Het-node2vec: second order random walk sampling for heterogeneous graphs embedding: Supplementary Information

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

[1]AnacletoLab, Dipartimento di Informatica, Università degli Studi di Milano, Milan, Italy
[2]Department of Computer Science and Artifitial Intelligence, Univerisidad de Granada, Granada, Spain
[3]Lawrence Berkeley National Laboratory, Berkeley, USA
[4]Berlin Institute of Health - Charité, Universitätsmedizin, Berlin, Germany
[5]ELLIS - European Laboratory for Learning and Intelligent Systems, Milan unit, Italy
[*]mauricio.soto@unimi.it
[+]these authors contributed equally to this work

Contents

1 HGRL methods: background

The highly informative representation provided by graphs that include different types of entities and relationships motivates the development of increasingly complex and heterogeneous networks[1], also including Knowledge Graphs[2,3], integrating and representing the information carried by multiple data sources in the form of concepts (nodes of the graph, characterized by multiple types) interconnected by edges representing the existence of any relationship between them. Following these advancements, Heterogeneous Graph Representation Learning (HGRL) algorithms have been recently proposed to process such complex, heterogeneous graphs[4–6].

The core issue with HGRL is to simultaneously capture the structural properties of the network and the semantic properties of the heterogeneous nodes and edges; in other words, we need node and edge type-aware embeddings that can preserve both the structural and the semantic properties of the underlying heterogeneous graph.

In this context, from an algorithmic point of view, three main lines of research have recently emerged, both inspired by homogeneous network representation learning[1]: the first one leverages results obtained by homogeneous *Random-Walk (RW) based approaches*; they are based on the "distributional hypothesis"[1], firstly exploited to capture the semantic similarity of

[1]The distributional hypothesis was originally proposed in linguistics[7,8]. 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 purport similar meanings[8].

words[9], and then extended to capture the similarity between graph nodes[10]; the second one exploits neural networks specifically designed to process graphs, using e.g., convolutional filters[11], and more generally direct supervised feature learning through *Graph Neural Networks (GNNs)*[12]. The third research line[13–16] views heterogeneous graphs as knowledge bases expressing relationships (edges) between sources and destination nodes, and then learns the latent space where all such relationships are "optimally" represented.

Inheriting from the NLP field, *RW-based methods* share the assumption that nodes having the same structural context or being topologically close in the network (homophily) are also close in the embedding space. When applied to homogeneous graphs the node-neighborhoods are collected by running several RWs from each node and feed such neighborhoods to the training phase of NLP-based shallow neural-networks; once trained, the network weights are the node embeddings[17].

Heterogeneous RW-based methods leverage homogeneous ones by separately processing each homogeneous sub-network composing the original heterogeneous graph. As an example, in[18] the heterogeneous network is first projected into several homogeneous bipartite networks; then, an embedding representing the integrated multi-source information is computed by a joint optimization technique combining the skip-gram models individually defined on each homogeneous graph. A similar decomposition is initially applied in[19], where the original heterogeneous graph is split into a set of hierarchically structured homogeneous graphs. Each homogeneous graph is then processed through node2vec[10], and the embedding of the heterogeneous network is finally obtained by using recursive regularization, which encourages the different embeddings to be similar to their parent embedding. Another approach in this context constraints the RWs used to collect node contexts for the embeddings into specific *meta-paths*: the walker can step only between pre-specified pairs of vertices, thus better capturing the structural and semantic characteristics of the nodes[20]. Other related approaches combine vertex pair embedding with meta-path embeddings[21], or improves the heterogeneous Spacey RW algorithm by imposing meta-paths, graphs and schema constraints[22].

Differently from the distributional hypothesis approach that usually applies shallow neural networks to learn the embeddings, *GNN approaches* apply deep neural-network encoders to provide more complex representations of the underlying graph[23]. By this approach, the deep neural network recursively aggregates information from neighborhoods of each node in such a way that the node neighborhood itself defines a computation graph that learns how to propagate information across the graph to compute the node features[12, 24]. As it often happens for the distributional approach, the usual strategy used by GNNs to deal with heterogeneous graphs is to decompose them into its homogeneous components. For instance, Relational Graph Convolutional Networks[25] maintain distinct weight matrices for each different edge type, or Heterogeneous Graph Neural Networks[26], apply first-level Recurrent Neural Networks (RNN) to separately encode features for each type of neighbour nodes, and then a second level RNN to combine them. Also Decagon[27], which has been successfully applied to model polypharmacy side effects, uses a graph decomposition approach by which node embeddings are separately generated by edge type and the resulting computation graphs are then aggregated. Other approaches add meta-path edges to augment the graph[28] or learn attention coefficients that weight the importance of different types of vertices[29]. The drawback of all the aforementioned GNN approaches is that some relations may not have sufficient occurrences, thus leading to poor relation-specific weights in the resulting GNN. To overcome this problem, an Heterogeneous GNN[30] that uses the Transformer-like self-attention architecture have been recently proposed.

Relation-learning approaches[13–16] use contrastive learning techniques to project entities (head and tail nodes) and the relationships (edges) between them into low-dimensional latent spaces that preserve the relationships between entities and relationships. This is achieved by assigning a score to each (head, relation, tail) triple, which is maximized for true triples and minimized for "corrupted triples", that is triples not truly existing in the graph.

The simplest yet effective relation-learning technique is DistMult[14]; it projects triples into a latent space where the score maximized for true triples is the generalized dot product between the embeddings of the source node, destination node, and the edge connecting them. ComplEx[16] is an extension of DistMult that can deal with oriented relationships. To achieve this, it projects the graph entities into a complex space where the score maximized for true triples is computed as the generalized dot product in the complex space. TransE[13] and its extension to hyperplanes, TransH[15], are probably the most popular relation-learning techniques. TransE, projects triples into a latent space where, for true triples, the relation edge between two nodes is modeled as a translation vector between the source vector and the destination. The definition of TransE collapses all the triples into a unique latent space. This might decrease separability between relationships; therefore, TransH extends TransE by finding one hyperplane for each relation type.

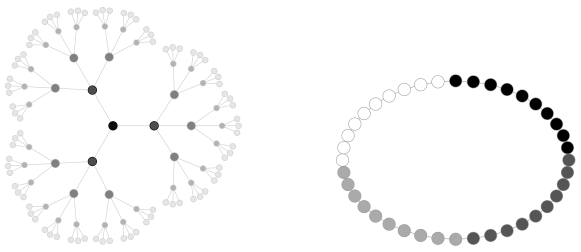
2 Impact of node2vec parameters on node embeddings

Figure 1 shows the embedding obtained by using different values for the inward and in-out parameters in two simple graphs: a 3-ary tree of depth five and a cycle of length 120. Graph topologies are outlined in Figure 1a. In these examples, the graphs are assumed to be unweighted, while the color of a node represents its global position in the graph. Namely, in the tree, the color of a node indicates its depth, while in the cycle, it represents the region where the node is placed according to a cyclic visit.

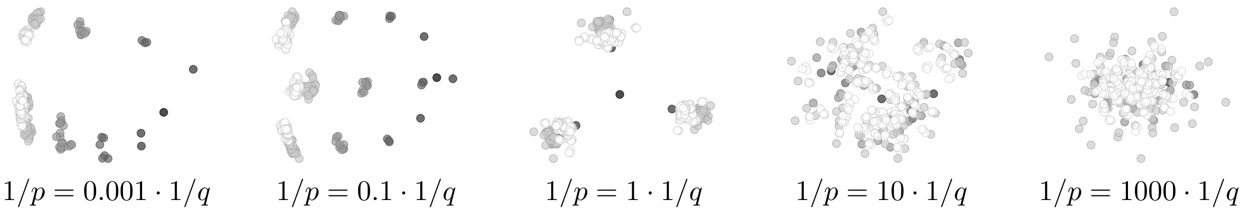
Notice that none of the graphs in the example contains a triangle (a complete subgraph of three vertices); therefore, the transition probabilities along the generation of a RW can take only the values $1/p$ or $1/q$. Therefore, the ratio q/p fully characterizes the parameter space by quantifying the relation between the local/global type of the generated paths.

Figures 1b,1c, 1d and 1e show the projections of the embeddings derived from node2vec for varying ratios of parameters p and q. The corpus of the embedding contains, starting from each node, ten random walks of length 128; the paths that resulted from these random walks were embedded using the Skipgram method with a window size of five, and embedded into a space of dimensions 25 for the tree and 5 for for the cycle, respectively. These embeddings were then projected into a two-dimensional space using both their first two principal components (PCA) and TSNE.

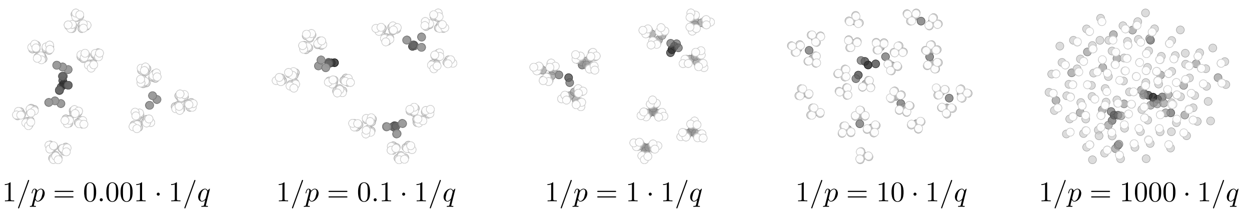
We notice that if the probability of return is smaller than that of exploration ($q/p < 1$), the resulting embedding better captures the global structure of the input graph. Conversely, if the return probability is larger, the embedding depicts the local structure of the graph around each node.



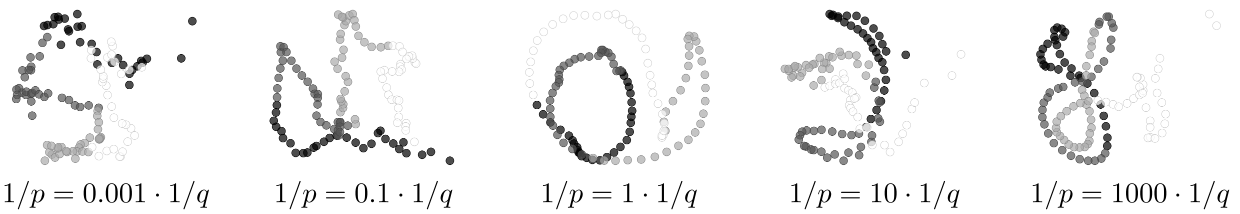
(a) Graph topologies, the colors of the nodes represent their position in the graph.



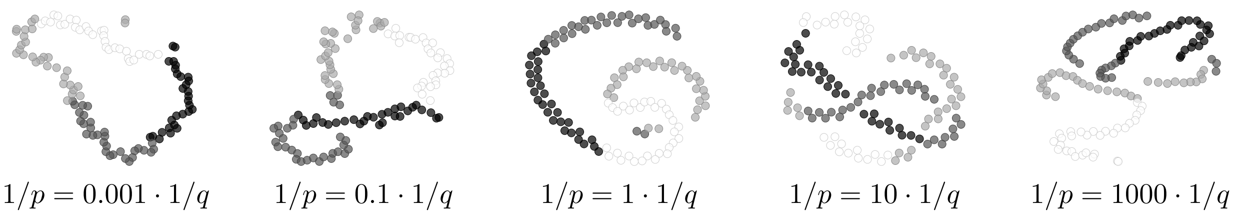
(b) PCA projections of the node2vec embeddings of the tree using a target space of dimension 25.



(c) TSNE projections of the node2vec embeddings of the tree using a target space of dimension 25.



(d) PCA projections of the node2vec embeddings of the cycle using a target space of dimension 5.



(e) TSNE projections of the node2vec embeddings of the cycle using a target space of dimension 5.

Figure 1. The figures depict the embeddings obtained for a 3-ary tree and a cycle using different values for the parameter p and q. The resulting path were embedded using the Skipgram architecture into a space of dimension 25 for the tree and 5 for the cycle, respectively. Final bi-dimensional projections were obtained using their first two principal components and TSNE.

3 Impact of *Het-node2vec* parameters on a Erdös-Rényi graph embedding

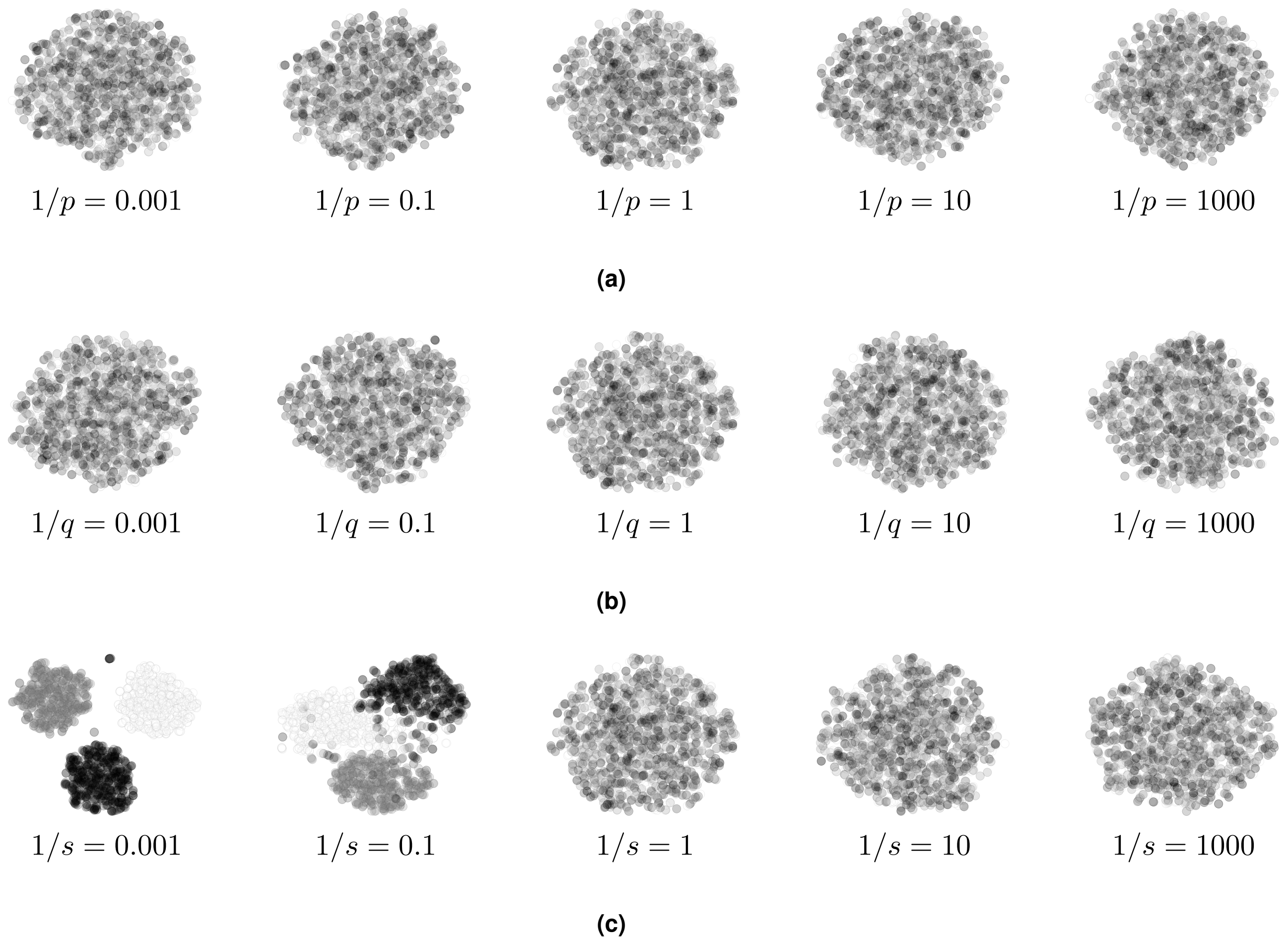
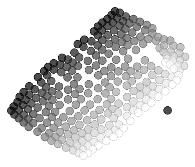
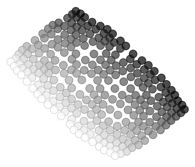
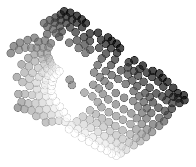


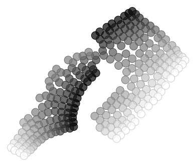
Figure 2. Comparison of *Het-node2vec* embeddings on a heterogeneous random graph generated according to a Erdös-Rényi model using 1500 nodes and a edge probability equal to 0.01. Nodes can have three node types and were assigned uniformly. For each setting of the s, q and p parameters in *Het-node2vec*, embeddings are computed by generating 10 RWs of length 128 from each node and a Skipgram architecture, with a window size equal to 5, to generate an embedding of the nodes into a 10D space. The final 2D representation is obtained by t-SNE. (a) *Het-node2vec* embeddings of the random graph from different inward switching values p and $q = s = 1$. (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 different node type switching values s and $p = q = 1$.

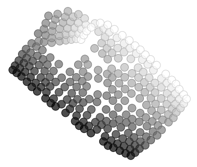
4 Impact of *Het-node2vec* parameters on the grid embedding

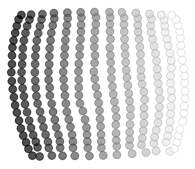

$1/p = 0.001$
$1/q = 0.001$
$1/s = 1$

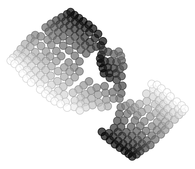

$1/p = 0.001$
$1/q = 0.1$
$1/s = 1$

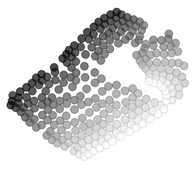

$1/p = 0.001$
$1/q = 1$
$1/s = 1$

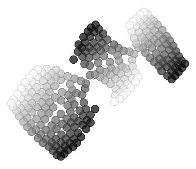

$1/p = 0.001$
$1/q = 10$
$1/s = 1$

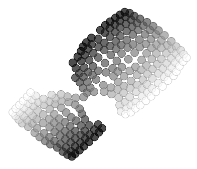

$1/p = 0.001$
$1/q = 1000$
$1/s = 1$

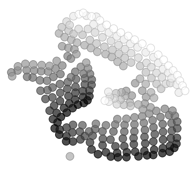

$1/p = 0.1$
$1/q = 0.001$
$1/s = 1$

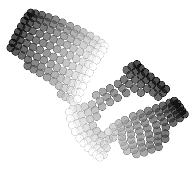

$1/p = 0.1$
$1/q = 0.1$
$1/s = 1$

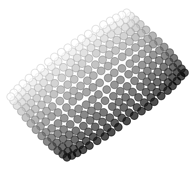

$1/p = 0.1$
$1/q = 1$
$1/s = 1$

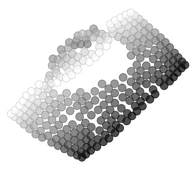

$1/p = 0.1$
$1/q = 10$
$1/s = 1$

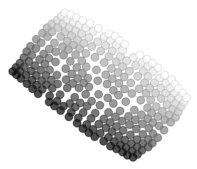

$1/p = 0.1$
$1/q = 1000$
$1/s = 1$

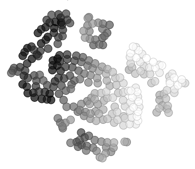

$1/p = 1$
$1/q = 0.001$
$1/s = 1$

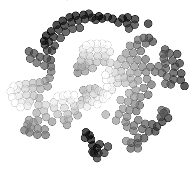

$1/p = 1$
$1/q = 0.1$
$1/s = 1$

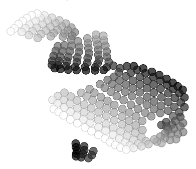

$1/p = 1$
$1/q = 1$
$1/s = 1$

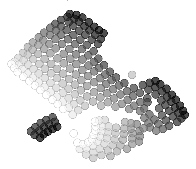

$1/p = 1$
$1/q = 10$
$1/s = 1$

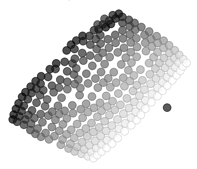

$1/p = 1$
$1/q = 1000$
$1/s = 1$

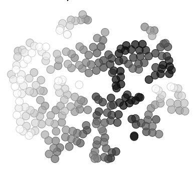

$1/p = 10$
$1/q = 0.001$
$1/s = 1$

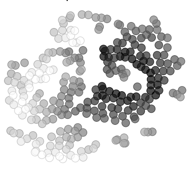

$1/p = 10$
$1/q = 0.1$
$1/s = 1$

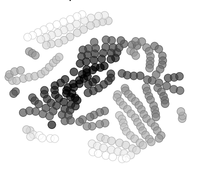

$1/p = 10$
$1/q = 1$
$1/s = 1$

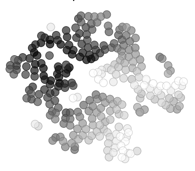

$1/p = 10$
$1/q = 10$
$1/s = 1$

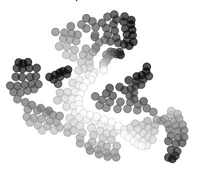

$1/p = 10$
$1/q = 1000$
$1/s = 1$

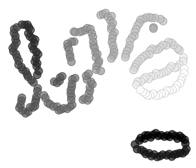

$1/p = 1000$
$1/q = 0.001$
$1/s = 1$

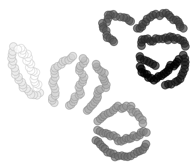

$1/p = 1000$
$1/q = 0.1$
$1/s = 1$

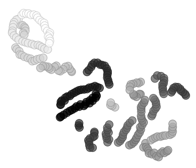

$1/p = 1000$
$1/q = 1$
$1/s = 1$

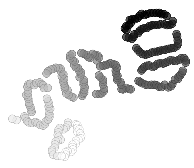

$1/p = 1000$
$1/q = 10$
$1/s = 1$

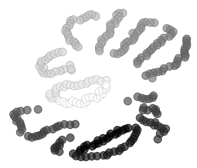

$1/p = 1000$
$1/q = 1000$
$1/s = 1$

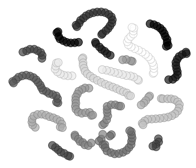

$1/p = 0.001$
$1/q = 0.001$
$1/s = 0.001$

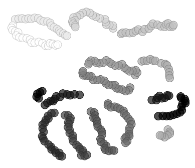

$1/p = 0.001$
$1/q = 0.1$
$1/s = 0.001$

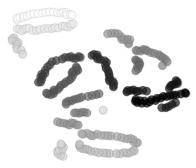

$1/p = 0.001$
$1/q = 1$
$1/s = 0.001$

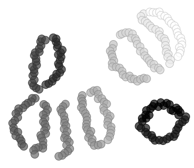

$1/p = 0.001$
$1/q = 10$
$1/s = 0.001$

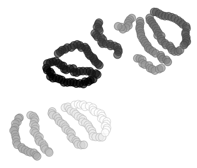

$1/p = 0.001$
$1/q = 1000$
$1/s = 0.001$

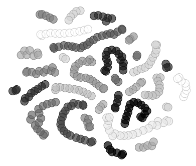

$1/p = 0.1$
$1/q = 0.001$
$1/s = 0.001$

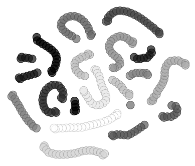

$1/p = 0.1$
$1/q = 0.1$
$1/s = 0.001$

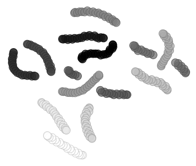

$1/p = 0.1$
$1/q = 1$
$1/s = 0.001$

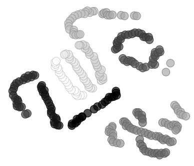

$1/p = 0.1$
$1/q = 10$
$1/s = 0.001$

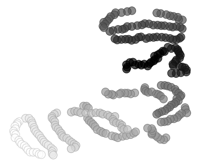

$1/p = 0.1$
$1/q = 1000$
$1/s = 0.001$

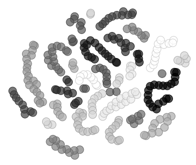

$1/p = 1$
$1/q = 0.001$
$1/s = 0.001$

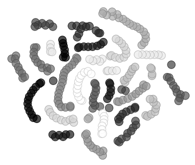

$1/p = 1$
$1/q = 0.1$
$1/s = 0.001$

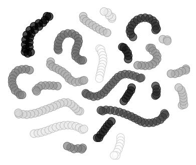

$1/p = 1$
$1/q = 1$
$1/s = 0.001$

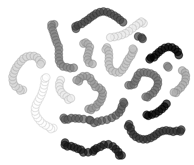

$1/p = 1$
$1/q = 10$
$1/s = 0.001$

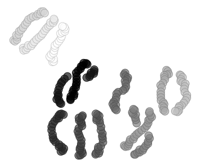

$1/p = 1$
$1/q = 1000$
$1/s = 0.001$

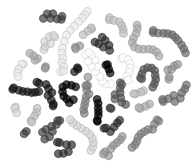

$1/p = 10$
$1/q = 0.001$
$1/s = 0.001$

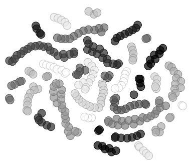

$1/p = 10$
$1/q = 0.1$
$1/s = 0.001$

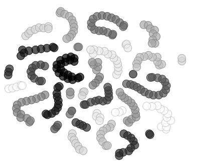

$1/p = 10$
$1/q = 1$
$1/s = 0.001$

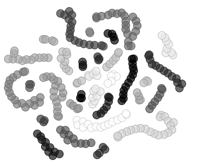

$1/p = 10$
$1/q = 10$
$1/s = 0.001$

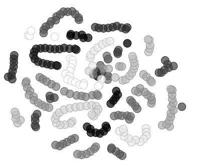

$1/p = 10$
$1/q = 1000$
$1/s = 0.001$


$1/p = 1000$
$1/q = 0.001$
$1/s = 0.001$


$1/p = 1000$
$1/q = 0.1$
$1/s = 0.001$


$1/p = 1000$
$1/q = 1$
$1/s = 0.001$


$1/p = 1000$
$1/q = 10$
$1/s = 0.001$


$1/p = 1000$
$1/q = 1000$
$1/s = 0.001$

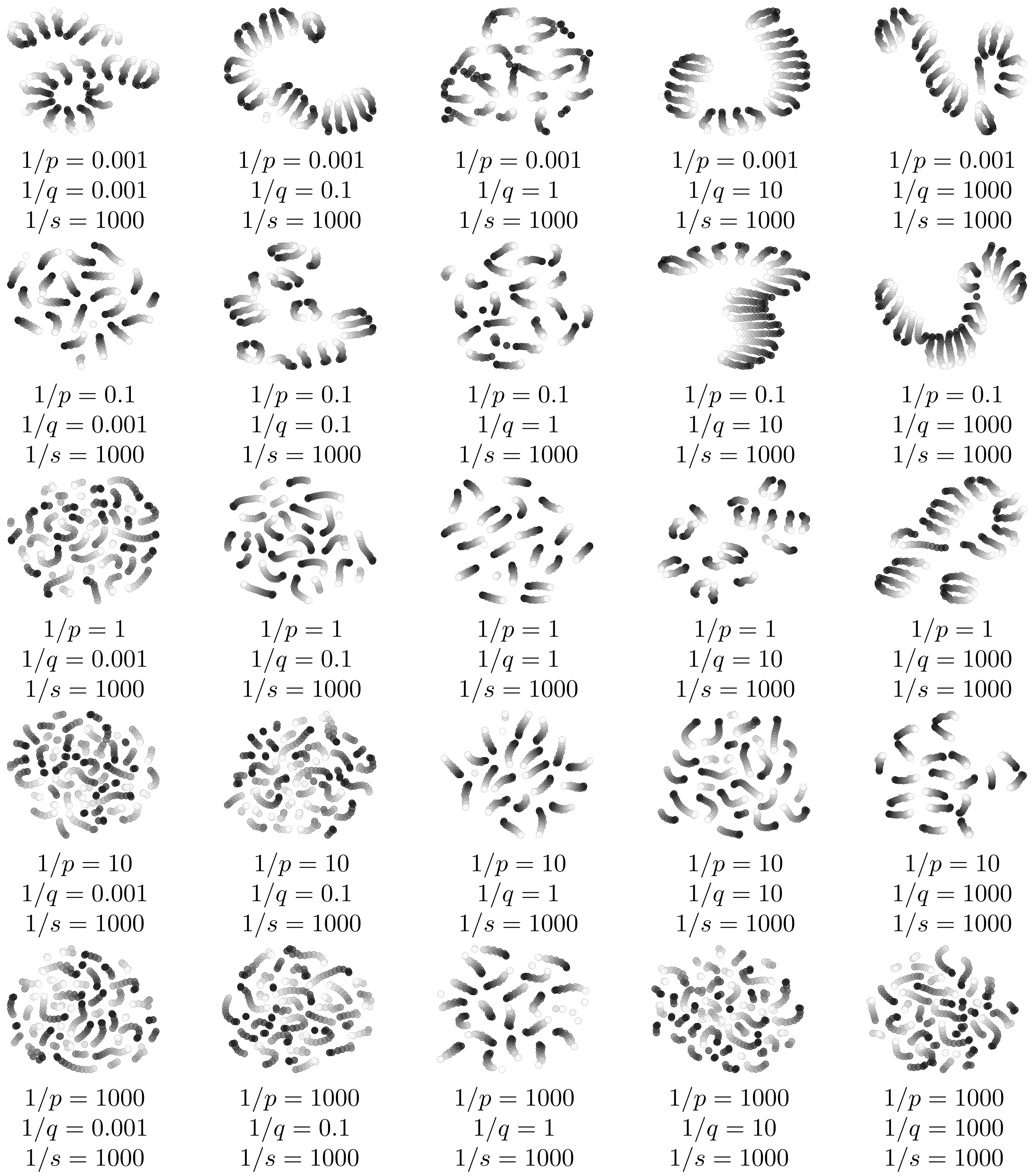


Figure 3. TSNE

Figure 4. Embedding obtaining for a grid using different values for the parameter p, q, s, which regulates the exploration biases of the generated random paths. The resulting paths were embedded using Skipgram, with a window of size five, into a space of dimensions 10 and then projected using TSNE.

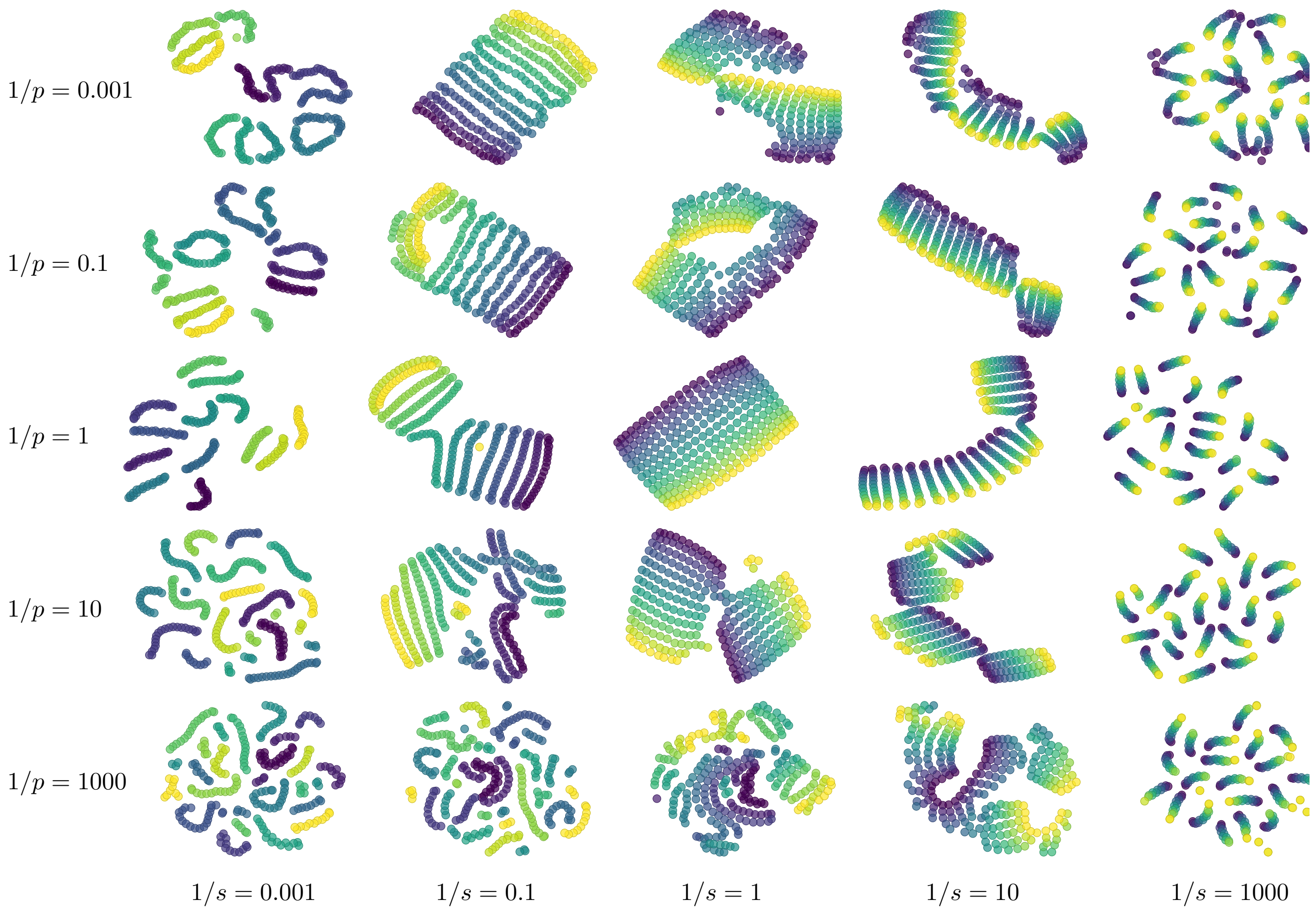


Figure 5. Comparison of *Het-node2vec* and node2vec embeddings on a synthetic heterogeneous graph having a grid topology for different values of the inward p and the type-switching s hyperparameter.

5 Supplementary experimental analysis of the switching parameters of *Het-node2vec*.

Figure 8 shows how *Het-node2vec* behaves on synthetic and real-world data sets when changing the s switching parameter.

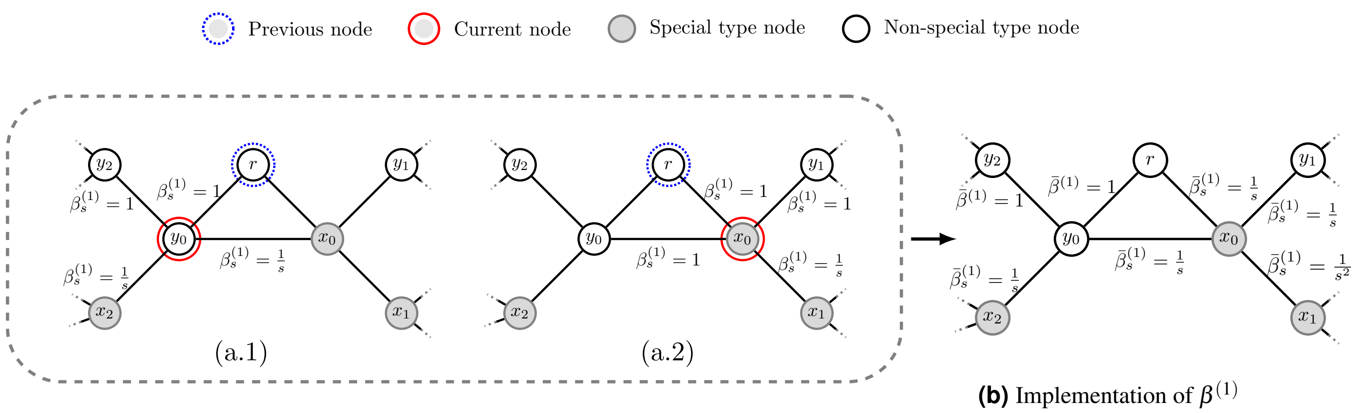
The figure shows the embeddings obtained on a synthetic graph composed of nine node types, organized into 3×3 square-grid, where each square is composed of nodes sharing the same type. The embeddings are computed using *Het-node2vec* with $p = 1$, $q = 1$, and by setting different values for the generic switching parameter $1/s$ (Supplementary Figure 8b). It can be seen that, by discouraging the node type switching ($1/s \ll 1$), we obtain embeddings that tend to cluster the nodes belonging to the same type. A similar example in a real-world graph is depicted in Supplementary Figure 8c where different embeddings are generated for the well-known Cora network[31] representing the relation between scientific publications. Both examples show that the t-SNE projections of *Het-node2vec*-embeddings can separate the different node types more effectively.

6 Implementation of special node-type swicthing: equivalence of transition probabilities on special node-switching strategies

Redefinition of $\beta_s^{(1)}$. To efficiently implement $\beta_s^{(1)}$ we redefine it as $\bar{\beta}_s^{(1)}(v,x)$ so that the normalized transition probability *Het-node2vec* obtains by using $\bar{\beta}_s^{(1)}(v,x)$ is proportional to the one it would obtain when using the original definition of the bias $\beta_s^{(1)}(v,x)$ (see also figure 6):

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

where $\oplus$ denotes the exclusive OR operator. In this section we show that this redefinition of $\beta_s^{(1)}(v,x)$, as well as the redefinition $\bar{\beta}_s^{(2)}(v,x)$ introduced for $\beta_s^{(2)}(v,x)$ (see Section 2.3 of the main manuscript), does not change the normalized transition probabilities computed by *Het-node2vec*.



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

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

Figure 6. Definition (subfigures a) and implementation (subfigure b) of special node-type switching strategy using $\beta_s^{(1)}$. (Subfigures a): when the RW starts from a non-special node (y_0 - subfigure a.1), $\beta_s^{(1)}$ biases the transition probability to promote/demote transitions towards special node types (i.e. the jump from y_0 to x_2). When starting from a special node (x_0 - subfigure a.2), $\beta_s^{(1)}$ promotes/demotes the transition toward another special node (e.g. x_1), independent of the type of the preceding node in the RW. (Subfigure b): each node-type switching function is implemented so that it only depends on the edge used in the transition and not on the direction of the RW. In the case of $\beta_s^{(1)}$, the bias applied to edges connecting two non-special nodes is 1; if the edge connects a non-special node to a special node the bias is $1/s$, while the weight over the edge connecting two special node types is $1/s^2$. Applying this implementation strategy (subfigure b), when we consider a walk starting from, e.g., y_0 and we normalize the biases, the computed switching probability will promote/demote switches towards a special node (x_0). When we instead consider a walk starting from a special-node (e.g., x_0) the jump toward a special node (x_1) is $1/s$ times lower than the bias applied to all other outbound edges; after normalizing these biases to ensure they sum to 1, the probability of switching from x_0 to x_1 will be $1/s$ times larger (if $s < 1$) or lower (if $s > 1$) than the probability of switching to any other neighbor, according to the *Het-node2vec* design.

The two special node-type switching strategies defined in Section 2.2 of the main manuscript are:

$$\beta_s^{(1)}(v,x) = \begin{cases} \frac{1}{s} & \text{if } \phi(x) \in \Sigma_{\phi_{\mathscr{S}}} \\ 1 & \text{otherwise.} \end{cases} \quad \text{and} \quad \beta_s^{(2)}(v,x) = \begin{cases} \frac{1}{s} & \text{if } \phi(v) \notin \Sigma_{\phi_{\mathscr{S}}} \wedge \phi(x) \in \Sigma_{\phi_{\mathscr{S}}} \\ 1 & \text{otherwise.} \end{cases}$$

$\bar{\beta}_s^{(1)}$ and $\bar{\beta}_s^{(2)}$ are the redefined implementations for these strategies defined in Equation 10 and 11:

$$\bar{\beta}_s^{(1)}(v,x) = \begin{cases} \frac{1}{s} & \text{if } \phi(v) \in \Sigma_{\phi_{\mathscr{S}}} \veebar \phi(x) \in \Sigma_{\phi_{\mathscr{S}}} \\ \frac{1}{s^2} & \text{if } \phi(v) \in \Sigma_{\phi_{\mathscr{S}}} \wedge \phi(x) \in \Sigma_{\phi_{\mathscr{S}}} \\ 1 & \text{otherwise,} \end{cases} \quad \text{and} \quad \bar{\beta}_s^{(2)}(v,x) = \begin{cases} \frac{1}{s} & \text{if } \phi(v) \in \Sigma_{\phi_{\mathscr{S}}} \vee \phi(x) \in \Sigma_{\phi_{\mathscr{S}}} \\ 1 & \text{otherwise.} \end{cases}$$

We first notice that for both strategies $\beta_s^{(k)}$, $k \in \{1,2\}$ it holds that

(1) if v is a non-special node, then $\bar{\beta}_s^{(k)}(v,x) = \beta_s^{(k)}(v,x)$ and,

(2) if v is a special node, then $\bar{\beta}_s^{(k)}(v,x) = \frac{1}{s}\beta_s^{(k)}(v,x)$.

In the following, we show that transition probabilities induced by the strategies $\beta_s^{(1)}$ and $\beta_s^{(2)}$ are equal to those defined by their implementations $\bar{\beta}_s^{(1)}$ and $\bar{\beta}_s^{(2)}$ respectively.

Let v be the node currently visited by a random walk coming from a node r and generated according to the strategy $\bar{\beta}_s^{(k)}$. We observe that there are two possible cases for node v, i.e. v maybe or maybe not a special node, and the functions $\beta_s^{(k)}$ completely define the possible switching to a node x that can be special or non special. Hence, the normalized transition probability from node v to its neighbours $x \in \mathscr{N}(v)$ includes two possible cases:

Case 1. If v is a non-special node then

$$\begin{aligned} P\Big(X_{t+1} = x | X_t = v, X_{t-1} = r\Big) &= \frac{\bar{\beta}_s^{(k)}(v,x) \cdot \alpha_{pq} \cdot w_{vx}}{\sum\limits_{vz \in E} \bar{\beta}_s^{(k)}(v,z) \cdot \alpha_{pq} \cdot w_{vz}} \\ &= \frac{\beta_s^{(k)}(v,x) \cdot \alpha_{pq} \cdot w_{vx}}{\sum\limits_{vz \in E} \beta_s^{(k)}(v,z) \cdot \alpha_{pq} \cdot w_{vz}}. \end{aligned}$$

Case 2. If v is a special node then

$$\begin{aligned} P\Big(X_{t+1} = x, | X_t = v, X_{t-1} = r\Big) &= \frac{\bar{\beta}_s^{(k)}(v,x) \cdot \alpha_{pq} \cdot w_{vx}}{\sum\limits_{vz \in E} \bar{\beta}_s^{(k)}(v,z) \cdot \alpha_{pq} \cdot w_{vz}} \\ &= \frac{\frac{1}{s}\beta_s^{(k)}(v,x) \cdot \alpha_{pq} \cdot w_{vx}}{\sum\limits_{vz \in E} \frac{1}{s}\beta_s^{(k)}(v,z) \cdot \alpha_{pq} \cdot w_{vz}} \\ &= \frac{\beta_s^{(k)}(v,x) \cdot \alpha_{pq} \cdot w_{vx}}{\sum\limits_{vz \in E} \beta_s^{(k)}(v,z) \cdot \alpha_{pq} \cdot w_{vz}}. \end{aligned}$$

In both cases the strategies $\beta_s^{(k)}$ and $\bar{\beta}_s^{(k)}$ define the same transition probabilities, hence $\beta_s^{(k)}$ and $\bar{\beta}_s^{(k)}$ model the same type-aware switching strategy.

Implementation 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. This is because the node-type switching strategies we introduce either depend solely on the types of nodes involved in a transition or can be reformulated to do so, without relying on transition direction. As a result, the switching bias can be precomputed and integrated into the edge weights prior to RW execution, without increasing the time complexity of the random walk generation process. Once this preprocessing step is completed, the standard node2vec algorithm can be applied without modification.

Implementation of the generic node-type switching strategy. the generic node-type switching strategy only depends on the source and destination-node types; therefore, the switching-bias can be applied by simply scaling the weights of edges connecting nodes of different types by a factor $\beta_s = \frac{1}{s}$.

Implementation of the special node-type switching strategy.

The special node-type switching strategy $\beta_s^{(2)}$ defines a bias that depends on the direction of the transition; however, it can be reformulated to remove this directional dependence, enabling the same preprocessing-based approach.

In Equation 2 we provide an alternative definition, $\bar{\beta}_s^{(2)}(v,x)$, of $\beta_s^{(2)}(v,x)$; their effect is equivalent because the normalized transition probabilities computed by *Het-node2vec* are maintained. Equation 2 is independent on the transition direction and it only depends on the type of the edge used to transition.

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

Similar considerations hold for $\bar{\beta}_s^{(1)}(v,x)$, i.e. the redefinition and implementation of $\beta_s^{(1)}(v,x)$ (see Figure 6a).

We outline that The equivalence between the *Het-node2vec* formulation and its implementation, where the β_s parameters are encapsulated in the graph weights, is established and becomes clear when the biases are normalized to sum to 1 to resemble a probability distribution.

Model complexity and scalability

In its simplest implementation, RW implementation as node2vec only stores the node neighborhood, which requires $\mathscr{O}(|E|)$ 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 $\mathscr{O}(\bar{d}^2|V|)$ space requirement, where $\bar{d}$ denotes the mean degree of a node, which is small in most applications[10]. 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 the time complexity of $\mathscr{O}(\frac{L}{l(L-l)})$ per sample[10]. Furthermore, the sampling procedure can be further optimized by parallelization and the use of succinct data structures[31,32]. 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 $\mathscr{O}(|E|)$ 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.

To explore the scalability of the model, and similarly to[10], we performed an empirical analysis of the computation time evolution using the generic strategy in *Het-node2vec* for both RW sampling process and sampling process plus the embedding construction (Figure 7). The evaluation is conducted on two random graph models: the Erdős-Rényi model, where the edge probability is set to obtain a graph with average degree of ten; and a model generating random graphs with power-law degree distribution characterized by an exponent $\alpha = 2.2$[33], which is a common graph topology in many practical applications. As can be seen, in both models, the evolution of the computation time in the plot follows a linear trend, as in the case of node2vec[10].

Link prediction performance comparison between node-type switching strategies

Table 1. Performance metrics comparison between the node-switching strategies for the link prediction and edge prediction tasks on a benchmark dataset (Pubmed). The dataset is described in Section 7 while the experimental setting for the supervised tasks is described in Section 4.2

HetNode2vec special	Node label prediction (Strategy 1/Strategy 2)				Edge prediction (Strategy 1/Strategy 2)			
Parameter	Macro F1		Micro F1		AUC		MRR	
$1/s = 0.1$	9.25	18.84	16.29	21.58	68.88	70.59	89.17	88.17
$1/s = 1$	9.89	11.33	16.73	14.09	58.52	69.31	87.80	87.79
$1/s = 10$	10.46	10.60	16.96	14.31	59.30	63.24	85.18	84.26
$1/s = 100$	10.59	8.77	16.74	11.67	59.38	65.88	85.56	86.03

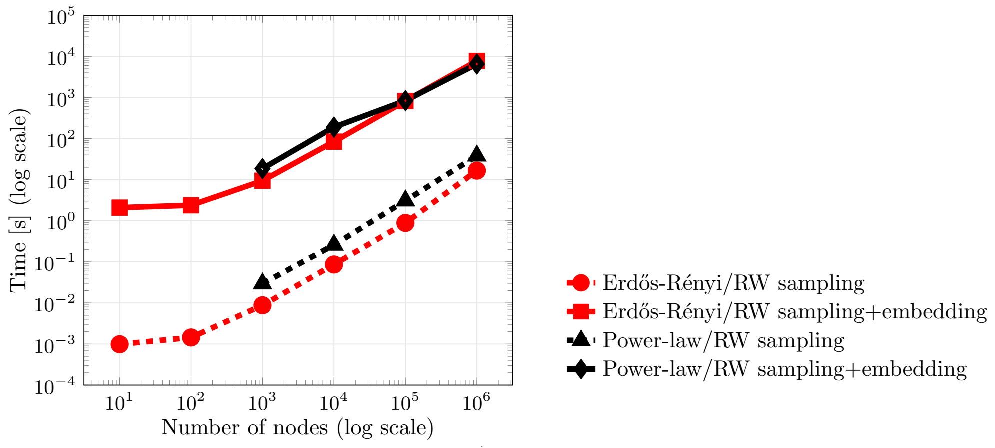


Figure 7. Computation time evolution of the *Het-node2vec* representation in Erdős-Rényi graphs with an average degree of 10 and random power-law graphs with exponent $\alpha = 2.2$. For each family, graphs nodes and edges were randomly assigned to one of ten classes. Starting from each node, 10 RWs of length 100 were generated using the parameters $p = 0.25, q = 4$, and $s = 0.5$. The set of random walk was used as the input of a Skipgram model to produce a vectorial embedding of size 100. Computations have been performed using a processor AMD Rome 7452, 2.3 GHz, 32 cores with a RAM of 1024GB.

7 Datasets

The Freebase network is constructed based on the collaborative knowledge base Freebase[2], containing relations between books, films, music, sports, people, locations, organizations, and businesses. Each node possesses a unique type and no features, meaning that the nodes are not attributed, while each edge is defined by a unique type characterized by the types of its endpoint nodes.

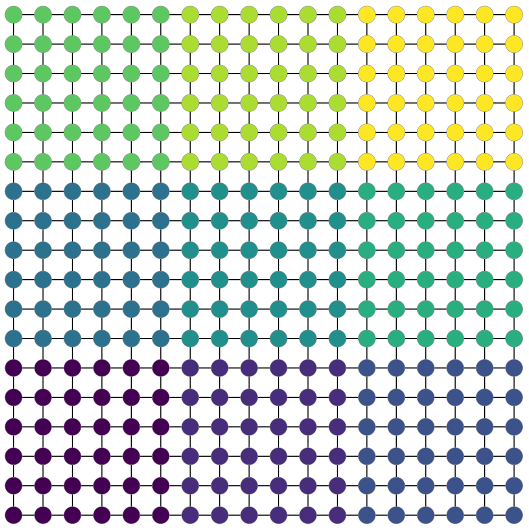
Freebase is the largest of the graphs concerning node-cardinality and exhibits the highest number of types for nodes and edges, comprising eight node types connected by 36 edge types. Both node and edge types have a highly unbalanced distribution, where two node/edge types are significantly more prevalent compared to others (see Table 2 and Figure 9). As shown in the CCDF in Figure 10a , the degree distribution of the nodes shows a long tail across all types of nodes, with the `music` nodes exhibiting the highest mean degree (mostly connected to other nodes), while the (target) `book` nodes have the lowest mean degree.

The DBLP network is derived from the well known dataset DBLP[3], and collects bibliographical information related to computer science publications. The creators of the dataset [47] built on DBLP dataset to construct an attributed multi-graph where node types are authors, phrases, venues, and years; and where node attributes are characterized by a 300-dimensional feature vector. Phrases nodes represent “quality phrases” extracted semi-automatically from each paper using the AutoPhrase algorithm , and followed by human curation. Each phrase node is connected to the authors/venue/year nodes associated with the paper from which the phrase was sourced. For the construction of attributes for `phrases` and `paper` nodes, authors aggregated the word2vec representations of their constituent words. For `author`, `venue`, `year` nodes, attributes are formed by the aggregation of the feature vectors derived from their related papers (for instance, those published by an author, in a particular venue, or within a certain year). Additionally, as a result of a web mining process, a small group of authors have been categorized into 12 research groups spanning four research areas and used for the node-label prediction task.

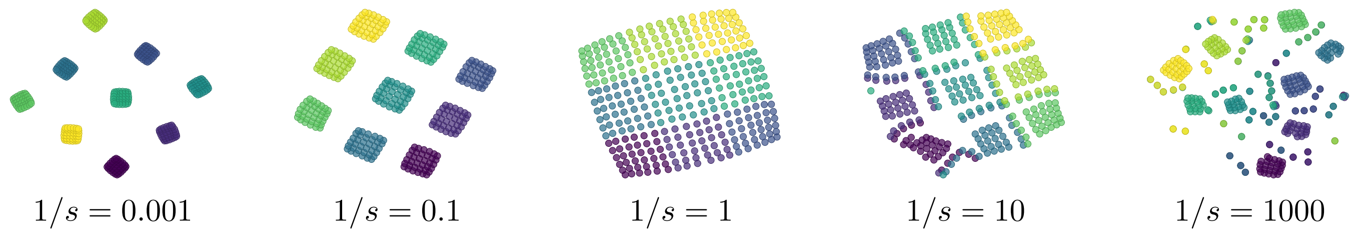
DBLP ranks second in terms of node cardinality, but it exhibits the largest number of edges and therefore the highest values for the node degrees.

The Yelp network represents relationships between reviews (phrases), businesses, locations, and stars (rating) nodes extracted from the Yelp dataset[4]. As shown in Table 2, `phrase` nodes form most of the graph, covering close to 91% of the overall nodes. Nodes of type *business* follow next accounting for almost 9% of the total nodes, while the remaining two types, `location` (25 nodes) and `stars` (9 nodes), represent a negligible portion of the network and could be viewed as hub nodes.

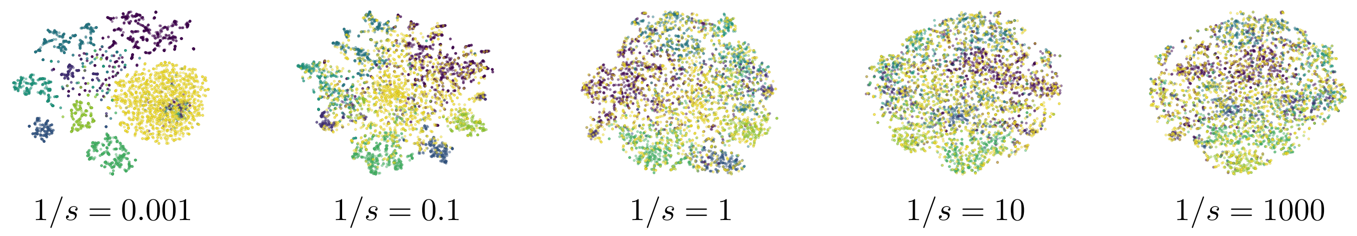
[2]http://www.freebase.com
[3]https://dblp.org
[4]https://www.yelp.com/dataset/challenge



(a) A grid structure with groups; the graph is generated by considering a grid partitioned into different regions composed of intervals of contiguous rows and columns defining a node type.



(b) *Het-node2vec* embeddings of the clustered grid for different node type switching values s and $p = q = 1$.



(c) *Het-node2vec* embeddings of the Cora graphs for different node type switching values s and $p = q = 1$.

Figure 8. Embedding obtained by using different values for the node switching parameter s ($p = 1$, $q = 1$). Using this *Het-node2vec* setting, embeddings are computed by generating 10 RWs of length 128 from each node. These paths are input to a Skipgram architecture, with a window size equal to 5, to generate an embedding of the nodes into a 10D space. The final 2D representation is obtained by t-SNE. (a) A grid topology divided into nine node types according to their region. (b) Embeddings obtained from the clustered grid. (c) Embeddings obtained from the Cora network.

The PubMed network is constructed starting from the PubMed database [5], and contain nodes representing genes, diseases, chemicals, and species. All nodes are extracted from PubMed papers through the AutoPhrase algorithm, nodes are typed using bioNER, and filtered by human experts. All the nodes in the networks are attributed. To construct the node features, PubMed papers were represented by 200 dimensional vectors via word2vec, and node vectors were subsequently obtained through an aggregation process.

Among the networks, PubMed is the smallest one with all node types exhibiting a long-tail distribution on their degrees (see Figure 10d). Compared to other networks, PubMed displays a lower level of unbalancing of nodes types, edge types, and (disease) label distribution, as shown in Table 2 and Figures 9,11.

8 Prediction tasks on real-world data sets

Predictive tasks. With the Freebase dataset, node-label classification is applied to nodes of type *book*, the second most common node type covering 22.8% of the overall nodes, surpassed only by *music* nodes (46.6% – Table 2). A significant portion of book nodes has been labeled into eight unbalanced literature genres, resulting in an unbalanced multi-class classification problem (Figure 11a). The edge prediction task is performed on the `book-book` edge type, accounting for 5.95% of the edges. It is the second most common edge type after edges connecting two `music` nodes (`music-music`), which make up 61.77% of the edges (Figure 9a).

With DBLP, node-type prediction task is performed with *author* nodes. The authors have been categorized into 12 research groups spanning four research areas and used for the node-label prediction task. In this graph, `author` and `phrases` nodes represent almost the entire graph, accounting for around 96% of both nodes and edges. The edge-prediction task is focused on the two `author-author` edge types, i.e. the relations of type `co-author` and `cite`, which account for about the 59% of all edges in the graph (Figure 9a).

With Yelp, the `business` node has been selected for the node-label prediction task. Business nodes account for about 9% of the overall nodes and are labeled with one or multiple categories among sixteen potential ones. These categories have an uneven distribution, which results in an unbalanced multi-label prediction (Figure 11c). For edge prediction the `business-phrase` has been selected, thus resulting in a prediction task involving almost the 9% of the overall Yelp edges (Figure 9c).

Node-label prediction task in PubMed is applied to `disease` nodes, which were labeled into eight categories, thus resulting in a multi-class classification problem. Disease nodes make up approximately 32% of the total nodes, representing the second most common type. `Disease-disease` edges, on which edge prediction task is focused, constitute about 14% of the network edges (Figure 9d).

[5] https://www.ncbi.nlm.nih.gov/pubmed/

Table 2. Node type distribution and basic degree statistics in the four heterogeneous network datasets. Proportion refers to the ratio of a given type of node with respect to the total number of nodes. The target node types used for the node type prediction task are highlighted in bold.

Graph	Node type	Proportion	Degree		
			mean	min	max
Freebase	music	0.466	14.52	1	1,086,802
	book	0.228	3.73	1	131,957
	people	0.130	5.17	1	130,116
	location	0.060	9.35	1	684,726
	film	0.051	11.36	1	100,825
	business	0.041	11.15	1	445,716
	organization	0.013	7.85	1	174,646
	sports	0.011	19.289	1	1,170,520
DBLP	**author**	0.8880	188.68	1	71,861
	phrase	0.1094	739.55	1	294,453
	venue	0.0026	981.97	1	54,396
	year	0.00004	58742.77	1	385,014
Yelp	phrase	0.9088	761.57	1	121,333
	business	0.0906	357.14	78	3,625
	location	0.0005	191.64	1	2,443
	stars	0.0001	830.44	4	2,239
Pubmed	chemical	0.420	8.04	1	7,272
	disease	0.319	7.48	1	18,714
	gene	0.215	6.83	1	2,474
	species	0.045	5.69	1	1,008

edge-type distribution	
music-and-music	0.6177
book-and-book	0.0595
business-about-music	0.0436
location-and-location	0.0348
film-and-film	0.0347
people-and-people	0.0213
people-to-sports	0.0197
people-to-book	0.0191
people-to-music	0.0187
people-on-location	0.0134
business-and-business	0.0132
people-to-film	0.0114
music-in-film	0.0105
book-on-location	0.0075
music-in-book	0.0075
book-to-film	0.006
music-on-location	0.0057
location-in-film	0.0055
people-in-business	0.0051
sports-and-sports	0.0049
sports-in-film	0.0046
business-about-film	0.0043
business-on-location	0.0039
organization-to-music	0.003
business-about-book	0.003
book-about-organ	0.0029
organization-on-location	0.0025
organization-for-business	0.0025
sports-on-location	0.0023
organization-in-film	0.0023
organization-and-organization	0.0022
people-in-organization	0.002
music-for-sports	0.0019
book-on-sports	0.0017
business-about-sport	0.0007
organization-to-sport	0.0006

(a) Freebase

edge-type distribution		
author-author	: cite	0.555
phrase-phrase	: co-occur	0.273
author-phrase	: study	0.092
author-author	: co-author	0.041
author-venue	: publish-in	0.020
author-year	: active-in	0.019

(b) DLBP

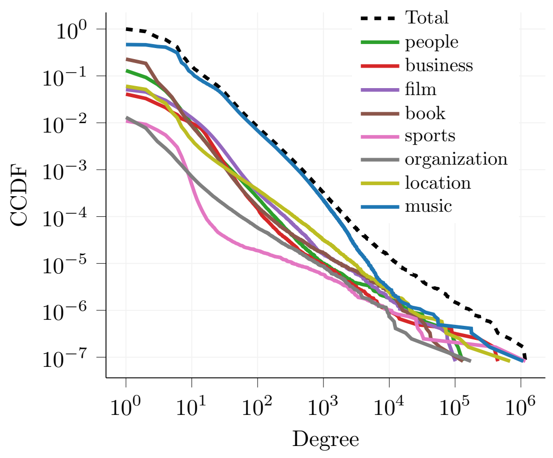
edge-type distribution	
business-in-location	0.0003
business-rate-stars	0.0003
business-described with-phrase	0.0888
phrase-context-phrase	0.9107

(c) Yelp

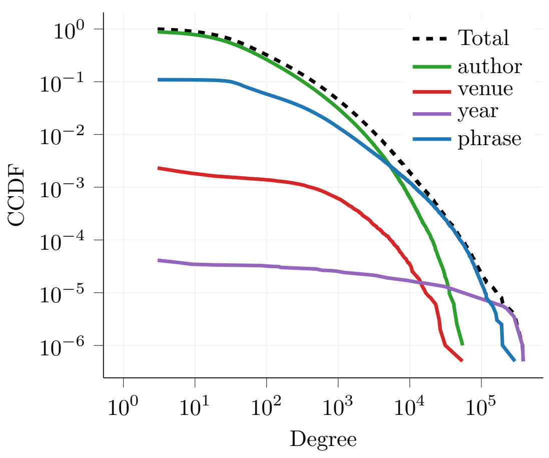
edge-type distribution	
chemical-and-chemical	0.263
chemical-in-disease	0.217
disease-and-disease	0.144
chemical-in-gene	0.132
gene-causing-diserse	0.111
gene-and-gene	0.068
chemical-in-species	0.027
species-with-disease	0.022
species-with-gene	0.013
species-and-species	0.003

(d) Pubmed

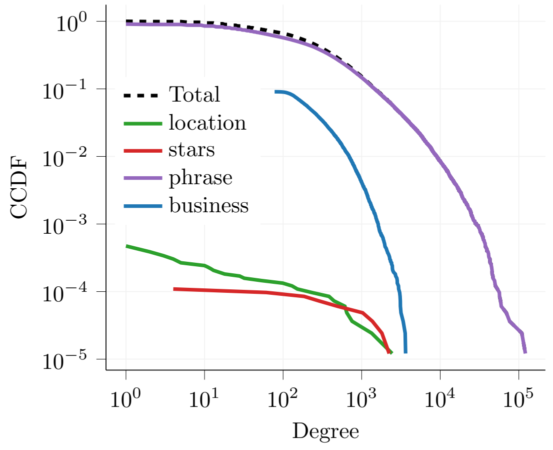
Figure 9. Edge-type distribution of the training set for the benchmark datasets. The target edge type used for the link prediction task in each dataset is highlighted in gray.



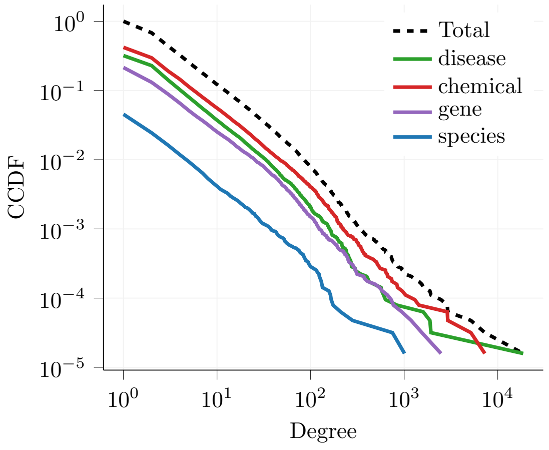
(a) Freebase



(b) DBLP



(c) Yelp



(d) PubMed

Figure 10. Complementary Cumulative Distribution Function of degrees with respect to their node types.

label distribution for node type **book**

Book	0.085
Film	0.382
Music	0.240
Sports	0.024
People	0.145
Location	0.085
Organization	0.047
Business	0.012

(a) Freebase

label distribution for node type **author**

Class 1	0.167
Class 2	0.039
Class 3	0.068
Class 4	0.029
Class 5	0.159
Class 6	0.037
Class 7	0.052
Class 8	0.086
Class 9	0.128
Class 10	0.023
Class 11	0.149
Class 12	0.040
Class 13	0.024

(b) DLBP

label distribution for node type **business**

Shopping	0.043
Event Planning & Services	0.032
Automotive	0.035
Italian	0.029
Beauty & Spas	0.055
Pizza	0.032
Sandwiches	0.039
Food	0.085
Bars	0.071
Breakfast & Brunch	0.044
Restaurants	0.295
American (Traditional)	0.050
Nightlife	0.074
Burgers	0.024
Mexican	0.043
American (New)	0.046

(c) Yelp

label distribution for node type **disease**

Cardiovascular disease	0.152
Glandular disease	0.115
Nervous disorder	0.105
Communicable disease	0.095
Inflammatory disease	0.216
Pycnosis	0.141
Skin disease	0.083
Cancer	0.093

(d) Pubmed

Figure 11. Node-label distribution of target node types in the four heterogeneous network dataset benchmarks.

9 Experimental hyperparameters

Table 3. Hyperparameters of the methods used in the construction of the embedding and their evaluation. While the default value for the number of RWs is 10 in Grape, we used five-fold cross-validation to assess whether a ten times larger value (no of RWs = 100) could improve the unsupervised embedding performance.

	hyperparameter	value
RW generation	Number of RWs (iterations)	10-50
	$1/p$	0.25
	$1/q$	4
	Walk length	100
Skipgram	Embedding size	50
	Epochs	10
	Ratio neg:pos samples	10:1
	Window size	5
	Max neighbours	100
	Learning rate	0.01
	Learning rate decay	0.9
Linear SVM	penalty	l2
	tol	0.0001
	C	0.1, 0.001
	max iter	3000
Random Forest	Number of estimators	100
	criterion	gini
	min samples	1
	max depth	None
	max iter	1000
	max features	sqrt

10 *Het-node2vec* results for the edge prediction tasks on HNE benchmark data sets

For the edge-prediction task, 20% of left-out edges are integrated with an equal amount of randomly sampled negative edges (i.e., non-existing edges) of the same type; this creates a balanced set of positive and negative edges. Next, the Hadamard operator is applied to the embeddings of the source and destination nodes to obtain the edge embeddings.

The edge-prediction task is then carried out using five-holdout cross-validation (80:20 train:test ratio) and linear support vector machines as classifier models. The evaluation is conducted by averaging the AUC (area under the ROC curve) and the MRR (mean reciprocal rank) performance measures obtained on each holdout. While AUC is a standard measure for classification when link prediction is regarded as a binary (existence/non-existence) classification problem, MRR is a standard measure for ranking when link prediction is used for link retrieval. Formally, if V_{test} is the set of source nodes of the test edges, the MRR is computed as

$$\text{MRR} = \frac{1}{|V_{test}|} \sum_{v \in V_{test}} \frac{1}{|E_+|} \sum_{e \in E_+} \frac{1}{\text{rank}_e},$$

where E_+ is the set of positive edges starting from the source node $v \in V_{test}$, and rank_e is the rank of the value computed by the svm for the positive test edge e with respect to all edges starting from v (both negative and positive). In practice, the MRR assesses the model's ability to assign high scores to true edges and low scores to negative edges.

The results obtained in the edge-prediction task, depicted in Figure 12, provide further evidence that the proposed type-aware RWs improve node2vec results.

When observing results achieved by the generic-type switching strategy (left column in Figure 12), better results are always achieved when a higher value of the switching parameter is used. In other words, promoting the heterogeneity of the node types, i.e., biasing the walk to step through edges with endpoints of different types, correlates with an improved graph representation.

This is due to a better exploration of the heterogeneous context of nodes at the endpoints of the targeted edges (highlighted in gray in Figure 9 in the Supplementary Information).

When using the special node-type switching strategy (right column of Figure 12), we note an opposite decreasing trend for all graphs with the exception of Yelp. For Freebase, DBLP, and PubMed, the types of edges targeted by the edge-prediction task, as well as the node types at their endpoints, are well represented in the graph. Using high β_s values to promote switching to special node types (i.e., the node types at the endpoints of the targeted edges) results in almost homogeneous RWs that repeatedly visit only special nodes and use only the targeted edges, neglecting others. This results in embeddings that are not sufficiently informative.

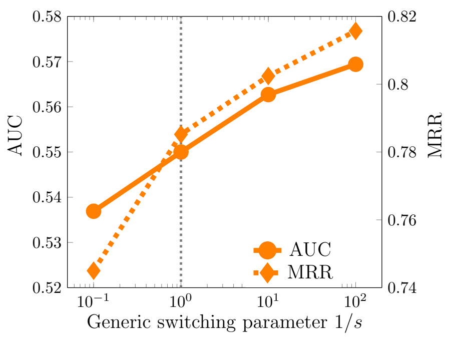
In Yelp, we instead observe an increasing trend mainly because the edges targeted by the link prediction (`business--described-with-phrase`, see Figure 9 in the Supplementary Information) connect the special node type `business`, which is relatively rare (approximately 9% of all nodes in the graph), to the non-special node type `phrase` which dominates the graph (approximately 91% of all nodes). As a result, the edges targeted by the link prediction are much less represented (approximately 9% of all edges) than the most common `phrase-context-phrase` edges (approximately 91% of all edges). Focusing the walk on `business` nodes enhances the likelihood of traversing the edge of interest, thereby producing embeddings that are informative for link prediction.

Summarizing, we observe that for both the node label and edge prediction tasks, the switching process that characterizes *Het-node2vec* with respect to the classical node2vec algorithm ($1/s = 10^0$, vertical dotted lines in Figure 12) improves the prediction results.

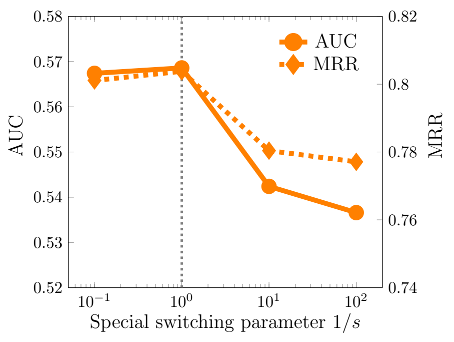
Table 4 summarizes the results of the comparison of *Het-node2vec* with other state-of-the-art methods for edge prediction on real-world benchmark data sets. Table 5 outlines the results achieved by *Het-node2vec* when their node and edge type-aware embeddings are used to train a Random Forest for node label and edge prediction problems.

Table 4. Performance metric for the link prediction task on the benchmark graphs. In the table, for both the *Het-node2vec* settings, we report only the values obtained by the two extreme values of the node-type switching parameter $1/s = 0.1, 100$.

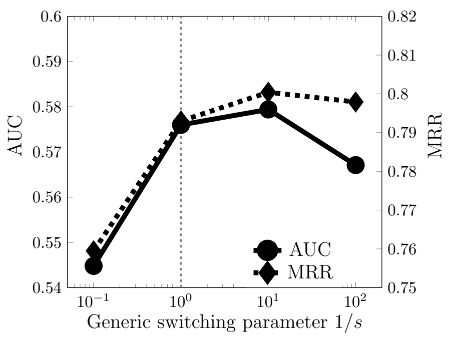
		Edge prediction (AUC/MRR)							
Model		DBLP		Yelp		Freebase		Pubmed	
node2vec		57.60	79.39	57.27	99.59	55.00	78.52	67.67	86.52
HetNode2vec generic	$1/s = 0.1$	54.48	75.95	60.22	99.38	53.69	74.50	65.60	84.35
	$1/s = 100$	56.71	79.79	79.13	99.34	56.94	81.57	70.04	87.05
HetNode2vec generic	$1/c = 0.1$	54.99	77.29	74.14	99.38	55.90	78.32	70.25	88.19
	$1/c = 100$	52.38	73.26	75.73	99.66	54.35	78.23	68.47	87.66
HetNode2vec special	$1/s = 0.1$	59.12	81.04	69.15	99.54	56.74	80.11	70.59	88.17
	$1/s = 100$	57.04	78.69	78.62	99.53	53.66	77.71	65.88	86.03
HetNode2vec special	$1/c = 0.1$	54.99	77.29	56.60	99.54	54.48	77.58	71.46	88.55
	$1/c = 100$	55.18	76.00	78.80	99.61	55.35	82.24	58.82	82.57
metapath2vec		65.26	90.68	**80.52**	99.72	56.14	78.24	69.38	84.79
PTE		57.72	77.51	50.32	68.84	57.89	78.23	70.36	89.54
HIN2Vec		53.29	75.47	51.64	66.71	58.11	81.65	69.68	84.48
AspEm		**67.21**	**91.46**	76.10	95.18	55.80	77.70	68.31	87.43
HEER		53.00	72.76	73.72	95.92	55.78	78.31	69.06	88.42
R-GCN		50.50	73.35	72.17	97.46	50.18	74.01	63.33	81.19
HAN		50.24	73.10	-	-	51.50	74.13	65.85	85.33
MAGNN		50.10	73.26	50.03	69.81	50.12	74.18	61.11	90.01
HGT		59.98	83.13	79.00	99.66	55.68	79.46	73.00	88.05
TransE		63.53	86.29	69.13	83.66	52.84	75.80	67.95	84.69
DistMult		52.87	74.84	80.28	**99.73**	54.91	78.04	70.61	90.64
ComplEx		65.92	90.01	80.11	**99.73**	**60.43**	**84.22**	**75.96**	**92.47**
ConvE		54.03	75.31	78.55	99.70	54.29	76.11	71.81	89.82



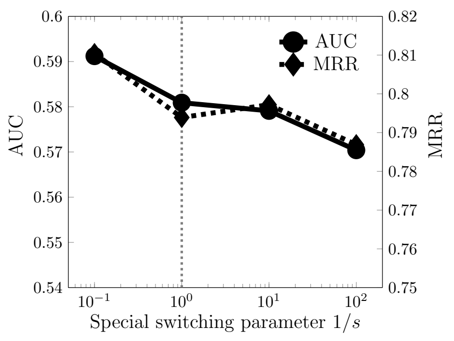
(a) Freebase generic



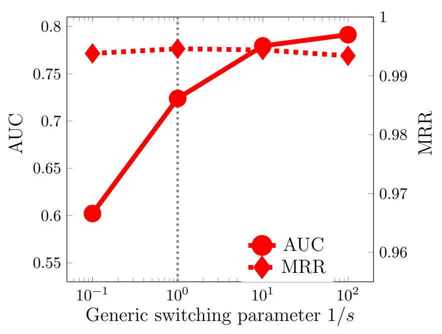
(b) Freebase special



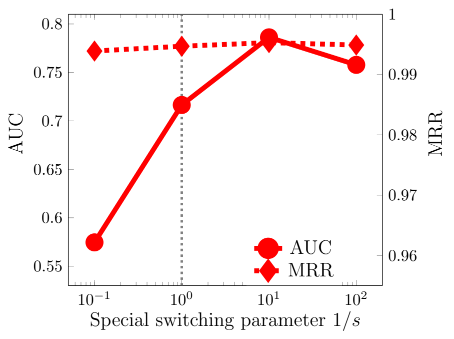
(c) DBLP generic



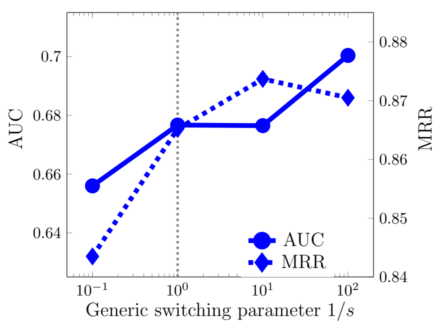
(d) DBLP special



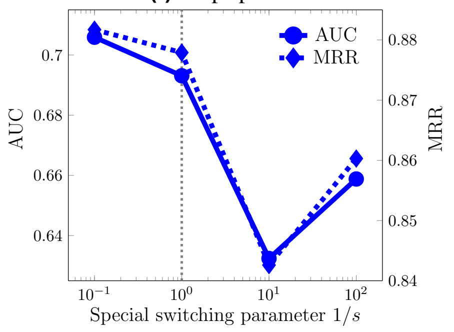
(e) Yelp generic



(f) Yelp special



(g) Pubmed generic



(h) Pubmed special

Figure 12. Edge prediction: AUC (left axis - continuous line) and MRR (right-axis - dashed line) values obtained for varying values of the γ_c parameter. Plots a,c,e,g in the first column depict the variation of the performance metrics when the generic switching strategy is used; the second column (subplots b,d,f,h) shows the results obtained when using the special switching strategy. In each plot, different scales are used for the left (AUC) and right (MRR) axis. The plots referring to the same graph use the same AUC and MRR scales to allow a comparison between the generic and the special node-type switching strategy.

Table 5. Performance metric for node label prediction and edge prediction tasks on the benchmark graphs using a Random Forest as the predictive model trained in the *Het-node2vec* graph representation.

		Node label prediction (Macro-F1/Micro-F1)							
		DBLP		Yelp		Freebase		Pubmed	
HetNode2vec generic	$1/s = 0.1$	24.26	38.19	5.16	23.53	43.12	60.06	9.75	15.64
HetNode2vec generic	$1/s = 100$	28.24	39.96	6.66	25.78	37.03	57.24	12.44	18.95
HetNode2vec special	$1/s = 0.1$	29.50	42.06	5.26	23.71	35.50	56.12	13.33	18.72
HetNode2vec special	$1/s = 100$	33.26	46.28	6.16	24.65	47.44	62.65	9.18	14.76

		Edge prediction (AUC/MRR)							
		DBLP		Yelp		Freebase		Pubmed	
HetNode2vec generic	$1/s = 0.1$	86.58	99.06	76.52	99.47	84.08	97.59	81.92	98.10
HetNode2vec generic	$1/s = 100$	84.82	98.99	80.60	99.42	86.59	97.69	84.10	98.78
HetNode2vec special	$1/s = 0.1$	86.93	99.08	76.21	99.58	84.65	97.49	84.12	98.51
HetNode2vec special	$1/s = 100$	86.31	99.16	79.62	99.56	85.49	97.80	81.63	98.47

11 RNA-KG node type distribution

Type	Count	Proportion	Mean degree
Chemical	199178	0.29559	3.86
Protein	122741	0.18216	9.08
GO	42417	0.06295	14.83
Cell	40840	0.06061	3.68
Disease	25150	0.03732	13.95
piRNA	24048	0.03569	6.02
Gene	20791	0.03086	62.66
tRF	20744	0.03079	13.43
mRNA	20632	0.03062	185.09
Phenotype	19188	0.02848	7.68
Riboswitch	18471	0.02741	3.80
lncRNA	17578	0.02609	163.02
Anatomy	14489	0.02150	8.22
Variant (SNP)	11950	0.01773	5.26
tsRNA	10254	0.01522	15.29
Pseudogene	9846	0.01461	27.12
circRNA	9148	0.01358	46.81
Small protein	8050	0.01195	4.06
Vaccine	7174	0.01065	5.68
Viral RNA	5654	0.00839	2.04
miRNA	4634	0.00688	739.17
Pathway	3846	0.00571	9.71
Species	3204	0.00475	43.56
ASO	2633	0.00391	3.02
Genomic feature	2391	0.00355	107.27
snRNA	1820	0.00270	16.72
snoRNA	916	0.00136	23.63
Glycan	797	0.00118	11.75
tRNA	620	0.00092	376.97
Other RNA	571	0.00085	22.61
Environment	453	0.00067	4.86
Environmental exposure	419	0.00062	3.76
rRNA	357	0.00053	5.70
Chromosome	310	0.00046	5.18
Plant	267	0.00040	4.55
ncRNA	197	0.00029	8.33

Type	Count	Proportion	Mean degree
Aptamer	184	0.00027	3.25
Food	181	0.00027	2.86
Parasite lifecycle	180	0.00027	6.55
Neuro behaviour	160	0.00024	3.86
Trait	150	0.00022	3.62
Protein modification	145	0.00022	226.79
s(i/h)RNA	117	0.00017	2.00
Medical action	114	0.00017	3.79
NCI thesaurus	85	0.00013	3.22
Non-RNA drug	85	0.00013	17.21
Mouse pathology	75	0.00011	10.55
Human developmental stage	72	0.00011	5.72
gRNA	72	0.00011	2.00
Dictyostelium discoideum anatomy	65	0.00010	5.45
Experimental factor	56	0.00008	87.98
tRNA nucleotide editing	54	0.00008	26.81
Viral mRNA	40	0.00006	8.00
Fungal anatomy	33	0.00005	2.82
RNA drug	25	0.00004	6.64
scaRNA	24	0.00004	24.00
lincRNA	22	0.00003	3.55
Ribozyme	18	0.00003	339.11
Mental functioning	14	0.00002	3.00
Mental disease	12	0.00002	2.92
mtRNA	10	0.00001	5.80
Unclassified RNA	8	0.00001	14.25
scRNA	6	0.00001	21.83
Infectious disease	6	0.00001	2.17
General medical science	6	0.00001	27.17
Retained intron	6	0.00001	6.17
Y RNA	4	0.00001	3.00
Adverse events	4	0.00001	1.75
Vault RNA	3	0.00000	5.33
Biological role	3	0.00000	58.67
eRNA	3	0.00000	2.67
miscRNA	2	0.00000	2.00
TEC	2	0.00000	2.00
Basic formal	1	0.00000	1141.00

12 RNA-KG hyperparameters

Table 6. Hyperparameters of the methods used in the construction of the embedding and the evaluation in RNA-KG graph

	hyperparameter	value
RW generation	Number of RWs (iterations)	128
	$1/p$	0.25
	$1/q$	4
	Walk length	16
Skipgram	Embedding size	25
	Epochs	10
	Ratio neg:pos samples	10:1
	Window size	5
	Max neighbours	100
	Learning rate	0.01
	Learning rate decay	0.9
Random Forest	Number of estimators	100
	criterion	gini
	min samples	1
	max depth	100
	max iter	1000
	max features	sqrt

References

1. 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.

2. 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).

3. Cavalleri, E. *et al.* An ontology-based knowledge graph for representing interactions involving RNA molecules. *Sci. Data* **11** (2024).

4. 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).

5. 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).

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

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

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

9. 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).

10. 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).

11. 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).

12. 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).

13. 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).

14. 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).

15. 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).

16. 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).

17. 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).

18. 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).

19. Zitnik, M. & Leskovec, J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* **33**, i190–i198, DOI: 10.1093/bioinformatics/btx252 (2017). https://academic.oup.com/bioinformatics/article-pdf/33/14/i190/25157097/btx252.pdf.

20. 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).

21. Park, C., Kim, D., Zhu, Q., Han, J. & Yu, H. Task-guided pair embedding in heterogeneous network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, 489–498, DOI: 10.1145/3357384.3357982 (Association for Computing Machinery, New York, NY, USA, 2019).

22. He, Y. *et al.* Hetespaceywalk: A heterogeneous spacey random walk for heterogeneous information network embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, 639–648, DOI: 10.1145/3357384.3358061 (Association for Computing Machinery, New York, NY, USA, 2019).

23. Wu, Z. *et al.* A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks Learn. Syst.* 1–21, DOI: 10.1109/TNNLS.2020.2978386 (2020).

24. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 1263–1272 (JMLR.org, 2017).

25. 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).

26. Zhang, C., Song, D., Huang, C., Swami, A. & Chawla, N. V. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, 793–803, DOI: 10.1145/3292500.3330961 (Association for Computing Machinery, New York, NY, USA, 2019).

27. Zitnik, M., Agrawal, M. & Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* **34**, i457–i466, DOI: 10.1093/bioinformatics/bty294 (2018). https://academic.oup.com/bioinformatics/article-pdf/34/13/i457/25098429/bty294.pdf.

28. 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).

29. Chen, C., Yang, X., Zhou, J., Li, X. & Song, L. Heterogeneous graph neural networks for malicious account detection. In *Proc. of CIKM*, 2077–2085, DOI: 10.1145/3269206.3272010 (2018).

30. 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).

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.