Knowledge Graph-Based Badminton Tactics Mining and Reasoning for Badminton Player Training Pattern Analysis and Optimization

Xingli Hu[1]*, Jiangtao Li[2], Ren Cai[3]
School of Physical Education, University of Sanya, Sanya 572022, Hainan, China[1, 2]
Hainan Medical College, Haikou 571157, Hainan, China[3]

***Abstract*—As the global emphasis on sports data analysis and athlete performance optimization continues to grow, traditional badminton training methods are increasingly insufficient to meet the demands of modern high-level competitive sports. The exploration and reasoning of badminton tactics can significantly aid coaches and athletes in better comprehending game strategies, playing a vital role in the analysis and optimization of training methods. By utilizing knowledge graph-based badminton tactics mining, an approach involving heterogeneous graph splitting is employed, coupled with the incorporation of a cross-relational attention mechanism within relational graph neural networks. This mechanism assigns varying weights based on the importance of neighboring nodes across different relations, facilitating information aggregation and dissemination across multiple relationships. Furthermore, to address the challenges posed by the complexity of large-scale knowledge graphs, which feature numerous entity relationships and intricate internal structures, techniques such as training subgraph sampling, positive-negative sampling, and block-diagonal matrix decomposition are introduced. These techniques help to reduce the computational load and complexity of model training, while also enhancing the model's generalization capabilities. Finally, comparative experiments conducted on a proprietary badminton tactics dataset demonstrated the effectiveness and superiority of the proposed model improvements when reasonable parameters were applied. The case study shows that this approach holds considerable promise for the analysis and optimization of badminton players' training strategies.**

Keywords*—Badminton tactical analysis; graph neural networks; attention mechanisms; training pattern optimization; heterogeneous graph splitting; artificial intelligence

I. Introduction

As competitive sports continue to evolve, the demands on athletes to demonstrate exceptional performance in matches have become increasingly stringent. To meet these challenges, athletes need not only strong physical and psychological qualities but also advanced technical and tactical skills to identify and address their weaknesses while maximizing their strengths. Effective and scientifically sound training is crucial for enhancing athletes' competitive levels. However, the current training systems still largely rely on coaches' experience, with insufficient support from systematic data. This is particularly evident in badminton training, where the focus on technical and tactical training is relatively low, often depending on the subjective judgments of coaches and the personal feelings of athletes. This reliance on traditional methods has led to the underutilization of critical match data and a lack of scientifically based standards for evaluating technical and tactical performance.

In this context, the application of information technology is essential for improving traditional training models. However, current information technologies face significant challenges in data collection, analysis, and training optimization: data collection is often untimely, incomplete, and inaccurate, making integration difficult; data analysis lacks depth, failing to produce actionable insights; and personalized technical and tactical training programs are scarce and lack scientific backing. Additionally, the speed of information feedback does not match the fast pace of training and competition, ultimately affecting athletes' performance.

Therefore, the effective collection and utilization of training and competition data, as well as the development of personalized training programs based on this data, have become urgent tasks. With the advancement of modern database technologies and data mining techniques, these tools have increasingly been applied in the sports domain. Establishing a badminton sports database and data analysis system can significantly enhance the efficiency and depth of data collection, providing valuable insights through in-depth data analysis to inform coaches' training decisions and optimize training methods.

To fill these gaps, this paper explores how badminton tactics mining and reasoning based on knowledge graphs can assist coaches and athletes in better understanding match strategies and play a crucial role in the analysis and optimization of training methods. We propose a tactics mining approach based on heterogeneous graph decomposition, which incorporates a cross-relational attention mechanism to assign different weights in relational graph neural networks, enabling the aggregation and transmission of information across relationships. Additionally, to address the complexity challenges posed by large-scale knowledge graphs with intricate relational structures, we introduce techniques such as training subgraph sampling, positive-negative sampling, and block-diagonal matrix decomposition to reduce computational complexity and enhance model generalization. Comparative experiments conducted on a proprietary badminton tactics dataset demonstrate the effectiveness and superiority of the improvements made to our model.

*Corresponding Author

II. Literature Review

This section reviews existing related work on data mining to highlight the gaps in existing research.

A. Data Mining Techniques

Data mining technology has found widespread use in the global sports arena [1]. A well-known example is the Advanced Scout system employed by teams in the American professional basketball league, which processes game data to generate key insights, such as identifying high-percentage shooting areas and determining the most effective player rotations. This system aids head coaches in developing evidence-based training plans and making real-time strategic decisions during games [2]. In the National Football League (NFL), Schatz's study of league statistics revealed a link between league averages and the performance success of players in specific offensive roles [3]. Similarly, in the National Collegiate Athletic Association (NCAA), Jay Coleman and his team analyzed historical sports data, formulating a predictive model for game outcomes that boasts an accuracy rate of 84.5% [4].

In the realm of tennis, Damien Demaj tracked the serve trajectories of opponents and, utilizing spatial and temporal data analysis techniques, discerned patterns in player movements. This allowed athletes to gain a better understanding of their opponents' technical and tactical tendencies, ultimately improving their match performance [5][6]. Additionally, Rajiv Maheswara and his team leveraged data mining and machine learning to scrutinize the movements of players, referees, and basketballs captured by high-speed cameras during NBA games. Their analysis focused on offensive scoring efficiency and defensive effectiveness, providing coaches with critical information to structure more strategic training programs for their teams [7].

B. Traditional Knowledge Reasoning

Rule-based reasoning relies on a predefined set of rules to logically derive new knowledge, directly incorporating expert domain knowledge. While this approach is effective, it is often time-consuming to create these rules, and the system's scalability is generally limited. For instance, knowledge graphs like NELL [8] and YAGO [9] use rule-based methods to expand their knowledge bases. Meanwhile, first-order probabilistic language models focus on achieving precise "local" reasoning, proving effective in tasks such as entity resolution by extending stochastic logic programs and employing PageRank variants [10] for inference. Additionally, the TensorLog system integrates knowledge graph reasoning with gradient-based deep learning [11], achieving linear computational efficiency through a tractable reasoning process.

Ontology-based reasoning, on the other hand, generates new knowledge by interpreting, reasoning, and integrating ontologies—core structures that describe entities, attributes, relationships, and concepts. For example, a semi-automatic schema construction approach [12] addresses the complex schema challenges in RDF knowledge bases, while a system based on Markov logic networks is employed to clean and refine the original knowledge base [13]. Furthermore, the method proposed by Pujara et al. [14], which uses joint inference with probabilistic soft logic [15][16], tackles issues of noise and incomplete information in large-scale knowledge graphs. The ontology path discovery algorithm OP developed by Chen et al. further enhances knowledge graphs through optimization techniques [17].

In summary, although traditional reasoning methods can be effectively applied to knowledge graphs, ontology-based methods typically offer higher accuracy, making them suitable for scenarios where precision is critical. However, these methods face challenges with large-scale knowledge instantiation, struggling with inference efficiency and recall, while noise in the raw data can lead to inference errors, limiting the scope and applicability of these traditional approaches.

C. Distributed Knowledge Reasoning Approach

Distributed representation learning centers on transforming traditional symbolic representations into numerical representations in vector space through mapping functions as a way to mitigate dimensionality catastrophe and capture implicit connections between entities and relationships. These methods are generally divided into two main categories: rule-based reasoning and ontology-based reasoning.

The TransE model [18] is a distance-based approach that embeds entities and relations in a low-dimensional vector space to model multi-relational data. It predicts links by making the sum of the head entity vector and the relation vector as close as possible to the tail entity vector. TransE has demonstrated strong performance across several knowledge bases, surpassing advanced methods of its time, and thus gained attention. While simple and fast to train, its main limitation is that it only handles one-to-one relationships. To address this, Zhen et al. [19] introduced an improved version that considers complex relationships, such as one-to-many, many-to-one, and many-to-many. The TransH model followed, performing translation on a hyperplane to better handle diverse relationships, while maintaining similar complexity to TransE. Additionally, specific strategies were introduced to mitigate incorrect negative labels, improving performance on tasks like link prediction and fact extraction in knowledge graphs like WordNet and Freebase.

The RESCAL model, proposed by Nickel et al. [20], is a tensor decomposition-based inference method that enables collective learning through its latent components and offers efficient factorization. RESCAL demonstrated significant improvements in speed and accuracy over other tensor methods. Based on RESCAL, Chang et al. [21] proposed TRESCAL, which excels in relationship extraction by leveraging entity type information, allowing faster and more accurate discovery of new relations in databases.

Inference methods based on distributed representations are more advanced than traditional methods. The TransE series, known for its simplicity and effectiveness, has become a research focus, achieving significant results despite limited further research space. The RESCAL series, favored for its interpretability and performance, holds great potential, despite higher model complexity.

D. Research Gaps

Although progress has been made in applying graph neural networks and association rule mining to knowledge graphs [22], significant gaps remain in their application to badminton tactical

reasoning and training. Current algorithms fail to fully consider dynamic player positioning, opponent adaptability, and the data requirements of input models.

1) Model adaptability and flexibility: Existing graph neural networks can simulate some aspects of badminton tactics but struggle with the complexity and real-time nature of tactical changes. These models require deeper integration of tactical theory and practical experience, and current approaches are often inefficient due to heavy reliance on data annotation and feature design.

2) Dataset standardization: The lack of unified standards for data collection and preprocessing in badminton tactics leads to inconsistencies and hinders the generalizability of different studies. Without standardized data representation, tactical reasoning methods face challenges in coherence and adaptability.

3) Complex tactical reasoning: Current techniques struggle with recognizing and reasoning about complex tactical combinations and rapidly changing scenarios. They often underperform in long-term tactical evolution and opponent strategy adaptation, requiring further optimization.

In summary, future research should focus on developing more adaptable models and standardized datasets to enhance tactical reasoning, thereby improving technical and tactical teaching and game strategy analysis in badminton.

III. Ideas for Improving Reasoning in Very Large Scale Knowledge Graphs

A. Heterogeneous Graph Splitting

Entities and relationships in a knowledge graph often exhibit diverse types and properties, creating a heterogeneous graph structure. This heterogeneity is crucial in reasoning since different types of entities and relationships may involve distinct rules, constraints, and semantics.

In this study, we employ a knowledge graph splitting approach. Heterogeneous graph splitting involves dividing a large, complex graph into smaller, more manageable subgraphs based on specific rules. This method enhances the scalability, maintainability, and processing efficiency of large knowledge graphs, particularly in distributed environments where managing smaller subgraphs is more practical.

The process of splitting a heterogeneous knowledge graph involves several key steps:

- Defining Splitting Rules: Based on the characteristics and application needs of the knowledge graph, we establish rules for splitting—such as dividing by entity types, relationship types, or attributes.
- Constructing Topologies: Using the defined rules, we construct the topology for each subgraph. This can be achieved through graph-based clustering algorithms or similarity calculations based on meta-paths.
- Performing the Split: Entities and relationships in the original graph are divided into multiple subgraphs according to the constructed topologies.
- Storing and Managing Subgraphs: Each subgraph is stored in different computing nodes or distributed storage systems, with appropriate management and maintenance.

In our study, we focus on splitting subgraphs based on relationship types. During the reasoning process, these subgraphs are embedded to better capture the distinct semantic information inherent in the heterogeneous graph structure. For example, as illustrated in Fig. 1, an original directed graph with five nodes and two types of relationships can be split into two subgraphs, each retaining the directed nature of the original structure. This allows for more precise reasoning based on the specific types of relationships in each subgraph.

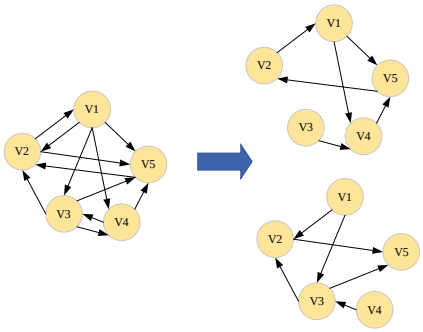

Fig. 1. Schematic diagram of knowledge graph heterogeneous graph splitting.

B. Training Subgraph Sampling

Training graph neural networks on large-scale datasets presents challenges, such as high computational complexity and issues like gradient vanishing or explosion. To address these, this study employs random sampling to generate subgraphs for training. Random sampling reduces noise by excluding irrelevant nodes and edges, thereby lowering computational and storage demands and improving training efficiency. However, tuning sampling parameters is necessary to balance subgraph size with the importance of sampled components.

Additionally, positive and negative sampling is used to preprocess training data by selecting real (positive) and non-existent (negative) edges or nodes. This approach offers several benefits:

- Addressing Data Imbalance: By balancing the number of positive and negative examples, the model avoids bias towards predicting positive instances, enhancing its overall prediction capabilities.
- Reducing Computation: Sampling limits the data used for training, cutting down on computational load and improving efficiency.
- Improving Generalization: Positive and negative sampling enables the model to learn a broader range of features, enhancing its adaptability to different graph datasets.

In summary, the introduction of positive and negative sampling in this study is crucial for addressing data imbalance, boosting computational efficiency, and enhancing the model's generalization ability.

IV. Improved Knowledge Graph Link Prediction Algorithm

In a knowledge graph, link prediction involves forecasting the relationship between two entities based on the existing graph structure, often referred to as relationship extraction. This process is crucial for expanding the knowledge graph, as manually labeling every entity and relationship is impractical given their vast number. Link prediction enables the automatic discovery of new entities and relationships from large volumes of unlabeled data, thereby enhancing the graph's scale and richness. The primary objective is to predict potential relationships between entities or to identify links between entities and relationships. This task has broad applications in fields such as information retrieval, natural language processing, recommender systems, and Q&A, where it helps to uncover correlations within the knowledge graph, thereby improving the effectiveness of these systems.

In this study, building on the concepts outlined in Fig. 1, we aim to enhance the knowledge graph link prediction algorithm by integrating key techniques such as knowledge graph embedding, data preprocessing, and graph neural networks. This approach will enable convolutional learning of entities and relationships, utilize the DistMult decoder for scoring, and apply association rule mining to select the ternary elements for evaluation. The overall architecture of the improved algorithm is depicted in Fig. 2.

A. Knowledge Graph Input to the Algorithm

Before modeling a knowledge graph, the entities and relationships within it must be converted into vector representations. This conversion is crucial because it allows us to use vector space distance and similarity measures to assess relationships between entities and infer connections between unknown entities—this process is known as knowledge graph embedding.

As illustrated in Fig. 3, the entity set in the knowledge graph is first encoded as text. These encoded entities are then mapped to corresponding sequence numbers, which typically start from 0 and increment sequentially. A dictionary is constructed to link each sequence number to an entity in the knowledge graph. During model inference, these sequence numbers are consistently used to process entity information. Additionally, a dictionary is created to map sequence numbers to vector representations, ensuring that each sequence number corresponds to a low-dimensional vector representing the final entity.

For the edge set in the knowledge graph, each edge type e_i is transformed into a relation r_i. Similar to entities, a dictionary is created to map relations to sequence numbers, and another dictionary links these sequence numbers to relation vectors. Through this process, the knowledge graph is transformed into $G = (V, \varepsilon, R)$, converting the knowledge within the graph into vector representations that can be used as inputs for subsequent algorithms.

B. Node Information Aggregation based on Node Cross-Relational Attention Mechanism

After transforming the entities and relationships in the knowledge graph into vectors, the structure of graph neural network is used to aggregate the information of the nodes to realize the task of link prediction in the existing knowledge graph. The entire knowledge graph is represented as a set of $G = (V, \varepsilon, R)$ vectors, where $\{V\} = \{1,2,3,\ldots,n\}$ is the set of all the entities in the knowledge graph, $\{\varepsilon\}$ is the set of all the edges, and $\{R\}$ is the set of relationships. Assuming that the head node of the triad is $v_i \in V$, the tail node is $v_j \in V$, and the relation between nodes is $r \in R$, then this triad can be represented as $(v_i, r, v_j) \in \varepsilon$.

In GCN, the embedding vector of a node can be computed by using the Eq. (1).

$$h^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}h^{(l)}W^{(l)}\right) \quad (1)$$

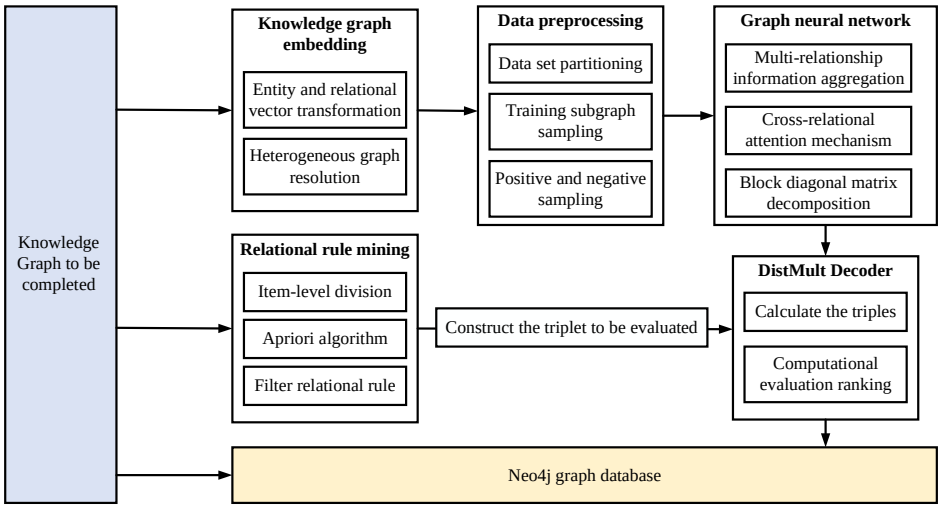

Fig. 2. Algorithm architecture diagram.

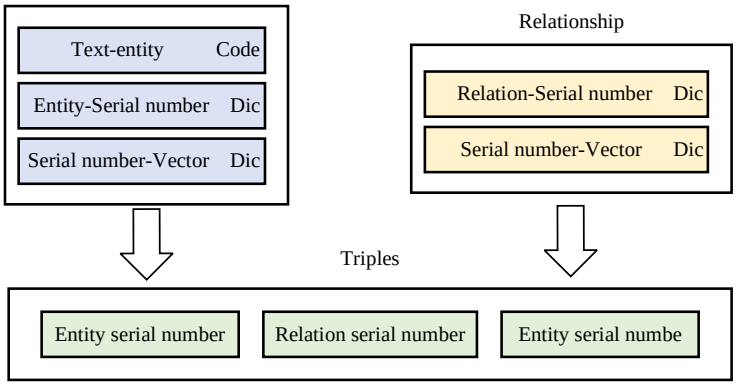

Fig. 3. Schematic of the conversion of knowledge graph triples to vectors.

where, $h^{(l)}$ is the node embedding vector of layer l, $W^{(l)}$ is the trainable weight matrix of layer l, $\tilde{A} = A + l$is the sum of the adjacency and self-connectivity matrices of the graph, $\widetilde{D}$ is the degree matrix of $\tilde{A}$, and σ is the activation function.

Based on Eq. (1), the process of GCN information aggregation can be expressed by Eq. (2).

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in N_i} \frac{1}{\sqrt{d_i}d_j} h_j^{(l)} W^{(l)}\right) \quad (2)$$

where, $h_i^{(l+1)}$ denotes the embedding vector of node i at layer $l+1$, $W^{(l)}$is the trainable weight matrix, N_i is the set of neighbors of node i, $\widetilde{d}_i$ and $\widetilde{d}_j$ are the degrees of node i and node j in matrix $\tilde{A}$, respectively.

As can be seen from Eq. (2), in GCN, the information of nodes converges in different relations sharing the same weight matrix W, so GCN actually treats the graph as a homomorphic graph for processing, in this study, we will consider the heteromorphism in the knowledge graph, and we have already transformed the entities and relations in the knowledge graph into vectors in the above paper and constructed the mapping between the serial numbers and vectors of the entities and relations, using the correspondence between serial number and vector, which makes it easier to split the knowledge graph into different heteromorphic graphs for processing according to different relations, which is reflected in the fact that different relations construct different maps between entities and relations. Using the correspondence between the serial number and the vector, we can easily split the knowledge graph into different heterogeneous graphs according to different relationships, which is reflected in the construction of different weight matrices for different relationships, instead of using the same weight matrix for all relationship types as in the case of GCN.

On the basis of GCN, considering the heterogeneity of the graph, the information is aggregated for different types of edges between nodes, as shown in Eq. (3).

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^N} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right) \quad (3)$$

Where, $h_i^{(l+1)}$ denotes the embedding of node i in the $l+1$ layer, σ is the activation function, R is the set of relations, $N_i^r$is the set of neighboring nodes of node i under relation r, $c_{i,r}$is the normalization constant, and $W_r^{(l)}$ is the weight matrix of relation r. Since the information of nodes themselves needs to be taken into account, the self-loop of nodes is introduced. The self-loop of a node can be treated as a special type of edge, and the self-loop weight matrix is denoted by $W_0^{(l)}$.

As can be seen from Eq. (3), for each relationship, a separate weight matrix W_r is computed. In large-scale knowledge graphs, due to the large number of relationships, the number of parameters in the inference process is also unusually large, which further increases the complexity of the inference process. Specifically, assuming that a graph G contains N nodes and M relationship types, each relationship type has a corresponding weight matrix W_r, size $D_l \times D_{l-1}$, where D_l is the feature dimension of the current layer of the relationship type, and D_{l-1} is the feature dimension of the previous layer. Since each node has different embedding vectors under different relation types, these weight matrices need to be computed separately at each node, resulting in an extremely large number of parameters.

In order to reduce the complexity of the model and improve the generalization performance, the diagonal decomposition of the weight matrix is performed, which transforms the complex weight matrix into a block diagonal matrix, thus reducing the number of parameters and improving the generalization performance of the model. Specifically, as shown in Eq. (4), where the size of $Q_{br}^{(l)}$ matrix is $\left(d^{(l+1)}/B\right) \times \left(d^{(l)}/B\right)$, the block diagonal decomposition of the weight matrix through a series of matrices of the summation, which greatly reduces the size of the parameters of the weight matrix and simplifies the computation process.

$$W_r^{(l)} = \bigoplus_{b=1}^{B} Q_{br}^{(l)} = diag(Q_{1r}^{(l)}, \ldots, Q_{Br}^{(l)}) \quad (4)$$

In the above node information aggregation process, each relation type uses an independent weight matrix for information transfer, and the node embedding is obtained by splicing the information of each relation type. This approach may lead to the uneven contribution of each relationship type in node embedding, and the contribution of some relationship types may be masked or weakened. To solve this problem, this study introduces the Node-level Across Relation Attention mechanism, which adjusts the weight of each relation type in the node

embedding by learning a node-level attention vector for each node.

Before calculating the node's attention mechanism, firstly, for the similarity between node i and node j under a specific relation r, the weight matrix corresponding to relation r is multiplied and spliced with node i and node j, respectively, and then multiplied with a trainable attention vector, and then the similarity between node i and node j under relation r is finally obtained by an activation function. As shown in Eq. (5), where e_{ij}^r denotes the similarity between node i and node j under relation r, $\vec{a}^r$ is a trainable attention vector, $\vec{W}_r$ is the weight matrix under relation r, and $\oplus$ denotes the vector splicing operation.

$$e_{ij}^r = \mathrm{ReLU}\left(\vec{a}^r \cdot \left[\vec{W}_r h_i^{(1)} \oplus \vec{W}_r h_j^{(1)}\right]\right) \quad (5)$$

The calculation process of attention is shown in Fig. 4. The final information aggregation process of a single node is shown in Fig. 5.

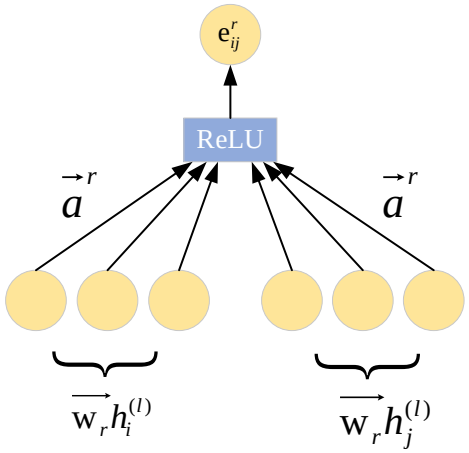

Fig. 4. Attention calculation process.

Based on Eq. (5), in the iterative process of graph neural network, each node in the knowledge graph calculates the similarity of its neighboring nodes under all the relationships, and then uses the similarity to calculate the attention of the node's neighboring nodes, and ultimately the information convergence of nodes in each layer is shown in Eq. (6). The whole information aggregation process is shown in Fig. 6, and finally the whole network outputs the embedding vector representation of nodes and relationships.

$$h_i^{(1+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r}\frac{exp\left(\mathrm{ReLU}\left(\vec{a}^r\cdot\left[\vec{W}_r h_i^{(l)}\oplus\vec{W}_r h_j^{(1)}\right]\right)\right)}{\sum_{k\in N_i^r} exp\left(\mathrm{ReLU}\left(a^r\cdot\left[\vec{W}_r^h h_i^{(l)}\oplus\vec{W}_r h_k^{(1)}\right]\right)\right)} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right) \quad (6)$$

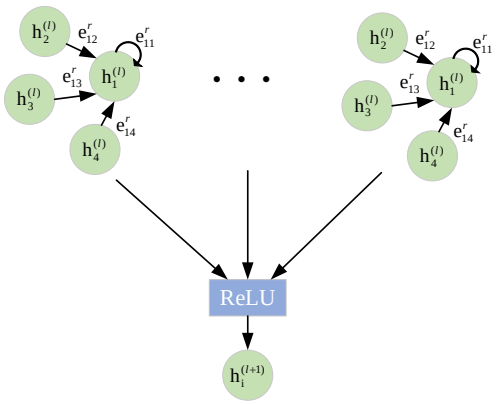

Fig. 5. Individual node final information aggregation process.

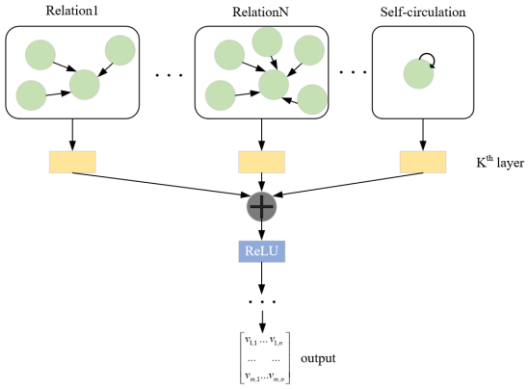

Fig. 6. Schematic diagram of node information aggregation process.

C. Training Sampling and Loss Functions

In the link prediction task, dealing with an entire knowledge graph can be challenging due to its size, often exceeding the capacity of a single GPU's memory. To address this, we sample smaller subgraphs for training. By using random sampling, we can maintain the integrity of the graph's structure while avoiding memory bottlenecks associated with processing large-scale knowledge graphs. This study employs uniform random sampling, where each node or edge is selected with equal probability. This method is straightforward, offering good randomness and repeatability, and can be executed using a random number generator.

The sampling process involves the following steps:

1) Determine the sample size: Set a hyperparameter, batch_size, to define the number of nodes or edges to sample.

2) Initialize the sample set: Create a set to store the sampled nodes or edges.

3) Random sampling: For each node or edge, generate a random number; if it falls below the sampling probability, include the node or edge in the sample set.

4) Return the sample set: Use this set for subsequent training or other tasks.

After random sampling, a training subgraph is obtained. To further enhance prediction accuracy and generalization, the study introduces positive and negative sampling. For each node pair (i, j), both positive samples (existing links) and negative samples (non-existent links) are generated. Negative samples are constructed using a random sampling strategy, with a negative sampling factor μ, where the ratio of positive to negative samples is $1:\mu$.

The loss function for the final model training is shown in Eq. (7) as follows.

$$L = -\sum_{(i,j)\in E} log\,\sigma\,(y_{(i,j)}) - \sum_{(i,j)\in \tilde{E}} log\,\sigma\,(\tilde{y}_{(i,j)}) \quad (7)$$

Where, E denotes the set of positive sample edges, E denotes the set of negative sample edges, and $y_{(i,j)}$ denotes the predicted probability of the existence of an edge between node *i* and node *j*, which can be computed by Eq. (8).

$$y_{(i,j)} = \frac{1}{k}\sum_{}^{t=1}{}^{k} \sigma(h_i^{(L)} \cdot r_i^{(L)} h_j^{(L)}) \quad (8)$$

where *k* denotes the number of negative samples sampled, $h_i^{(l)}$ and $h_j^{(l)}$ denote the embedding vectors of node *i* and node *j* in the last layer, and $r_t^{(L)}$ denotes the weight vector of relation type r_t in the Lth layer. σ denotes the sigmoid function, which is used to map the predicted values to a range between 0 and 1. The first term of the loss function is the cross-entropy loss for positive samples, indicating that the higher the probability that the model predicts the existence of positive edges, the lower the loss. The second term is the cross-entropy loss of negative samples, which means that the higher the probability that the model predicts that the negative sample edge does not exist, the lower the loss. By minimizing the loss function, it allows the model to predict the likelihood of edge existence more accurately between nodes.

V. Comparative Experiments and Analysis

This section presents comparative experiments to evaluate the model's effectiveness, focusing on the experimental environment, dataset, evaluation criteria, control group design, and final result analysis to validate the improvements made in this study.

A. Experimental Environment and Datasets

Due to the large knowledge graph dataset used in this experiment and the complexity of model computation, the experiment needs to be carried out on GPUs. The server used for the experiment is the one equipped in the lab, Windows 10 64bit system, 16GB RAM and equipped with high-computing-power GPUs. In terms of software, Python was used as the programming language for the model, PyCharm was used to compile the software, and PyTorch was used as the basic implementation library for the model.

This paper presents a custom-built knowledge graph dataset named BadmintonKG, specifically created for the badminton domain. It is designed to support applications such as tactical reasoning, match analysis, and training assistance. BadmintonKG includes 9,742 entities, 135 types of relationships, and 198,563 triples (i.e., relationships between entities). Unlike other common knowledge graph datasets, BadmintonKG focuses on the specific domain of badminton, covering various entities and relationships such as players, tactics, courts, techniques, and coaching styles. The training, validation and test sets are divided as shown in Table I.

TABLE I. BadmintonKG Data Set Delineation Table

Dataset segmentation	Entities	Relations	triples
Original dataset	9,742	135	198,563
Training set	9,742	135	176,421
Validation set	9,742	135	19,989
Test set	9,742	135	11,072

B. Evaluation Criteria and Control Group Design

MRR (Mean Reciprocal Rank) is one of the commonly used evaluation metrics in the task of knowledge graph link prediction. The core idea of MRR is to find the true tail entity *t* among all possible tail entities *t`* for each test ternary (h,r,t), and compute the inverse of its score ranking. Finally, the average of the ranked inverts of all the test triples is used as the MRR score of the model. This is calculated as shown in Eq. (9).

$$MRR = \frac{1}{|T|}\sum_{(h,r,r)\in T}\frac{1}{\text{rank}_{(hr,t)}} \quad (9)$$

In knowledge graph link prediction, Hits@ N is a commonly used evaluation metric to measure whether the algorithm can correctly predict the correct entity or relation in the test set among the first N candidate entities or relations. Specifically, assuming that the correct answer for each ternary (h,r,t) in the test set is *t*, then for each (h,r) pair, we can sort all its possible entities according to the algorithm's prediction scores in descending order and compute whether the first *N* predicted entities contain the correct answer *t*. If the correct answer is *t*, then it is called a hit. If it does, it is called a hit (hit), otherwise it is called a miss. The final hit rate is the average hit rate of all test triples among the first N candidate entities. This is shown in Eq. (10).

$$\text{Hits@}N = \frac{1}{|T|}\sum_{(h,r,t)\in T}\Pi(\text{rank}_{(h,r,t)} \le N) \quad (10)$$

To validate the effectiveness of the proposed knowledge graph inference algorithm based on graph neural networks and association rule mining, experiments will be conducted on BadmintonKG. The experimental model will be compared with three other neural network models: Graph Convolutional Network (GCN), Graph Attention Network (GAT), and Relational Graph Convolutional Network (R-GCN). The comparison will focus on the evaluation metrics MRR and Hits@N, and the impact of different training subgraph sizes on model performance will also be assessed.

Knowledge Graph Link Prediction with GCN: GCN is a graph convolutional neural network that treats the knowledge graph as a homogeneous graph, using node neighbors to perform convolution. It aggregates neighbor information with a weighted adjacency matrix, enabling it to learn node representations through multi-layer convolution. GCNs are particularly effective for graphs with similar node features.

Knowledge Graph Link Prediction with GAT: GAT extends GCN by incorporating an attention mechanism that weights the importance of each neighboring node. This allows for the aggregation of neighbor information with varying degrees, resulting in richer and more accurate node representations.

Knowledge Graph Link Prediction with R-GCN: R-GCN is designed for multi-relational graphs, using relational matrices in its convolution operations. Unlike traditional GCNs, R-GCN employs a learnable convolution kernel that adjusts to different types of relations, making it more effective for handling multi-relational data.

C. Experimental Results and Analysis

In this control group, this experimental model is compared with the other two models on two datasets, and the number of model iterations for the experiments is set to 6000, the stochastic inactivation rate is set to 0.2, the learning rate is set to 0.01, the output dimension of the hidden layer is set to 500, and the sampling mode of the training subgraphs is set to uniform. The test set evaluation results from the above training results are plotted in a table, as shown in Table II.

The above table was plotted as a line graph as shown in Fig. 7 and 8. From the above experimental data performance, the model in this study shows excellent performance on both datasets, and outperforms the other three models in all indicators.

In the BadmintonKG dataset, when the size of the training subgraph is 30000, this model outperforms the GCN, GAT, and R-GCN models in all the metrics, but the difference is relatively small, and when the size of the training subgraph is 80000, the MRR metrics of this model reaches 0.2753, which is higher than that of 0.1910 for the GCN, GAT, and R-GCN models, 0.2503 and 0.2653, which is about 40% higher than that of the GCN model, 10% higher than that of the GAT model, and 3.8% higher than that of the R-GCN model.

TABLE II. SUMMARY OF EXPERIMENTAL RESULTS

BadmintonKG	The size of the training subgraph is 30000				The size of the training subgraph is 80,000			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Our	0.2458	0.1576	0.2677	0.4249	0.2753	0.1811	0.3066	0.4638
GCN	0.1673	0.0873	0.1927	0.3201	0.1910	0.0990	0.2283	0.3684
GAT	0.2144	0.1283	0.2373	0.3855	0.2503	0.1610	0.2695	0.4403
R-GCN	0.2412	0.1404	0.2531	0.4085	0.2653	0.1715	0.2856	0.4536

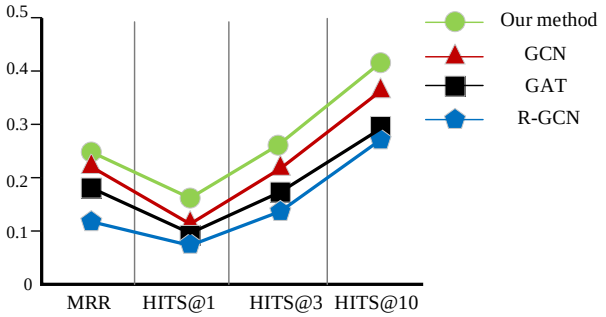

Fig. 7. BadmintonKG comparison of models (training subgraph size 30000).

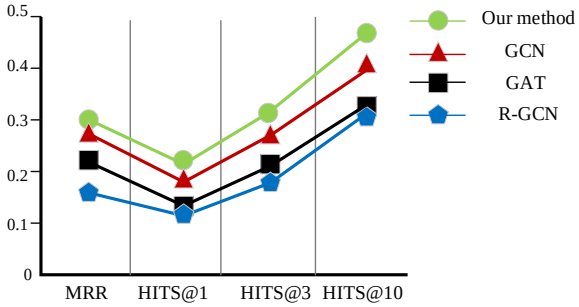

Fig. 8. BadmintonKG comparison of models (training subgraph size 80000).

Overall, the model in this study achieves better performance than the link prediction models based on GCN, GAT and R-GCN in the knowledge graph link prediction task, and shows good robustness and generalization ability, and the experimental results in this study prove the effectiveness and superiority of the proposed model in the knowledge graph link prediction task.

VI. CONCLUSION

In this study, we applied knowledge graph and graph neural network techniques to the mining and reasoning of badminton tactics, demonstrating the effectiveness of this approach in analyzing and optimizing players' training patterns. Technically, the introduction of training subgraph sampling, positive and negative sampling, and block diagonal matrix decomposition significantly reduced the computational complexity, improved the handling of large-scale knowledge graphs, and enhanced the model's generalization capabilities. These techniques not only optimized the data processing workflow but also improved the model's performance and accuracy in practical applications.

Through comparative experiments on a proprietary badminton tactics dataset, we validated the superiority of the proposed method over traditional approaches in tactical reasoning and training optimization. The results showed that our model more accurately predicted and reasoned about tactical changes in matches, offering strong scientific support for badminton training and competition. In conclusion, this study provides a novel technical approach for tactical analysis in badminton and offers a potential methodological reference for other sports.

ACKNOWLEDGMENT

Thanks to all the people and organizations that contributed to this study.

REFERENCES

[1] Fu T, Li P, Liu S. An imbalanced small sample slab defect recognition method based on image generation[J]. Journal of Manufacturing Processes, 2024, 118: 376-388.

[2] Fu T, Liu S, Li P. Intelligent smelting process, management system: Efficient and intelligent management strategy by incorporating large language model[J]. Frontiers of Engineering Management, 2024: 1-17.

[3] Schatz, A. Pro Football Prospectus: Statistics, Analysis, and Insight for the Information Age[M]. 2006 Edition. Workman Publishing Company,2006.

[4] Coleman, J.&A. Lynch. NCAA Men's Basketball Tournament Score Card[EB/OL].

[5] Fu T, Liu S, Li P. Digital twin-driven smelting process management method for converter steelmaking[J]. Journal of Intelligent Manufacturing, 2024: 1-17.

[6] Damien Demaj.Using spatial analytics to study spatio-temporal patterns in sport[EB/OL]. http://blogs.esri.com/esri/arcgis/2013/02/19/using-spatial-analytics-to-study-spatio-temporal -patterns-in-sport.

[7] Maheswaran R,Chang Y H,Henehan A,et al.Deconstructing the rebound with optical tracking data[C].The MIT Sloan Sports Analytics Conference.Boston,2012.

[8] Carlson A, Betteridge J, Kisiel B, et al. Proceedings of the Twenty-Fourth AAA I Conference on Artificial Intelligence (AAAI-10) Toward an Architecture for NeverEnding Language Learning[JI]. 2011.

[9] Suda M, Weidenbach C, Wischnewski P . On the Saturation of YAGO[C]// Inter national Conference on Automated Reasoning. Springer-Verlag, 2010.

[10] Page L, Brin S , Motwani R, et al. The PageRank citation ranking: Bringing ord er to the Web. Technical report. 1919.

[11] Cohen W W . TensorLog: A Differentiable Deductive Database:, 10.48550/arXiv. 1 605.06523[P].2016.

[12] L Bühmann, Lehmann J . Pattern Based Knowledge Base Enrichment[C]// 12th I nternational Semantic Web Conference, 21-25 October 2013, Sydney, Australia. Springer, Berlin, Heidelberg, 2013.

[13] Jiang S , Lowd D , Dou D . Learning to Refine an Automatically Extracted Kno wledge Base Using Markov Logic[C]// IEEE International Conference on Data Minin g;ICDM2012. 0 .

[14] Pujara J , Miao H , Getoor L, et al. Knowledge Graph Identification[C]// Intern ational Semantic Web Conference. Springer-Verlag New York, Inc. 2013.

[15] Zheng H, Liu S, Zhang H, et al. Visual-triggered contextual guidance for lithium battery disassembly: a multi-modal event knowledge graph approach[J]. Journal of Engineering Design, 2024: 1-26.

[16] Kimmig A, Bach S , Broecheler M , et al. A Short Introduction to Probabilistic Soft Logic[J]. Dec-2012, 2013.

[17] Chen Y , Goldberg S , Wang D Z, et al. Ontological Pathfinding[C]// the 2016 I nternational Conference. ACM, 2016.

[18] Bordes A, Usunier N , Garcia-Duran A, et al. Translating Embeddings for Mod eling Multi-relational Data[C]// Neural Information Processing Systems. Curran Asso ciates Inc. 2013.

[19] Zhen W , Zhang J , Feng J , et al. Knowledge Graph Embedding by Translating on Hyperplanes[C]// National Conference on Artificial Intelligence. AAAI Press, 201 4.

[20] Nickel M , Tresp V , Kriegel H P. A Three-Way Model for Collective Learning on Multi-Relational Data[C]// International Conference on International Conference o n Machine Learning. Omnipress, 2011.

[21] Chang K W , Yih W T , Yang B, et al. Typed Tensor Decomposition of Knowle dge Bases for Relation Extraction[J]. 2014.

[22] Liu S, Zheng P, Xia L, et al. A dynamic updating method of digital twin knowledge model based on fused memorizing-forgetting model[J]. Advanced Engineering Informatics, 2023, 57: 102115.