOI: 10.1145/3308558.3313562 (Association for Computing Machinery, New York, NY, USA, 2019).

43. Hu, Z., Dong, Y., Wang, K. & Sun, Y. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, WWW '20, DOI: 10.1145/3366423.3380027 (Association for Computing Machinery, New York, NY, USA, 2020).

44. Fu, X., Zhang, J., Meng, Z. & King, I. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, 2331–2341 (2020).

45. Dettmers, T., Minervini, P., Stenetorp, P. & Riedel, S. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32 (2018).

46. Torgano, F. *et al.* Rna knowledge-graph analysis through homogeneous embedding methods. *Bioinforma. Adv.* **5**, vbaf109, DOI: 10.1093/bioadv/vbaf109 (2025).

Funding

This paper is supported by FAIR (Future Artificial Intelligence Research) project, funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence), and by the National Center for Gene Therapy and Drugs Based on RNA Technology—MUR (Project no. CN_00000041) funded by NextGeneration EU program.

Part of this work was funded by the National Plan for NRRP Complementary Investments (PNC) in the call for the funding of research initiatives for technologies and innovative trajectories in the health—project n. PNC0000003—AdvaNced Technologies for Human-centrEd Medicine (project acronym: ANTHEM). CC acknowledges funding from grants PID2024-162155OB-I00 and PID2024-162141OB-I00 by MICIU/AEI/10.13039/501100011033/ERDF/EU.

Computational resources at CINECA for this work have been funded by UNITECH INDACO, which is an HPC project at the state University of Milan.

Additional information

Code and data availability *Het-node2vec* is implemented using the efficient RW-generation provided by the GRAPE library[31], which enables efficient first and second-order RW generation through the usage of efficient and succinct data structures and an optimized Rust implementation with Python binders. The *Het-node2vec* code is available at `https://github.com/AnacletoLAB/hetnode2vec_ensmallen`. The datasets and the benchmark pipeline used in the experiments follow the experimental set-up proposed in[16] and are available from `https://github.com/yangji9181/HNE`.

Competing interests The authors declare no competing interests

Author contributions statement

MSG (Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing—original draft, Writing—review & editing), CC and JR (Conceptualization, Investigation, Methodology, Writing—review), PNR (Conceptualization, Formal analysis, Investigation, Methodology, Writing—review), GV and EC (Conceptualization, Funding acquisition, Methodology, Supervision, Writing—original draft, Writing—review & editing). All authors reviewed the manuscript.