Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Network Embedding based on Community- aware and Random Walk for Community Detection in Attributed Networks

Shuo Gao
Harbin Institute of Petroleum
Qi Zhen
9c5c@163.com

Harbin Institute of Petroleum

Research Article

Keywords: random walk, community structure, network embedding, community detection, attributed network

Posted Date: June 23rd, 2025

DOI: https://doi.org/10.21203/rs.3.rs-6766789/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Network Embedding based on Community-aware and Random Walk for Community Detection in Attributed Networks

Shuo Gao, Qi Zhen,*

College of Intelligent Engineering, Harbin Institute of Petroleum, Harbin, 150028, Heilongjiang, China.

*Corresponding author(s). E-mail(s): 9c5c@163.com;

Abstract： Community detection aims to find a collection of closely connected nodes in a network, which is an important problem in network analysis. In recent years, network embedding techniques based on random walks have been widely applied to community detection tasks. However, these methods suffer from problems such as ignoring node attributes and locations, as well as the tediousness of manually setting the walking parameters. To address these issues, this paper proposes a network embedding based on community-aware and random walk for community detection in attributed networks (RWACD). Firstly, we calculate the node topology and attribute similarity to construct the fusion similarity matrix. Secondly, we construct transition probability matrices biased towards community boundary and internal walking, respectively, to enhance nodes' walking inside and at the boundary of the community and to better recognize the community structure. Next, we set the number and length of walking according to the node degree and the number of attributes to simplify the process of setting the walking parameters under the premise of ensuring the accuracy of the algorithm and avoiding the problems of overfitting caused by manually setting the walking parameters. Finally, we get the embedding vectors of nodes by Skip-Gram model and apply k-means clustering to get the community detection results. The effectiveness of the proposed algorithm is verified by conducting experiments on real-world and synthetic datasets.

Keywords： random walk; community structure; network embedding; community detection; attributed network

1. Introduction

Networks, as a common form of data in the real world, play a crucial role in the study of relationships between entities. In reality, the connections between entities in many domains can be modelled as networks and contain rich attribute information, such as citation networks, protein networks, social networks, etc., which are called attribute networks[1][2]. For attribute networks, it is

not enough to use the topological information of the nodes; the rich attribute information contained in the nodes is also crucial for community detection[3][4]. Community detection in attribute networks aims to exploit the community structure in the network[5], so that nodes within the same community are not only tightly connected structurally, but also have similar attributes, while nodes in different communities are sparsely connected structurally and have dissimilar attributes[6][7]. How to effectively use the topological and attribute information of nodes for community detection of attribute networks is currently a popular research.

In recent years, the network embedding technique[8] has been widely used in community detection tasks to achieve community delineation by clustering the embedded vectors, with the advantage that not only the topological information of the nodes but also the node attributes can be used to embed the high-dimensional sparse information of the nodes into the low-dimensional dense vector space and preserve the original topological and attribute features of the nodes[9]. Among them, random walk-based network embedding methods have been shown to be effective in community detection tasks. For example, the DeepWalk algorithm proposed by Perozzi et al[10] pioneered the use of random walking to generate a corpus, obtain the embedding vectors of the nodes through the Skip-Gram[11] model, and combine it with a clustering algorithm to obtain community detection results. However, some of these random walking methods only use the topological information of the nodes, ignoring the importance of the node attribute information, and some do not consider maintaining the community structure of the nodes for the community detection task at the time of embedding, which may lead to inaccurate community detection results.

To address the above problems, this paper proposes a network embedding based on community-aware and random walk for community detection in attributed networks (RWACD). First, the topological similarity and attribute similarity of the nodes are calculated to construct the fusion similarity matrix; then, the centrality and boundaries of the nodes are considered in constructing the transition probability matrix using the fusion similarity matrix, and the transition probability matrix for nodes with a bias towards walking within the community and a bias towards walking at the community boundaries are constructed. Finally, the number of walks and the length of walks for the nodes are set according to the degree and the number of attributes of the nodes, so as to simplify the process of setting the walking parameters under the premise of guaranteeing the accuracy of the algorithm, and to avoid the overfitting and oversampling of nodes caused by manually setting the walking parameters. Through the above strategy, high cohesive community detection results are obtained.

The main contributions of this paper are as follows:

1. Our algorithm exploits both topological and attribute information of nodes and makes nodes highlight the interior and boundaries of communities during random walking, which can identify communities more accurately compared to other random walking algorithms.

2. Our algorithm sets the number of walks and the length of walks for nodes based on their degrees and attribute counts, which can simplify the process of manually setting walk parameters while avoiding overfitting and over-sampling issues caused by fixed parameters.

3. We have conducted a large number of experiments on real-world and synthetic datasets to demonstrate the superiority of the present algorithm in community detection tasks.

2. Related Work

2.1 Traditional Community Detection Methods

Traditional community detection methods include label propagation, seed expansion, spectral clustering, graph segmentation, modularity optimisation, etc., and the representative algorithms are GN (Girvan-Newman)[12], LPA (Label Propagation Algorithm)[13], FN (Fast Newman)[14], CPM (Cluster Percolation Method)[15] and Louvai[16], etc. Ding et al[17] heuristically considered the effect of low quality seeds and communities on the expansion process and proposed to discard low quality seeds and communities during the expansion process, and finally obtained the community detection results through the optimisation process. Zhao et al[18] Reduced the graph size by compressing nodes with degree 1 or 2 to height nodes and filtered the seed nodes using quality function and node density and obtained community detection results through seed expansion.Berahmand K et al[19] constructed affinity matrix using topological similarity and attribute similarity of nodes through a parameter free method and obtained community detection results through spectral clustering. Zhe et al[20] mapped attributes into nodes through node-attribute relationship and added edges between two types of nodes for community detection of augmented graph through modularity maximisation model. Meng et al[21] proposed a coupled similarity calculation method to capture node-to-node, node-to-attribute and attribute-to-attribute similarity to generate the weights of edges between nodes and obtained community detection of weighted graph through a clustering algorithm delineation. Berahmand K et al[2] added weights to edges through topological similarity and

attribute similarity and calculated node influence using Laplace centrality to improve the selection step of the label propagation process.

2.2 Network Embedding-based Community Detection Methods

Existing network embedding techniques include random walk, matrix decomposition and deep learning, and community detection is achieved by combining them with clustering algorithms.

Among the matrix decomposition-based methods, Huang et al[22] proposed an attribute network embedding framework to jointly characterise topological and attribute information as embedding vectors, and proposed a distributed algorithm to speed up the operations of matrix decomposition. Xu et al[1] proposed the CDANE algorithm to jointly model the attribute and structural similarities of nodes, obtain node embedding vectors through matrix decomposition, and obtain community delineation results through non-parametric clustering. parametric clustering to obtain community detection results. Ye et al[23] proposed a new matrix updating method to deal with the optimisation problem of non-negative matrix decomposition, which uses the non-negative matrix decomposition framework to identify the community structure in the network. Gao et al[24] used the LDA topic model to generate the initial community to guide the random walking, and to control the walking bias towards the inside of the community. Lu et al[25] proposed the topology and attribute combination of non-negative matrix decomposition and non-negative matrix decomposition with topology and attribute symmetry, two non-parametric models to divide the network into communities. Wang et al[26] proposed the M-NMF model to maintain the community structure of the nodes by adding modularity constraint terms to their objectives.

Among the deep learning based approaches, Huang et al[27] proposed a deep neural network based attribute network embedding framework that preserves the community structure by combining the network topology, node attributes and community structure to obtain a vector representation of the nodes in a low-dimensional space. Jin et al[28] exploited the salient features of graph convolutional networks and Markov random fields in a complete end-to-end deep network architecture to solve the semi-supervised community detection problem in attribute networks with semantic information. Guo et al.[29] proposed the AWBA (Attentional-Walk-Based AutoEncoder) algorithm, which learns the attention coefficients through the encoder, thus guiding the nodes in the decoder to randomly walk to neighbouring nodes and capture the relationships between high and low order nodes. Cui et al.[30] used a Laplace smoothing filter to mitigate high frequency noise in the node features, followed by an adaptive encoder

that iteratively enhances the filtered features, which can be better embedded in the nodes for better application in downstream tasks such as community detection.

Among the random walk-based methods, the LINE (Large-scale Information Network Embedding) algorithm proposed by Tang et al[31] improves the embedding effect by optimising the first-order similarity and second-order similarity of the nodes. The Node2Vec algorithm proposed by Grover et al[32] is based on DeepWalk by controlling the walking direction and sampling strategy by parameters. Zhang et al[33] proposed DiaRW algorithm designs backtracking strategy and variable length strategy during walking to reduce redundant information considering the scale-free nature of the network. Sarmad N et al[34] set the number of walking times for the nodes proportionally according to the degree of the nodes to reduce the amount of computation and unbalanced data. Berahmand K et al[35] proposed the CSADW algorithm, which uses the topological similarity and attribute similarity between the nodes to construct the weight matrix and obtain the transmission probability matrix for random walking by normalisation. Wei et al[36] proposed the IANRW algorithm, which defines the topological neighbours and attribute neighbours of the nodes and controls the walking between the two types of nodes by parameters. Yang et al[37] proposed the TADW (text-associated DeepWalk) algorithm, proving that the DeepWalk algorithm is equivalent to matrix decomposition and extending it to perform supervised random walking using text attributes of nodes. Dou et al[38] perform walking on structural and attribute bipartite graphs to maintain network structure and attribute homogeneity respectively.

Some of the above methods only use the topological information of the nodes, some require extensive experimentation to determine the walking parameters, and some do not take into account the community structure of the nodes, resulting in inaccurate delineation of the community structure for various reasons. Unlike these methods, our method uses both topological and attribute information of the nodes, emphasises the community structure during random walking, and eliminates the tedious process of determining the walking parameters, allowing for more accurate identification of the community structure.

3. Proposed Algorithm

3.1 Related Descriptions

For an undirected attribute graph, it can be represented as $G=(V,E,A,F)$ where $V=\{V_1,V_2,...,V_n\}$ denotes the set of nodes in the network, and $\mathrm{E}=\{(\mathrm{v}_i,\mathrm{v}_j)|\mathrm{v}_i,\mathrm{v}_j\in\mathrm{V}\}$ denotes the set of edges between nodes in the network. $A\in R^{n\times n}$ denotes the adjacency matrix of the network, if there is an edge between nodes v_i and v_j, then $A_{ij}=1$, otherwise $A_{ij}=0$.

$\mathrm{F}=\left[F_1, F_2, \ldots, F_n\right]^T \in \mathrm{R}^{\mathrm{n} \times \mathrm{m}}$ is an attribute matrix, if node v_i has attribute f_j then $F_{ij}=1$, otherwise $F_{ij}=0$. Where n is the number of nodes in the network and m is the dimension of the attribute vector of the nodes in the network. The main symbols used in this paper and their meanings are listed in Table 1 below.

The framework of our algorithm is shown in Figure 1. First, the fusion similarity matrix is constructed by calculating the topological similarity matrix and the attribute similarity matrix of the attribute network. Next, the transition probability matrix for biased community boundary and internal walking are constructed through the fusion similarity matrix, respectively. Then, the number and length of walkings for nodes are automatically determined based on node degree and attribute number, boundary nodes and internal nodes are classified based on point-degree centrality, and walkings are sampled based on the corresponding transition probability matrices. Finally, the embedding vectors of the nodes are learned by the Skip-Gram model, and clustering is performed using k-means to obtain the community detection results.

Table1 Key symbols and meanings

Symbols	Meanings
G	Attributed network
V	Nodes set
E	Edges set
$A \in R^{n \times n}$	Adjacency matrix
$F \in R^{n \times m}$	Attributed matrix
n	Number of nodes
m	Number of attributes
d	Embedding vectors dimension
$\alpha \in [0,1]$	Scaling parameter for control matrix fusion
$\beta \in [0,1]$	Scale parameter for delineating interior and boundary nodes
Num	Number of walks of the node
$Length$	Length of walks of the node
$L_{\max}$	Maximum walk length of the node
$N(v_i)$	Set of node neighbours
$D(v_i)$	Degree of node
$S(v_i)$	Number of attributes of node
$\lvert\cdot\rvert$	Number of elements in the set
$TSim \in R^{n \times n}$	Topological similarity matrix
$ASim \in R^{n \times n}$	Attribute similarity matrix
$W \in R^{n \times n}$	Fusion similarity matrix
$W_{inside} \in R^{n \times n}$	Transition probability matrix for biased community internal walking
$W_{boundary} \in R^{n \times n}$	Transition probability matrix for biased community boundary walking

3.2 Relevant Definitions

(Definition 1 Set of node neighbours)The set containing all nodes connected to node v_i is defined as:

$$N(v_i)=\{v_j : v_j \in V,(v_i,v_j)\in E\} \tag{1}$$

Where v_i and v_j denote the nodes in the network, V denotes the set of nodes, E denotes the set of edges between nodes and $N(v_i)$ denotes the set of neighbours of node v_i.

(Definition 2 Degree of node)The degree of node v_i denotes the number of nodes connected to node v_i, that is, the number of neighbours of node v_i, defined as:

$$D(v_i)=|N(v_i)| \tag{2}$$

Where $|N(v_i)|$ denotes the number of elements in the set and $D(v_i)$ denotes the degree of the node.

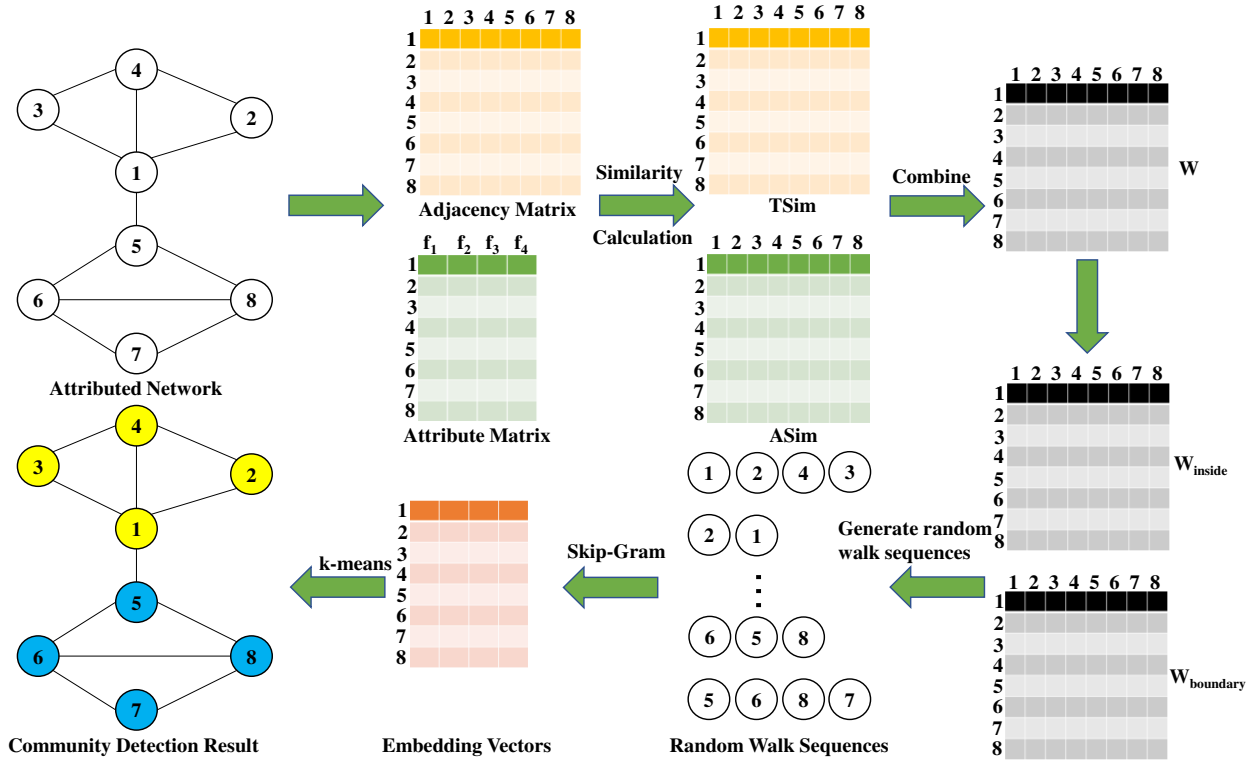


Figure 1 Framework of our algorithm

(Definition 3 Topological similarity calculation)The adjacency matrix of nodes only contains the connection relationship between nodes, which cannot distinguish the similarity between nodes, and it is difficult to achieve effective use of information. By mining the similarity between nodes, more and more effective information can be obtained. There are many methods to calculate the similarity of nodes, such as Jaccard similarity, cosine similarity, etc., among which the calculation of Jaccard similarity is shown in Equation 3 below:

$$Jaccard\left(v_i, v_j\right) = \frac{\left|N\left(v_i\right) \cap N\left(v_j\right)\right|}{\left|N\left(v_i\right) \cup N\left(v_j\right)\right|} \tag{3}$$

Where $N(v_i)$ and $N(v_j)$ denote the neighbours of nodes v_i and v_j respectively, $\left|N\left(v_i\right) \cap N\left(v_j\right)\right|$ denotes the number of elements in the intersection of the sets of neighbours of nodes v_i and v_j respectively, and $\left|N\left(v_i\right) \cup N\left(v_j\right)\right|$ denotes the number of elements in the concatenation of the sets of neighbours of nodes v_i and v_j respectively. A higher value of $Jaccard(v_i, v_j)$ denotes a higher degree of similarity between the two nodes.

However, when calculating the Jaccard similarity, all nodes are treated equally and the effect of a node's degree on the similarity is not considered. Considering that nodes with higher degree tend to be in a more important position, the higher the degree of a node in the common neighbourhood, the more it contributes to the similarity calculation of the two nodes. Accordingly, we propose a node degree based Jaccard similarity calculation as shown in Equation 4 below:

$$Degree_Jaccard\left(v_i, v_j\right) = \frac{\sum_{v_k \in N(v_i) \cap N\left(v_j\right)} D\left(v_k\right)}{\sum_{v_l \in N(v_i) \cup N\left(v_j\right)} D\left(v_l\right)} \tag{4}$$

Where v_k denotes the node in the intersection set of nodes v_i and v_j neighbour sets and v_l denotes the node in the concatenation set of nodes v_i and v_j neighbour sets. By considering the effect of node degree on similarity, the similarity between nodes can be better expressed.

(Definition 4 Topological similarity matrix)The topological similarity between nodes is calculated by equation 4, and the topological similarity matrix $TSim$ of the nodes can be obtained by equation5:

$$TSim(i, j) = \begin{cases} Degree_Jaccard\left(v_i, v_j\right), & if \quad N\left(v_i\right) \cap N\left(v_j\right) \neq \varnothing \\ 0, & otherwise \end{cases} \tag{5}$$

(Definition 5 Attribute similarity calculation)Cosine similarity is very effective in calculating the similarity between vectors considering only non-zero values, therefore we use cosine similarity to measure the similarity between node attribute vectors, which is calculated as shown in Equation 6 below:

$$CosinSim\left(v_i, v_j\right) = \frac{\sum_{t=1}^{m} F_{it} F_{jt}}{\sqrt{\sum_{t=1}^{m} \left(F_{it}\right)^2} \sqrt{\sum_{t=1}^{m} \left(F_{jt}\right)^2}} \tag{6}$$

Where m denotes the dimension of the attribute vector, F_{it}, F_{jt} denote the value of the $t-th$ element in the attribute vector of nodes v_i and v_j.

(Definition 6 Attribute similarity matrix)The cosine similarity between the attribute vectors of the nodes is calculated by equation 6, and the attribute similarity matrix $ASim$ of the nodes can be obtained by equation7:

$$ASim(i,j)=\begin{cases} CosinSim(v_i,v_j), & if \quad v_i,v_j \in V \\ 0, & otherwise \end{cases} \tag{7}$$

(Definition 7 Fusion similarity matrix)In order to make full use of the topological and attribute information of the nodes and to improve the accuracy of the node similarity measure in the attribute network, we fused the topological similarity matrix $TSim$ and the attribute similarity matrix $ASim$ of the nodes and controlled the weights by the parameter α to obtain the fused similarity matrix W, which is calculated as shown in Eq. 8 below:

$$W=\alpha * TSim+(1-\alpha)* ASim \tag{8}$$

Where the parameter $\alpha \in [0,1]$ is used to control the contribution of the node topology and attribute similarity matrices, and $TSim$ and $ASim$ denote the topology similarity matrix and attribute similarity matrix of the attribute network. The fusion similarity matrix W constructed by Equation 8 can improve the accuracy of node similarity measurement in attribute networks, while making full use of the node topology and attribute information.

3.3 Construction of Transition Probability Matrix with Community Awareness

The random walking approach based on the adjacency matrix considers each node in the network equally, and nodes in different locations adopt the same sampling strategy, which has a high degree of randomness in walking, and the generated walking sequences do not capture the community characteristics of the network well, resulting in the embedding vectors of nodes performing poorly in the community detection task. According to the degree centrality, it can be seen that nodes located in the interior of the network have a higher degree, while the degree of the boundary nodes is relatively small, and such a property usually applies to the interior and boundary of the community, nodes in the interior of the community have a higher degree, and nodes at the boundary have a lower degree. Also, nodes with higher common similarity are more likely to belong to the same community. Therefore, based on the joint similarity degree of nodes, we propose a way to construct a transition probability matrix that is biased towards walking inside the community and biased towards walking at the community boundary, so that nodes are community-aware when walking and can identify the community more accurately.

3.3.1 Construction of Transition Probability Matrix for biased Community Internal Walking

(Definition 8 Transition probability of biased internal walking)The probability of an internal node v_i jumping to node v_j is calculated as shown in equation 9:

$$P_{inside}\left(v_j|v_i\right)=W_{ij}*D\left(v_j\right) \tag{9}$$

Where $P_{inside}(v_j \mid v_i)$ is the probability that node v_i jumps to node v_j, W_{ij} is the fusion similarity of nodes v_i and v_j, and $D(v_j)$ is the degree of the target node of the jump. Nodes within a community have a higher degree, and there is a higher fusion similarity between nodes within the same community. Therefore, our method allows higher transition probabilities between internal nodes that may belong to the same community, and achieves the goal of encouraging internal community walking and better identifying the community interior.

(Definition 9 Transition probability matrix for biased community internal walking)The normalised transition probability matrix for biased community internal walking can be obtained by equation 10 and is calculated as follows:

$$W_{inside}\left(i,j\right)=\begin{cases}\dfrac{P_{inside}\left(v_j|v_i\right)}{\sum_{k=0}^{n}P_{inside}\left(v_k|v_i\right)}, & if \quad W_{ik}\neq 0\\ 0, & otherwise\end{cases} \tag{10}$$

W_{inside} is a transition probability matrix obtained by fusing the topological similarity of nodes, attribute similarity and the degree of a node to favour intra-community walking, which allows walking to occur with a higher probability between nodes within the community, identifying intra-community.

3.3.2 Construction of Transition Probability Matrix for biased Community Boundary Walking

(Definition 10 Transition probability of biased boundary walking)The probability of an boundary node v_i jumping to node v_j is calculated as shown in equation 11:

$$P_{boundary}\left(v_j|v_i\right)=\frac{W_{ij}}{D\left(v_j\right)} \tag{11}$$

Where $P_{boundary}(v_j \mid v_i)$ is the probability of node v_i jumping to node v_j, W_{ij} is the fusion similarity of nodes v_i and v_j, and $D(v_j)$ is the degree of the target node of the jump. Community boundary nodes have smaller degrees, but nodes in the same community have higher fusion similarity between them. Therefore, our method allows higher transition probabilities between boundary nodes that may belong to the same community to achieve the goal of encouraging community boundary walking and

better identifying community boundaries.

(Definition 11 Transition probability matrix for biased community boundary walking)The normalised transition probability matrix for biased community boundary walking can be obtained through Equation 12 and is calculated as follows:

$$W_{boundary}\left(i,j\right)=\begin{cases}\dfrac{P_{boundary}\left(v_j|v_i\right)}{\sum_{k=0}^{n}P_{boundary}\left(v_k|v_i\right)}, & if \quad W_{ik}\neq 0\\ 0, & otherwise\end{cases} \tag{12}$$

$W_{boundary}$ is a biased community boundary transition probability matrix obtained by fusing node topological similarity, attribute similarity and node degree, which allows walking with higher probability between nodes at the community boundary, identifying the community boundary.

3.4 Calculation of Walking Parameters and Sampling Strategy

Traditional methods use parametric experiments to determine the walking parameters, which not only increases the degree of experimental tedium, but can also lead to overfitting problems. Moreover, these methods set uniform walking parameters for all nodes, which may lead to oversampling of low degree nodes, thus generating a large amount of redundant information and causing problems such as poor embedding effect. Therefore, we propose a method to set the number of walking times and walking lengths for nodes based on the degree and number of attributes, which are calculated as shown in Equation 13 and Equation 14:

$$Num=\max\left(D(v_i),S(v_i)\right) \tag{13}$$

$$Length=\min\left(\frac{D\left(v_i\right)+S\left(v_i\right)}{2},L_{max}\right) \tag{14}$$

Where Num is the number of node walks times, $Length$ is the walking length of the node, $D(v_i)$ is the degree of the node, $S(v_i)$ is the number of attributes contained in the node, "⌈⌉" is an upward rounding symbol, and $L_{\max}$ is the maximum walking length. Compared to the unified walking parameters, our proposed method of calculating the walking parameters ensures full sampling of nodes with larger degrees and number of attributes, and avoids the oversampling problem for nodes with lower degrees and fewer attributes.

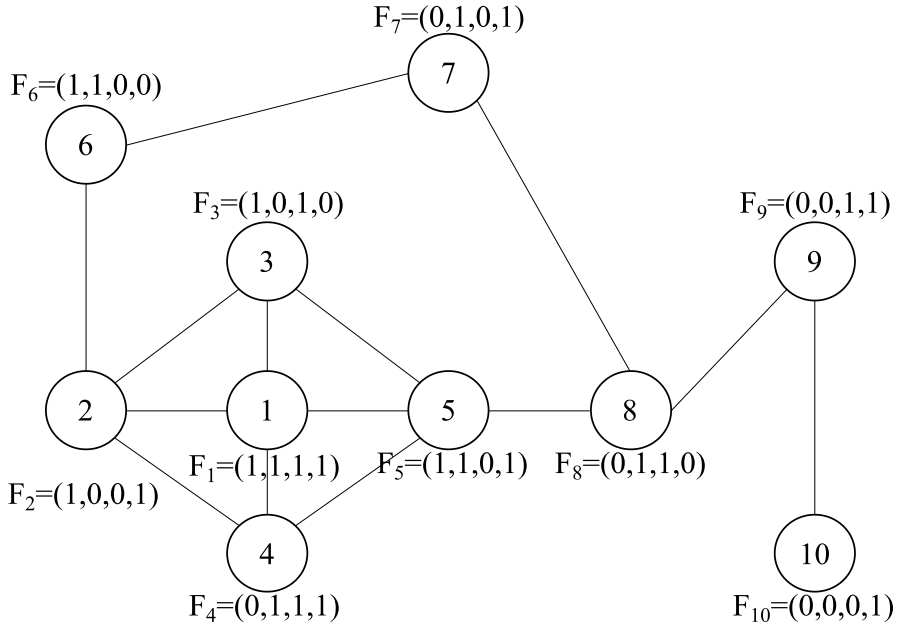


Figure2 Attributed network

When dividing the boundary nodes and internal nodes, we sort the nodes by degree from largest to smallest based on the point-degree centrality and slice the list by the parameter $\beta \in [0,1]$, and take the first $n \times \beta$ nodes as the internal nodes of the community, which are subjected to walking sampling through the biased internal walking transition probability matrix. The remaining nodes are treated as boundary nodes and are sampled by walking through the biased boundary walking transition probability matrix. When a node is walking, according to the corresponding transition probability matrix, the nodes with transition probability with the current node are sorted in ascending order of transition probability, from which the nodes in the top tenth of nodes are taken out as the set of nodes to be jumped, and a node is randomly selected from the set to join the walking sequence as the next walking node until the length of the sequence reaches the end of the corresponding walking length.

Figure 2 is used as an example to illustrate the process of walking sequence generation. First, the degree of each node is calculated and sorted in descending order to obtain a set of nodes [1,2,5,3,4,8,6,7,9,10], and then the set is divided to obtain [1,2,5,3,4,8] as internal nodes and [6,7,9,10] as boundary nodes, with specific values shown in Table 2. For node 1, the number of walkings can be calculated as 4 and the length of walkings can be calculated as 4 by Eqs. 13 and 14, and then according to the internal transition probability matrix obtained from Eqs. 9 and 10 for walking sampling, one of the walking sequences can be obtained as 1→5→4→2. For node 9, the number of walkings can be calculated as 4 and the length of the walkings can be calculated as 4 by Eqs. 13 and 14, the number of walkings can be calculated as 2 and the length of the walkings can be calculated as 2. Then

the walkings are sampled according to the boundary transition probability matrix obtained by Eqs. 11 and 12, and one of the walking sequences can be obtained as 9→10. And so on until all the walking sequences are obtained. The sequences generated by this walking method can maintain the boundary and internal structural characteristics of the community nodes, as well as make the nodes in the sequences more similar.

Table 2 Walking parameters corresponding to the nodes in Figure 2

Node	Degree	Number of attributes	Number of walks	Length of walks
1	4	4	4	4
2	4	2	4	3
3	3	2	3	3
4	3	3	3	3
5	4	3	4	4
6	2	2	2	2
7	2	2	2	2
8	3	2	3	3
9	2	2	2	2
10	1	1	1	1

3.5 Generation of Embedding Vectors and Communities

Network embedding aims to generate d-dimensional embedding vectors by learning a mapping function $\Phi : V \to R^d \left(d \square \ |V| \right)$, while preserving the original features of the nodes. Skip-Gram is a word embedding model whose core idea is to learn a vector representation through a central word prediction context that maximises the likelihood probabilities between nodes in the same window. Here, we take the walking sequences generated by random walking as a corpus and the nodes as words, and learn the embedding vectors of the nodes through the Skip-Gram model with the objective function:

$$argmax \sum_{v_i \in V} \log \Pr\left(N_s\left(v_i\right) | \Phi\left(v_i\right) \right) \tag{15}$$

Where $\Phi(v_i)$ is the embedding vector of node v_i, and $N_s(v_i)$ is the neighbour node of node v_i in the walking sequence. Assuming that the nodes in the context are independent of each other, the conditional probability of the above equation can be approximated as:

$$\Pr\left(N_s\left(v_i\right) | \Phi\left(v_i\right) \right) = \prod_{n_i \in N_s\left(v_i\right)} \Pr\left(n_i | \Phi\left(v_i\right) \right) \tag{16}$$

The conditional likelihood of the neighbouring nodes of the central node using the soft-max function is modelled as:

$$\Pr(n_i \mid \Phi\left(v_i\right)) = \frac{\exp\left(\Phi\left(n_i\right) \cdot \Phi\left(v_i\right) \right)}{\sum_{v_j \in V} \exp(\Phi\left(v_j\right) \cdot \Phi\left(v_i\right))} \tag{17}$$

Finally, for the node embedding vectors generated by the Skip-Gram model, clustering is performed by the k-means algorithm to obtain the final community detection results.

3.6 Algorithm Description

Table 3 RWACD algorithm description

Algorithm RWACD
Input: Attributed network $G=(V,E,A,F)$, α, β, d, $L_{\max}$, number of communities k, windows-size w.
Output: Community detection results C.
1. Calculate the topological similarity of nodes and the topological similarity matrix according to Eqs. (4) and (5).
2. Calculate the cosine similarity of node attribute vectors and the attribute similarity matrix according to Eqs. (6) and (7).
3. Construct the fusing similarity matrix according to Eqs. (8).
4. Construct the normalized transition probability matrix biased towards internal walks within communities according to Eqs. (9) and (10).
5. Construct the normalized transition probability matrix biased towards boundary walks between communities according to Eqs. (11) and (12).
6. Calculate the number of walks and the length of walks for each node according to Eqs. (13) and (14).
7. Sort the nodes by degree centrality in descending order, divide the internal nodes and boundary nodes based on the proportion, and sample walks based on the corresponding transition probability matrix.
8. Embed the walk sequences using the Skip-Gram model to obtain the embedding vectors of nodes, and cluster the vectors using k-means algorithm to obtain community detection results.
9. **return** C**.**

4. Experiments

In this section, we evaluate the effectiveness of the algorithm by conducting experiments on real-world and synthetic datasets using three evaluation metrics. First, we investigate the effect of the parameters α, β and d on the accuracy of the algorithm. Then, we compare the accuracy of the algorithm in this paper with six baseline algorithms, Louvain[16], DeepWalk[10], LINE[31], Node2Vec[32], AANE[22], and AWBA[29], on the community detection task. All simulation experiments were performed using Python 3.8 on a host computer with an AMD R7 5800H 3.20 GHz CPU, 16 GB RAM, and Windows 11 operating system.

4.1 Datasets and Evaluation Metrics

4.1.1 Real-World Datasets

We select two real-world datasets, Citeseer[39] and BlogCatalog[40], to validate the performance of the algorithm. Citeseer is

a representative citation network, where the nodes represent papers and the edges are formed by the citation relationships between papers. A set of keywords is selected from the paper to generate a binary attribute vector, and the value of the node attribute representing the word is set to 0 if there is no word corresponding to the node in the paper, otherwise it is set to 1. BlogCatalog is a blogging network, whose nodes represent users' blogs, and the edges are formed by the interactions between users. The binary attribute of a node indicates the presence of a word in the blog corresponding to that node. Table 4 gives specific information about these real-world datasets.

Table 4 Description of the specific information of the real-world dataset, n, e, f , and k are the number of nodes, the number of edges, the number of attributes and the number of communities

Datasets	n	e	f	k
Citeseer	3312	4715	3703	6
BolgCatalog	5196	171743	8189	6

4.1.2 Synthetic Datasets

We use the LFR benchmarking tool [41] to generate the artificial network and refer to the methods in literature[29] and literature[42] to add 100 binary attributes to the network.The parameters of LFR are shown in Table 5. Among them, the mixing parameter mu controls the fuzziness of the boundary between communities, and the larger the value of mu , the more fuzzy the community structure in the network. The synthetic network specific information is shown in Table 6.

Table 5 Parameters and meanings of LFR synthetic network

Parameters	Meanings
n	Number of nodes
q	Average degree
$\max q$	Maximun degree
mu	Mixed parameter with values in the range [0,1]
$\min c$	Minimum community size
$\max c$	Maximum community size

Table 6 Description of the specific information of the synthetic network

	n	q	$\max q$	mu	$\min c$	$\max c$
Syn1	1000	20	50	0.1-0.7	10	100
Syn2	2000	20	50	0.1-0.7	10	100

4.1.3 Evaluation Metrics

We use three widely used community detection evaluation metrics[43], NMI (Normalised Mutual Information), ARI(Adjusted Rand Index) and ACC, to measure the accuracy of the algorithms.

NMI is commonly used in clustering and community detection tasks to measure the difference between the predicted results and the true results. The value of NMI is taken in the range [0,1] and higher value means better prediction.The NMI is calculated as follows:

$$NMI\left(X,Y\right)=\frac{2\times MI\left(X,Y\right)}{H\left(X\right)+H\left(Y\right)} \tag{18}$$

$$MI\left(X,Y\right)=\sum_{i=1}^{|X|}\sum_{j=1}^{|Y|}P\left(i,j\right)\log\left(\frac{P\left(i,j\right)}{P\left(i\right)P\left(j\right)}\right) \tag{19}$$

$$H\left(X\right)=-\sum_{i=1}^{|X|}P\left(i\right)\log\left(P\left(i\right)\right) \tag{20}$$

$$H\left(Y\right)=-\sum_{i=1}^{|Y|}P\left(j\right)\log\left(P\left(j\right)\right) \tag{21}$$

Where $MI\left(X,Y\right)$ is the mutual information between the real community detection result X and the predicted community detection result Y, indicating the correlation between the two sets, and $H(X)$, $H(Y)$ are the entropy of the real community detection result X and the predicted community detection result Y.

ACC is used to evaluate the accuracy of the algorithm's predicted community detection results compared to the actual community detection results, taking a value between [0,1], with larger values indicating higher prediction accuracy. The calculation is as follows:

$$ACC=\frac{\sum_{i=1}^{n}\delta\left(t_i,map(p_i)\right)}{n} \tag{22}$$

Where t_i is the true label of node v_i and p_i is the predicted label of node v_i. The map function maps the true label to the predicted label and $\delta\left(t_i,map(p_i)\right)$ is 1 if they are equal and 0 otherwise.

ARI is used to evaluate the agreement between the predicted result and the actual result and takes the value between [-1,1], the larger value indicating the higher prediction accuracy.RI and ARI are calculated as follows:

$$RI=\frac{TP+TN}{TP+FP+TN+FN} \tag{23}$$

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)} \tag{24}$$

Where TP is the number of actual positive, predicted positive labels; FP is the number of actual negative, predicted positive labels; TN is the number of actual positive, predicted negative labels; FN is the number of actual negative, predicted negative labels.

4.2 Parameter Experiments

We study the effect of important parameters α, β and d on the accuracy of the algorithm on real-world and synthetic datasets.

For the real-world datasets Citeseer and BlogCatalog, we first set $\beta = 0.7$, $d = 128$ and change $\alpha \in [0,1]$ gradually in steps of 0.1, as shown in Table 7, the algorithm reaches the maximum value of the three evaluation metrics on the two datasets when $\alpha = 0.7$. Next, with $\alpha = 0.7$ and $d = 128$, the algorithm gradually changes $\beta \in [0,1]$ in steps of 0.1, as shown in Table 8, the algorithm reaches the maximum value of the three metrics on the Citeseer dataset when $\beta = 0.6$, and the maximum value of the three metrics on the BlogCatalog dataset when $\beta = 0.7$. Finally, the algorithm gradually changes the value of the embedding dimension d by setting $\alpha = 0.7$ and $\beta = 0.6$ on the Citeseer dataset and $\alpha = 0.7$ and $\beta = 0.7$ on the BlogCatalog dataset, as shown in Table 9, and the algorithm reaches the maximum value of the three metrics at $d = 128$ on the Citeseer dataset and $d = 512$ on the BlogCatalog dataset. The three metrics reach their maximum values at $d = 512$ on the BlogCatalog dataset.

Table 7 Effect of parameter α on the accuracy of the algorithm on the real-world dataset

	Citeseer			BlogCatalog		
α	NMI	ARI	ACC	NMI	ARI	ACC
0	0.245	0.232	0.511	0.348	0.226	0.511
0.1	0.269	0.249	0.563	0.342	0.219	0.501
0.2	0.273	0.268	0.580	0.356	0.229	0.519
0.3	0.302	0.291	0.604	0.370	0.273	0.562
0.4	0.300	0.287	0.598	0.375	0.253	0.527
0.5	0.310	0.284	0.593	0.438	0.297	0.596
0.6	0.326	0.289	0.586	0.505	0.365	0.651
0.7	**0.327**	**0.296**	**0.599**	**0.523**	**0.413**	**0.675**
0.8	0.305	0.284	0.583	0.443	0.361	0.638
0.9	0.315	0.290	0.590	0.259	0.170	0.454
1.0	0.058	0.021	0.290	0.202	0.136	0.389

Table 8 Effect of parameter β on the accuracy of the algorithm on the real-world dataset

	Citeseer			BlogCatalog		
β	NMI	ARI	ACC	NMI	ARI	ACC
0	0.315	0.277	0.579	0.506	0.375	0.653
0.1	0.320	0.286	0.578	0.514	0.371	0.652
0.2	0.321	0.290	0.593	0.505	0.376	0.653
0.3	0.326	0.294	0.586	0.503	0.372	0.651
0.4	0.325	0.291	0.585	0.508	0.380	0.654
0.5	0.308	0.269	0.572	0.501	0.367	0.650
0.6	**0.327**	**0.296**	**0.599**	0.505	0.372	0.651
0.7	0.313	0.280	0.576	**0.523**	**0.413**	**0.675**
0.8	0.322	0.287	0.582	0.494	0.360	0.641
0.9	0.314	0.286	0.582	0.496	0.370	0.648
1.0	0.320	0.294	0.590	0.502	0.375	0.654

Table 9 Effect of parameter d on the accuracy of the algorithm on the real-world dataset

	Citeseer			BlogCatalog		
d	NMI	ARI	ACC	NMI	ARI	ACC
16	0.275	0.254	0.555	0.495	0.400	0.665
32	0.293	0.293	0.566	0.498	0.387	0.658
64	0.294	0.294	0.563	0.506	0.385	0.658
128	**0.327**	**0.296**	**0.599**	0.513	0.381	0.655
256	0.308	0.273	0.575	0.492	0.363	0.642
512	0.327	0.293	0.586	**0.523**	**0.413**	**0.675**

For the synthetic dataset Syn1, the mixing parameter mu is taken from 0.1 to 0.7. First, by setting $\beta = 0.7$ and $d = 128$, $\alpha \in [0,1]$ is gradually changed in steps of 0.1. As shown in Table 10, the algorithm reaches the maximum value of the three metrics on the two synthetic datasets when $\alpha \in [0.6, 0.7]$. Next, setting $\alpha = 0.7$, $d = 128$, and changing $\beta \in [0,1]$ gradually in steps of 0.1, as shown in Table 11, the evaluation metrics fluctuate in the range of 0.02 with the change of β and reach the maximum value at $\beta \in [0.6, 0.7]$.Finally, by setting $\alpha = 0.7$ and $\beta = 0.7$ and gradually changing the value of the embedding dimension d, as shown in Table 12, the accuracy of the algorithm continues to fluctuate in a small range.

Therefore, combining the results of the parameter experiments, in the remaining experiments we set $\alpha \in [0.6, 0.7]$, $\beta \in [0.6, 0.7]$ and $d = 128$.

Table 10 Effect of parameter α on the accuracy of the algorithm on Syn1 synthetic dataset

	α	mu=0.1	mu=0.2	mu=0.3	mu=0.4	mu=0.5	mu=0.6	mu=0.7
NMI	0	0.969	0.983	0.953	0.979	0.952	0.951	0.942
	0.1	0.980	0.984	0.967	0.974	0.955	0.952	0.951
	0.2	0.976	0.990	0.977	0.974	0.959	0.951	0.950
	0.3	0.992	0.989	0.986	0.980	0.974	0.959	0.950
	0.4	0.995	0.992	0.975	0.968	0.962	0.969	0.953
	0.5	1.0	0.991	0.991	0.976	0.970	0.948	0.972
	0.6	**1.0**	**1.0**	0.991	0.978	**0.987**	0.962	0.972
	0.7	1.0	0.993	0.993	**0.991**	0.967	0.958	0.956
	0.8	0.998	0.993	**1.0**	0.987	0.979	0.963	**0.976**
	0.9	0.991	0.990	0.996	0.974	0.986	**0.970**	0.966
	1.0	1.0	0.998	0.989	0.956	0.985	0.434	0.457
ARI	0	0.962	0.978	0.949	0.972	0.943	0.940	0.934
	0.1	0.953	0.982	0.946	0.955	0.950	0.936	0.932
	0.2	0.952	0.989	0.977	0.973	0.941	0.944	0.948
	0.3	0.977	0.988	0.987	0.978	0.945	0.955	0.945
	0.4	0.993	0.990	0.963	0.958	0.950	**0.969**	0.952
	0.5	1.0	0.982	0.990	0.964	0.954	0.938	**0.977**
	0.6	**1.0**	**1.0**	0.990	0.961	**0.981**	0.960	0.972
	0.7	1.0	0.990	0.990	0.977	0.935	0.950	0.949
	0.8	0.992	0.993	**1.0**	**0.983**	0.965	0.949	0.971
	0.9	0.990	0.988	0.996	0.964	0.977	0.949	0.940
	1.0	1.0	0.998	0.986	0.938	0.708	0.234	0.261
ACC	0	0.977	0.981	0.952	0.969	0.94	0.933	0.928
	0.1	0.983	0.982	0.959	0.978	0.951	0.931	0.934
	0.2	0.983	0.982	0.973	0.974	0.963	0.939	0.939
	0.3	0.985	0.983	0.98	0.98	0.98	0.944	0.938
	0.4	0.995	0.984	0.968	0.967	0.966	**0.956**	0.943
	0.5	1.0	0.985	0.983	0.972	0.966	0.944	0.965
	0.6	**1.0**	**1.0**	0.983	0.968	**0.989**	0.953	**0.965**
	0.7	1.0	0.985	0.985	**0.985**	0.957	0.95	0.942
	0.8	0.985	0.985	**1.0**	0.969	0.981	0.955	0.965
	0.9	0.985	0.984	0.985	0.967	0.985	0.944	0.944
	1.0	1.0	0.999	0.982	0.976	0.865	0.531	0.552

Table 11 Effect of parameter β on the accuracy of the algorithm on Syn1 synthetic dataset

	β	mu=0.1	mu=0.2	mu=0.3	mu=0.4	mu=0.5	mu=0.6	mu=0.7
NMI	0	1.0	0.994	0.984	0.990	0.976	0.961	0.958
	0.1	1.0	0.989	0.982	0.975	0.979	0.951	0.953
	0.2	0.989	0.995	0.982	0.971	0.971	0.958	0.956
	0.3	0.994	0.991	0.984	0.980	0.967	0.970	0.968
	0.4	1.0	1.0	0.995	0.977	0.979	**0.970**	0.957
	0.5	1.0	0.994	0.986	0.982	**0.980**	0.951	**0.970**
	0.6	**1.0**	**1.0**	0.987	**0.992**	0.976	0.957	0.966
	0.7	1.0	0.992	0.990	0.984	0.967	0.951	0.965
	0.8	0.992	0.989	0.985	0.985	0.974	0.956	0.957
	0.9	0.991	0.993	**0.995**	0.980	0.976	0.958	0.960
	1.0	1.0	1.0	0.986	0.978	0.967	0.961	0.968
ARI	0	1.0	0.991	0.979	0.987	0.972	0.942	0.953
	0.1	1.0	0.983	0.981	0.953	0.974	0.939	0.938
	0.2	0.983	0.994	0.973	0.954	0.961	0.945	0.948
	0.3	0.992	0.986	0.978	0.972	0.956	0.966	0.963
	0.4	1.0	1.0	**0.994**	0.958	0.975	**0.966**	0.948
	0.5	1.0	0.993	0.982	0.974	**0.977**	0.947	0.965
	0.6	**1.0**	**1.0**	0.984	**0.990**	0.963	0.956	0.966
	0.7	1.0	0.991	0.988	0.981	0.952	0.943	0.956
	0.8	0.991	0.989	0.979	0.971	0.96	0.946	0.947
	0.9	0.985	0.993	0.994	0.959	0.963	0.950	0.957
	1.0	1.0	1.0	0.982	0.970	0.954	0.964	**0.969**
ACC	0	1.0	0.985	0.971	**0.985**	0.966	0.945	0.948
	0.1	1.0	0.979	0.983	0.964	0.97	0.935	0.938
	0.2	0.985	0.985	0.97	0.965	0.96	0.942	0.94
	0.3	0.985	0.985	0.97	0.966	0.97	0.952	0.952
	0.4	1.0	1.0	0.985	0.969	0.969	**0.958**	0.949
	0.5	1.0	0.985	0.977	0.97	**0.977**	0.937	0.946
	0.6	**1.0**	**1.0**	**0.985**	0.978	0.976	0.949	0.959
	0.7	1.0	0.985	0.972	0.97	0.96	0.943	0.957
	0.8	0.985	0.985	0.976	0.969	0.968	0.946	0.945
	0.9	0.985	0.985	0.985	0.969	0.968	0.941	0.945
	1.0	1.0	1.0	0.969	0.961	0.957	0.955	**0.964**

Table 12 Effect of parameter d on the accuracy of the algorithm on Syn1 synthetic dataset

	d	mu=0.1	mu=0.2	mu=0.3	mu=0.4	mu=0.5	mu=0.6	mu=0.7
NMI	16	**1.0**	**1.0**	**1.0**	**0.999**	**0.990**	**0.984**	**0.978**
	32	1.0	1.0	0.993	0.998	0.986	0.977	0.974
	64	1.0	1.0	0.993	0.993	0.974	0.961	0.960
	128	1.0	1.0	0.978	0.989	0.974	0.959	0.954
	256	1.0	0.993	0.985	0.979	0.969	0.958	0.958
	512	1.0	0.991	0.978	0.973	0.973	0.950	0.969
ARI	16	**1.0**	**1.0**	**1.0**	**0.998**	**0.990**	**0.987**	**0.982**
	32	1.0	1.0	0.992	0.998	0.984	0.981	0.966
	64	1.0	1.0	0.992	0.993	0.965	0.957	0.958
	128	1.0	1.0	0.969	0.984	0.949	0.955	0.945
	256	1.0	0.983	0.985	0.962	0.962	0.957	0.945
	512	1.0	0.990	0.964	0.958	0.960	0.941	0.949
ACC	16	**1.0**	**1.0**	**1.0**	**0.999**	**0.994**	**0.987**	**0.982**
	32	1.0	1.0	0.984	0.999	0.991	0.975	0.974
	64	1.0	1.0	0.984	0.985	0.976	0.962	0.962
	128	1.0	1.0	0.97	0.985	0.968	0.948	0.96
	256	1.0	0.985	0.969	0.963	0.953	0.944	0.946
	512	1.0	0.979	0.97	0.969	0.971	0.931	0.943

4.3 Accuracy Experiments

4.3.1 Experimental Results on Synthetic Datasets

Tables 13 and 14 show the experimental results for the three evaluation metrics NMI, ARI and ACC on the synthetic datasets Syn1 and Syn2 for RWACD and the algorithms DeepWalk and Node2Vec, which are also based on random walk. The results show that the RWACD algorithm consistently outperforms the other two algorithms. This is because the RWACD algorithm considers the community structure while walking, adopts different walking strategies for internal and boundary nodes, and uses the attribute information of the nodes, so the community delineation results are more accurate.Both the DeepWalk and Node2Vec algorithms also have high accuracy on the synthetic dataset, which is due to the fact that topology plays an important role in the synthetic dataset, while they both achieve effective use of the topological information of the nodes. However, as the value of mu increases, the community boundaries gradually become blurred, and when mu reaches 0.7, the accuracy of DeepWalk and Node2Vec decreases significantly, while the accuracy of RWACD remains stable, which proves the effectiveness and stability of the random walk strategy that takes into account the community structure for community detection.

Table 13 Results of accuracy experiments on synthetic network Syn1

	DeepWalk			Node2Vec			RWACD		
mu	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
0.1	**1.0**	**1.0**	**1.0**	0.999	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
0.2	**1.0**	**1.0**	**1.0**	0.983	0.954	0.985	**1.0**	**1.0**	**1.0**
0.3	**1.0**	**1.0**	**1.0**	0.999	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
0.4	**1.0**	**1.0**	**1.0**	0.996	0.996	0.998	**1.0**	**1.0**	**1.0**
0.5	0.991	0.990	0.995	0.973	0.947	0.969	**0.993**	**0.991**	**0.997**
0.6	0.870	0.828	0.915	0.845	0.789	0.886	**0.962**	**0.959**	**0.953**
0.7	0.269	0.073	0.294	0.226	0.072	0.281	**0.954**	**0.945**	**0.940**

Table 14 Results of accuracy experiments on synthetic network Syn2

	DeepWalk			Node2Vec			RWACD		
mu	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
0.1	0.989	0.984	0.991	0.983	0.959	0.988	**0.995**	**0.990**	**0.993**
0.2	0.987	0.977	0.987	0.981	0.956	0.981	**0.990**	**0.995**	**0.991**
0.3	0.983	0.973	0.982	0.980	0.956	0.986	**0.987**	**0.982**	**0.990**
0.4	0.979	0.970	0.980	0.979	0.942	0.981	**0.985**	**0.979**	**0.986**
0.5	0.931	0.922	0.955	0.970	0.931	0.968	**0.979**	**0.965**	**0.975**
0.6	0.915	0.908	0.965	0.966	0.922	0.962	**0.971**	**0.958**	**0.970**
0.7	0.539	0.234	0.474	0.629	0.361	0.605	**0.836**	**0.573**	**0.786**

4.3.2 Experimental Results on Real-World Datasets

Table 15 shows the experimental results of RWACD with other baseline algorithms on two real-world datasets, Citeseer and BlogCatalog, for three evaluation metrics, NMI, ARI and ACC. Among them, Louvain, DeepWalk, LINE and Node2Vec use only the topology information of the nodes, while AANE and AWBA use both topology and attribute information of the nodes. It can be seen that most of the methods that can use both topology and attribute information are better than those that only use topology information. The RWACD algorithm outperforms the comparison algorithms in all metrics except the NMI metrics, where it is slightly lower than the AWBA algorithm on the Citeseer dataset. On the Citeseer dataset, although RWACD achieves the best results, the DeepWalk and Node2Vec algorithms, which only use topological information, also achieve good results, probably because topological information in the Citeseer dataset has a stronger effect on community structure than attribute information. For the BlogCatalog dataset, the RWACD algorithm has a large improvement over the DeepWalk and Node2Vec algorithms, indicating that attribute information has a stronger influence on the community structure of the BlogCatalog dataset. RWACD also has a significant

advantage over the AANE and AWBA algorithms. The experimental results demonstrate the effectiveness of our consideration of node attribute information and the proposed walking strategy in identifying community structure.

4.4 Comparison Experiment of Walking Parameter Setting Methods

Table 15 Results of accuracy experiments on real-world network

	Citeseer			BlogCatalog		
	NMI	ARI	ACC	NMI	ARI	ACC
Louvain	0.124	0.112	0.353	0.160	0.101	0.330
DeepWalk	0.235	0.124	0.453	0.228	0.142	0.417
LINE	0.136	0.112	0.314	0.212	0.144	0.393
Node2Vec	0.245	0.129	0.473	0.225	0.137	0.414
AANE	0.209	0.158	0.433	0.320	0.225	0.481
AWBA	**0.338**	0.276	0.535	0.277	0.176	0.436
RWACD	0.327	**0.296**	**0.599**	**0.523**	**0.413**	**0.675**

We conduct experiments on two real-world datasets, Citeseer and BlogCatalog, to verify the effectiveness and flexibility of the proposed strategy of automatically determining the number of wanders and the walking length compared to the manual setting method. First, we set the number of wanders and wander length to $[10,100]$, the step size to 10, and keep the rest of the parameters consistent. Then, the optimal results on the two datasets are taken and compared with the results of RWACD, and the experimental results are shown in Figure 3 below. The results show that our algorithm not only has no loss of accuracy, but also eliminates the need for tedious parameter experiments to determine the optimal parameters for different datasets, avoiding overfitting while greatly simplifying the parameter setting process. It also reduces the negative impact on the experimental results caused by the oversampling of boundary nodes, which results in a large amount of redundant information when the walking parameters are set too large.

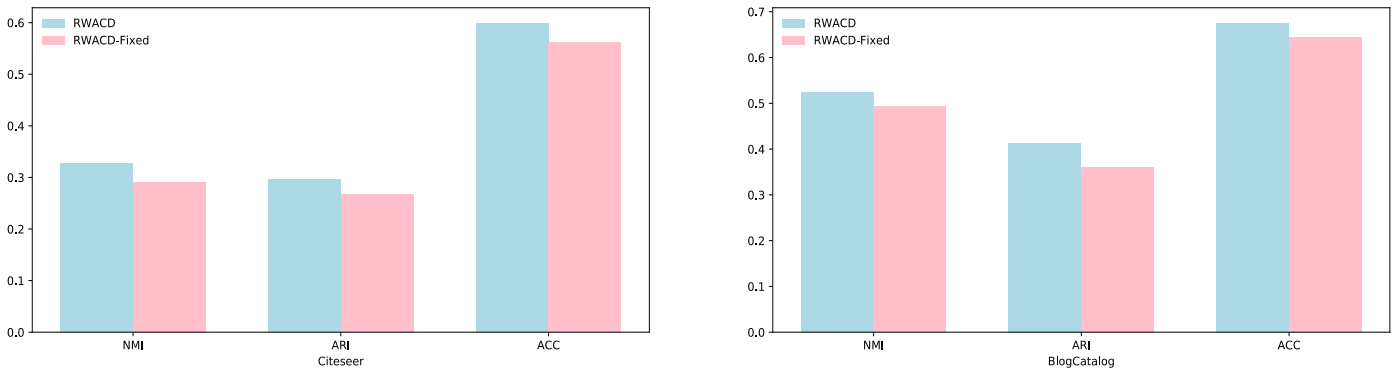


Figure 3 Experimental results of automatic determination of walking parameters and manual setting of walking parameters

4.5 Visualisation

In the visualization experiment, this study uses the t-SNE algorithm to reduce the dimensionality of node embedding vectors

and community detection results to a two-dimensional space for visualization. Nodes of the same color belong to the same community. Figures 4 to 7 show the visualization results of the synthetic and real datasets for DeepWalk, Node2Vec, and RWACD algorithms. The parameter mu of the synthetic dataset is 0.7. It can be seen that the embedding vectors obtained by the RWACD algorithm are more concentrated compared to other results, and there are almost no isolated node groups, which demonstrates the effectiveness of our algorithm's embedding vectors in community detection tasks.

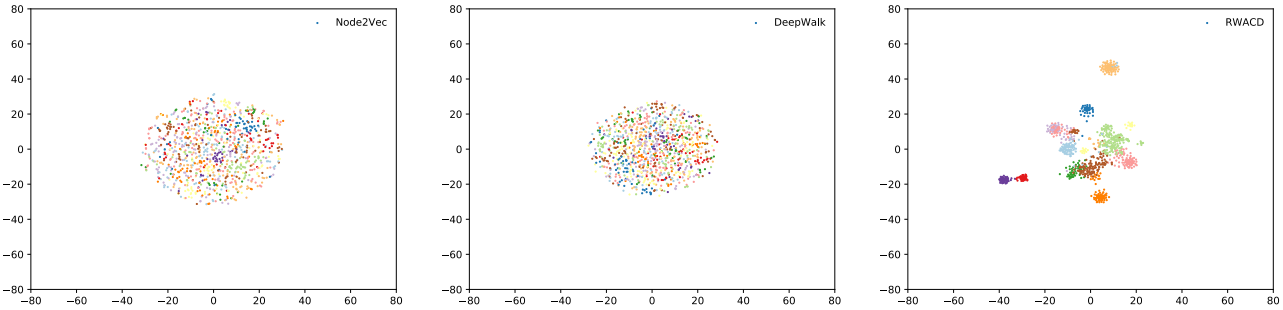


Figure 4 Visualization results of the Syn1 dataset

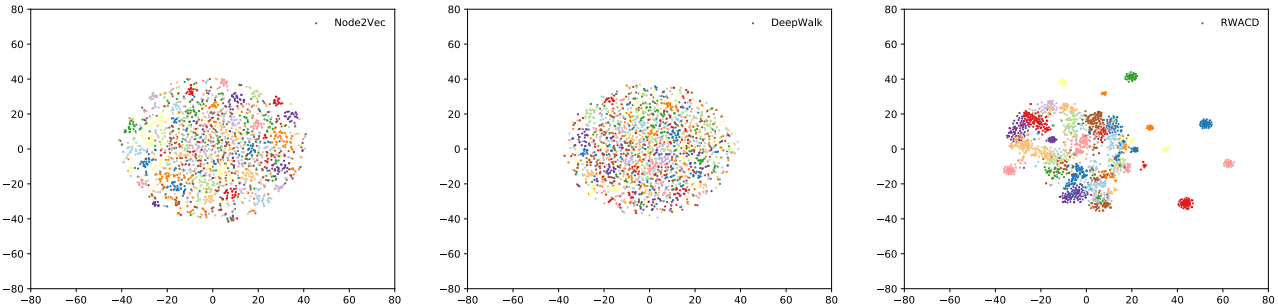


Figure 5 Visualization results of the Syn2 dataset

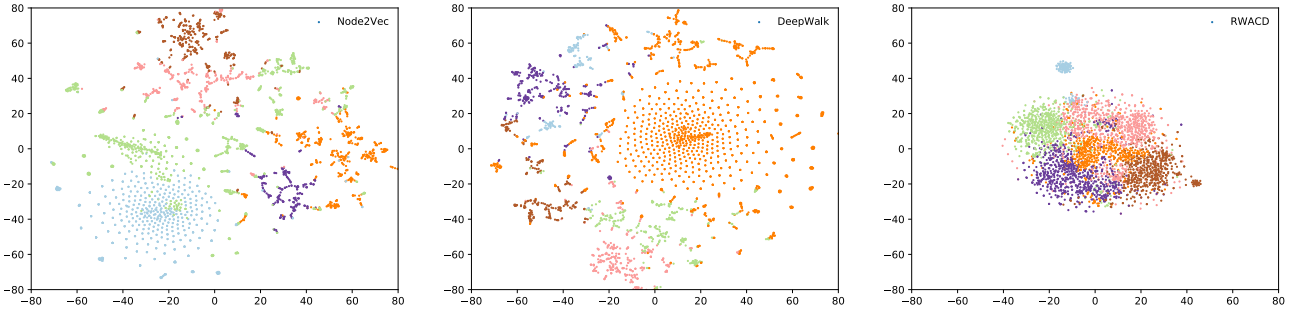


Figure 6 Visualization results of the Citeseer dataset

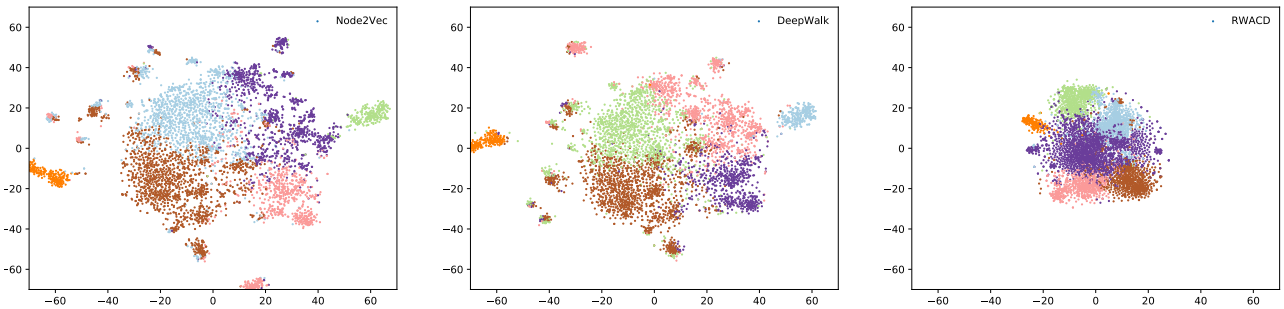


Figure 7 Visualization results of the BlogCatalog dataset

5. Conclusion

In this paper, we propose a network embedding based on community-aware and random walk for community detection in attributed networks. Firstly, we construct the community similarity matrix using the topological similarity and attribute similarity of nodes, which ensures the effective use of node topological and attribute information. Second, we construct transition probability matrices that favour walking within the community and at the community boundary, respectively, to enhance node walking within the community and at the community boundary to better maintain and identify the community structure. Finally, we set the walking times and lengths for nodes according to their degree and attribute numbers to simplify the process of setting the walking parameters and to avoid the problems of overfitting as well as oversampling of nodes caused by manually setting the walking parameters, under the premise of guaranteeing the accuracy of the algorithm. Experimental results on real-world and synthetic datasets show that our algorithm outperforms other baseline methods. In the future, based on the existing work, we consider the development of a community detection algorithm that automatically determines the number of communities.

Reference

[1] Xu X L, Xiao Y Y, Yang X H, et al. Attributed network community detection based on network embedding and parameter-free clustering[J]. Applied Intelligence, 2022: 1-14.

[2] Berahmand K, Haghani S, Rostami M, et al. A new attributed graph clustering by using label propagation in complex networks[J]. Journal of King Saud University-Computer and Information Sciences, 2022, 34(5): 1869-1883.

[3] Berahmand K, Mohammadi M, Faroughi A, et al. A novel method of spectral clustering in attributed networks by constructing parameter-free affinity matrix[J]. Cluster Computing, 2022: 1-20.

[4] Meng F, Rui X, Wang Z, et al. Coupled node similarity learning for community detection in attributed networks[J]. Entropy, 2018, 20(6): 471.

[5] Falih I, Grozavu N, Kanawati R, et al. Community detection in attributed network[C]//Companion proceedings of the the web conference 2018. 2018: 1299-1306.

[6] Chang Y, Ma H, Chang L, et al. Community detection with attributed random walk via seed replacement[J]. Frontiers of Computer Science, 2022, 16(5): 165324.

[7] Ding Y, Wei H, Hu G, et al. Unifying community detection and network embedding in attributed networks[J]. Knowledge and Information Systems, 2021, 63(5): 1221-1239.

[8] Cui P, Wang X, Pei J, et al. A survey on network embedding[J]. IEEE transactions on knowledge and data engineering, 2018, 31(5): 833-852.

[9] Huang X, Li J, Zou N, et al. A general embedding framework for heterogeneous information learning in large-scale networks[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2018, 12(6): 1-24.

[10] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014: 701-710.

[11] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[12] Newman M E J, Girvan M. Finding and evaluating community structure in networks[J]. Physical review E, 2004, 69(2): 026113.

[13] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical review E, 2007, 76(3): 036106.

[14] Newman M E J. Fast algorithm for detecting community structure in networks[J]. Physical review E, 2004, 69(6): 066133.

[15] Palla G, Derényi I, Farkas I, et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. nature, 2005, 435(7043): 814-818.

[16] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of statistical mechanics: theory and experiment, 2008, 2008(10): P10008.

[17] Ding X, Yang H, Zhang J, et al. CEO: Identifying Overlapping Communities via Construction, Expansion and Optimization[J]. Information Sciences, 2022, 596: 93-118.

[18] Zhao X, Liang J, Wang J. A community detection algorithm based on graph compression for large-scale social networks[J].

Information Sciences, 2021, 551: 358-372.

[19] Berahmand K, Mohammadi M, Faroughi A, et al. A novel method of spectral clustering in attributed networks by constructing parameter-free affinity matrix[J]. Cluster Computing, 2022, 25(2): 869-888.

[20] Zhe C, Sun A, Xiao X. Community detection on large complex attribute network[C]//Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019: 2041-2049.

[21] Meng F, Rui X, Wang Z, et al. Coupled node similarity learning for community detection in attributed networks[J]. Entropy, 2018, 20(6): 471.

[22] Huang X, Li J, Hu X. Accelerated attributed network embedding[C]//Proceedings of the 2017 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2017: 633-641.

[23] Ye Z, Zhang H, Feng L, et al. CDCN: a new nmf-based community detection method with community structures and node attributes[J]. Wireless Communications and Mobile Computing, 2021, 2021: 1-12.

[24] Gao Y, Gong M, Xie Y, et al. Community-oriented attributed network embedding[J]. Knowledge-Based Systems, 2020, 193: 105418.

[25] Lu D D, Qi J, Yan J, et al. Community detection combining topology and attribute information[J]. Knowledge and Information Systems, 2022: 1-22.

[26] Wang X, Cui P, Wang J, et al. Community preserving network embedding[C]//Proceedings of the AAAI conference on artificial intelligence. 2017, 31(1).

[27] Huang T, Zhou L, Wang L, et al. Attributed network embedding with community preservation[C]//2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA). IEEE, 2020: 334-343.

[28] Jin D, Liu Z, Li W, et al. Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks[C]//Proceedings of the AAAI conference on artificial intelligence. 2019, 33(01): 152-159.

[29] Guo K, Zhang P, Guo W, et al. An attentional-walk-based autoencoder for community detection[J]. Applied Intelligence, 2022: 1-19.

[30] Cui G, Zhou J, Yang C, et al. Adaptive graph encoder for attributed graph embedding[C]//Proceedings of the 26th ACM

SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 976-985.

[31] Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding[C]//Proceedings of the 24th international conference on world wide web. 2015: 1067-1077.

[32] Grover A, Leskovec J. node2vec: Scalable feature learning for networks[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864.

[33] Zhang Y, Shi Z, Feng D, et al. Degree-biased random walk for large-scale network embedding[J]. Future Generation Computer Systems, 2019, 100: 198-209.

[34] Mohammed S N, Gündüç S. Degree-based random walk approach for graph embedding[J]. Turkish Journal of Electrical Engineering and Computer Sciences, 2022, 30(5): 1868-1881.

[35] Berahmand K, Nasiri E, Rostami M, et al. A modified DeepWalk method for link prediction in attributed social network[J]. Computing, 2021, 103: 2227-2249.

[36] Wei H, Pan Z, Hu G, et al. Attributed network representation learning via DeepWalk[J]. Intelligent Data Analysis, 2019, 23(4): 877-893.

[37] Yang C, Liu Z, Zhao D, et al. Network representation learning with rich text information[C]//IJCAI. 2015, 2015: 2111-2117.

[38] Dou W, Zhang W, Weng Z. An Attributed Network Representation Learning Method Based on Biased Random Walk[J]. Procedia Computer Science, 2020, 174: 291-298.

[39] Sen P, Namata G, Bilgic M, et al. Collective classification in network data[J]. AI magazine, 2008, 29(3): 93-93.

[40] Huang X, Li J, Hu X. Label informed attributed network embedding[C]//Proceedings of the tenth ACM international conference on web search and data mining. 2017: 731-739.

[41] Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms[J]. Physical review E, 2008, 78(4): 046110.

[42] Sun H, He F, Huang J, et al. Network embedding for community detection in attributed networks[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2020, 14(3): 1-25.

[43] Chakraborty T, Dalmia A, Mukherjee A, et al. Metrics for community analysis: A survey[J]. ACM Computing Surveys (CSUR),

2017, 50(4): 1-37.

[44] Van der Maaten L, Hinton G. Visualizing data using t-SNE[J]. Journal of machine learning research, 2008, 9(11).



Shuo Gao received the B.E. and M.E. degrees from Harbin University of Science and Technology, in 2009, and 2013, respectively. Currently, he is an Associate Professor at the College of Intelligent Engineering, Harbin Institute of Petroleum, Harbin. His main research interests include data mining and network security.



Qi Zhen received the B.E. and M.E. degrees from Northeast Forestry University, in 2016, and 2018, respectively. Currently, he is a lecturer at the College of Intelligent Engineering, Harbin Institute of Petroleum, Harbin. His main research interests include data mining, deep learning, and network security.