Complex-Path: Effective and Efficient Node Ranking with Paths in Billion-Scale Heterogeneous Graphs

Jinquan Hang
JD Logistics
Rutgers University
jinquan.hang@rutgers.edu

Zhiqing Hong
Rutgers University
zhiqing.hong@rutgers.edu

Xinyue Feng
Rutgers University
xinyue.feng@rutgers.edu

Guang Wang*
Florida State University
guang@cs.fsu.edu

Dongjiang Cao
JD Logistics
caodongjiang1@jd.com

Jiayang Qiao
JD Logistics
qiaojiayang1@jd.com

Haotian Wang
JD Logistics
wanghaotian18@jd.com

Desheng Zhang
Rutgers University
desheng@cs.rutgers.edu

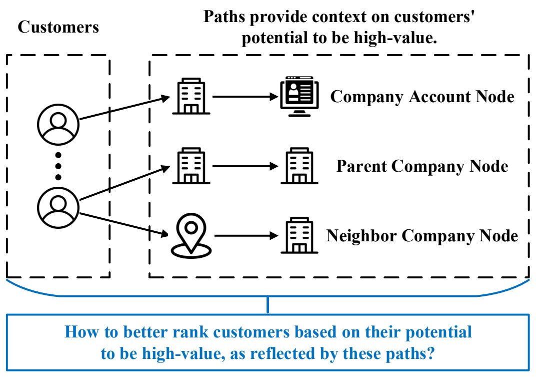
ABSTRACT

Node ranking in heterogeneous graphs, which quantifies the relative importance of nodes, can often be improved by incorporating information from relevant paths. Graph database and heterogeneous graph neural network (HGNN) are two main approaches to better solve this problem. Graph databases support efficient path queries for flexible path types but require manual design to combine results for node ranking. Conversely, current HGNNs can automatically integrate semantic information from multiple linear path types for accurate node ranking. However, our experiments show that they fail to outperform a multi-layer perceptron model that utilizes features extracted from multiple nonlinear conditional paths, which can be handled by graph databases. Therefore, we aim to enable HGNN to take advantage of these path types for better performance. However, HGNNs require a generalized path schema to define the structure of input paths, and incorporating each additional path type will significantly increase the required system memory and sampling time for HGNNs. To address these limitations, we introduce CompNode, a novel framework based on a new unified path schema definition called Complex-path, which is used to describe all the required path types, including nonlinear conditional path types. Then, we design a pre-aggregation method to reduce the required system memory and sampling time by pre-aggregating the same type of complex-path. Furthermore, we develop a model that combines semantic information from all aggregated complex-paths for accurate node ranking. Real-world experiments on identifying top potential high-value customers show CompNode outperforms state-of-the-art HGNNs by 20% in average precision and the previously deployed graph database method by 252% in success rate.

PVLDB Reference Format:
Jinquan Hang, Zhiqing Hong, Xinyue Feng, Guang Wang, Dongjiang Cao, Jiayang Qiao, Haotian Wang, and Desheng Zhang. Complex-Path: Effective and Efficient Node Ranking with Paths in Billion-Scale Heterogeneous Graphs. PVLDB, 17(12): 3973 - 3986, 2024.
doi:10.14778/3685800.3685820

*Guang Wang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685820



Customers
Paths provide context on customers' potential to be high-value.
Company Account Node
Parent Company Node
Neighbor Company Node
How to better rank customers based on their potential to be high-value, as reflected by these paths?

Figure 1: Identifying top potential high-value customers in a billion-scale heterogeneous graph.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/JQHang/Complex-path.

1 INTRODUCTION

Recently, node ranking in heterogeneous graphs [56, 64, 67] has become increasingly important. Node ranking [10, 53, 57] is the process of assigning scores to nodes based on their relevance or significance within the graph. Heterogeneous graphs [7, 54, 74] contain multiple types of nodes (entities) and edges (relations), which pose unique challenges for node ranking. In such graphs, a node's importance is often influenced by its relevance to different types of paths connecting it with other nodes, which capture the semantic relations and structural properties of the graph. Numerous research efforts have focused on effectively leveraging information from diverse paths in heterogeneous graphs. They have achieved notable success in fields such as recommendation systems [9, 80], fraud detection [20, 48, 61], and social network [32, 72, 79]. In this paper, we aim to address an industrial node ranking problem: **ranking customers based on the potential of being high-value**, as shown in Figure 1. Our goal is to leverage information from various useful paths in the graph to help sales accurately find top potential high-value customers. This node ranking problem is critical because finding the appropriate customer [37, 44, 45] to contact is the initial step for service providers to promote their products.

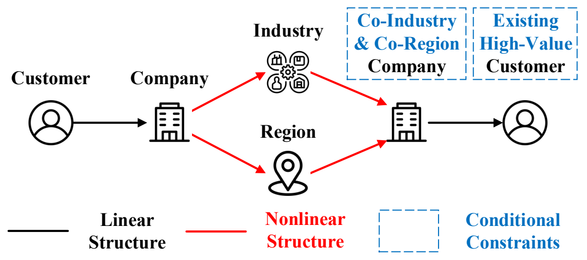


Figure 2: An example of nonlinear conditional path type. (Link each customer to existing high-value customers via co-industry and co-region relations.)

Two methods, graph database [3, 4] and heterogeneous graph neural network (HGNN) [13, 80], can potentially address our problem and identify top potential high-value customers. Graph databases can efficiently query paths [1, 40] matching flexible path types and use manually designed algorithms to calculate each node's final importance value based on the matching results through graph query languages (GQL). However, accurately estimating customers' potential value through a manually designed algorithm is challenging, as it is influenced by numerous conditions. In contrast, HGNNs [15, 19] can automatically learn to integrate the semantic information from multiple paths based on labeled nodes, often outperforming manually designed algorithms in complex tasks like product recommendations [9]. However, HGNNs can only learn from paths that conform to a generalized path schema [35]. The currently mainly used generalized path schema, Meta-path [55], can only represent *linear path types* composed of a sequence of nodes and edges, which overlooks some *nonlinear conditional path types* used by graph databases, such as path types involving fork-join or conditions in the path structure, as shown in Figure 2.

To verify the importance of these nonlinear conditional path types, we first transform some selected nonlinear conditional paths into corresponding features for each customer. For example, the path type in Figure 2 can be transferred into several features for each customer, including the number of existing high-value customers that can be connected through this path type and their average order count. We then concatenated these features with each customer's original features to train a multi-layer perceptron (MLP) [50] model based on labeled customers and compared it with several current state-of-the-art HGNNs. The results show that the MLP model outperformed all current HGNNs, demonstrating the value of information from these nonlinear conditional path types and the inability of current HGNNs to automatically utilize them. Moreover, an MLP model using only features transformed from meta-paths performed worse than most HGNNs, indicating that HGNNs can better utilize path structure information.

Therefore, introducing nonlinear conditional path types into HGNN can more effectively utilize information in heterogeneous graphs. This is, however, not trivial due to two major challenges.

- *(i) Generalized Path Schema.* To leverage HGNNs for learning semantic information from different path types, all paths must follow a generalized path schema based on the heterogeneous graph definition. However, path types expressed by GQL are based on the operators provided by graph databases and don't have a generalized path schema.
- *(ii) Scalability and Computational Efficiency.* For most existing HGNNs [19, 32, 65], incorporating each additional path type incurs a considerable increase in sampling time and system memory due to the rapid growth of the number of neighbors [18, 46, 70]. Consequently, it is challenging to simultaneously utilize a large number of path types on large-scale data.

To tackle these challenges, we developed a framework called CompNode. First, to address the lack of a generalized path schema, we introduced a new definition named Complex-path, which optimizes the existing Meta-path schema to represent all types of nonlinear conditional paths that we require. Second, to improve scalability and computational efficiency, we first developed a pre-aggregation method that can integrate with distributed systems to utilize more machines to complete the process of aggregating the same type of complex-path in advance, and designed a model based on the aggregated paths that can simultaneously consider intra-path and inter-path information for accurate node ranking. Consequently, CompNode can effectively and efficiently leverage a large number of selected complex-paths to rank customers based on their potential to become high-value.

In summary, this paper makes the following contributions.

- To the best of our knowledge, we are the first to study the problem of ranking nationwide customers based on their potential value using a billion-scale heterogeneous graph.
- We introduce a new framework, CompNode, which can leverage nonlinear conditional paths represented by Complex-path for efficient and effective node ranking in heterogeneous graphs.
- We performed comprehensive experiments using real-world datasets against several baselines. Our analysis, both theoretical and empirical, indicates that our framework outperforms baseline models by over 20% in average precision, maintaining its efficiency even with the increase of complex-paths.
- Our framework has been deployed at a major logistics company. In real-world A/B testing, it achieved a 252% increase compared to the previous deployed graph database strategy. After the testing, our framework has continued to run smoothly and has identified over 200,000 top potential high-value customers.

2 PRELIMINARY

2.1 Graph Databases

In graph databases, data is often structured in Labeled Property Graphs (LPGs) [2, 8]. LPGs enable nodes and edges to carry multiple labels and properties, with labels categorizing them into broad types and properties providing detailed information.

Definition 2.1. **Labeled Property Graph (LPG).** An LPG is a graph composed of nodes (entities) and edges (relations), defined as $LPG = (V, E, \Lambda, P)$. In an LPG:

- Each node $v_i \in V$ and edge $e_{ij} \in E$ is tagged with one or more labels via Λ, where $\Lambda : V \cup E \longrightarrow L$ and L is the set of labels.
- Each node and edge is also associated with properties, expressed as key-value pairs, through P, where $P : (V \cup E) \times K \longrightarrow Values$, K represents the keys, and $Values$ the associated values.

```
1:MATCH (c1:Customer)-[:RELATED_TO]->(cc1:Company),
2:      (c2:Customer)-[:RELATED_TO]->(cc2:Company),
3:      (cc1)-[:BELONGS_TO]->(i:Industry),
4:      (cc2)-[:BELONGS_TO]->(i),
5:      (cc1)-[:LOCATED_IN]->(r:Region),
6:      (cc2)-[:LOCATED_IN]->(r)
7:WHERE r IS NOT NULL AND i IS NOT NULL
      AND c2.value = 'high'
8:RETURN c1, cc1, r, i, cc2, c2
```

Listing 1: The Cypher query corresponding to Figure 2.

To obtain the corresponding paths [5, 6, 14] from the graph database based on path types, different graph databases have developed their unique graph query languages. For instance, Neo4j uses Cypher [12], Oracle uses PGQL [47], and TigerGraph uses GSQL [22]. These query languages offer a variety of operations, allowing users to define diverse path patterns.

For example, the path shown in Figure 2 corresponds to the Cypher query presented in Listing 1. Lines 1-2 represent the relationships between customer nodes c1 and c2 and their respective corresponding company nodes cc1 and cc2 through the RELATED_TO relationship. Lines 3-4 state that the company nodes cc1 and cc2 must be connected to the same Industry node i through the BELONGS_TO relationship. Lines 5-6 state that the company nodes cc1 and cc2 must be connected to the same Region node r through the LOCATED_IN relationship. Line 7 states that the relationships of being in the same Industry and same Region must be simultaneously satisfied and the value property of the customer node c2 must be 'high'.

2.2 Heterogeneous Graph Neural Network

Although graph databases offer high flexibility in querying flexible path types, achieving high accuracy can be challenging when the calculation of nodes' importance values cannot be precisely described manually. In contrast, by using labeled nodes, HGNNs [32, 63] can automatically and effectively integrate information from different types of subgraphs or paths. However, because HGNNs need to use a unified neural network structure [64, 67] to synthesize information, there are additional constraints on the construction of the graph, and the sampled subgraphs or paths must follow a generalized schema. In most cases, the heterogeneous graphs in HGNNs follow the definition below.

Definition 2.2. **Heterogeneous Graph (HG).** A heterogeneous graph is a graph consisting of different types of entities (i.e., nodes) and/or different types of relations (i.e., edges), each type having a fixed number of features (i.e., properties), which can be formally defined as $HG = (V, E)$. Specifically, within HG:

- Each node $v_i \in V$ is associated with one node type $o = \phi(v_i)$, $o \in O$, and a feature vector $\mathbf{f}_{v_i} \in \mathbb{R}^{d_o}$.
- Each edge $e_{ij} \in E$ is associated with one edge type $l = \psi(e_{ij})$, $l \in L$, and a feature vector $\mathbf{f}_{e_{ij}} \in \mathbb{R}^{d_l}$.
- There is a heterogeneity in the types of nodes or edges, represented as $|O| + |L| > 2$.

Compared to LPG, HG requires that each node and edge has exactly one type, and nodes and edges of the same type must have

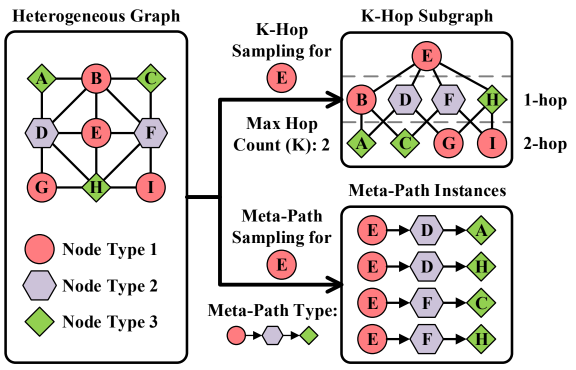


Figure 3: Two sampling methods in heterogeneous graphs.

the same set of features. Based on this definition, there are mainly two sampling methods for HGNNs: *k-hop sampling* and *meta-path sampling*. Figure 3 shows an example of these two methods.

The first method is k-hop sampling, primarily used to collect neighbor nodes within k hops from the target node, as well as the edges connecting these neighbors to their preceding nodes in the hop sequence. The definition of k-hop sampling is:

Definition 2.3. **K-hop Sampling.** Given a graph $HG = (V, E)$, a target node v_0, and a maximum number of hops k, k-hop sampling refers to the process of returning a subgraph $HG' = (V', E')$ where V' includes v_0 and all nodes v that are reachable from v_0 within k hops, E' includes all edges that connect nodes within V'.

However, due to the over-smoothing issue [33, 66], information from too many hops become indistinguishable, limiting its ability to leverage long-distance information. To address this, meta-path sampling was introduced. Meta-path [55], by defining a sequence of nodes and edges, more precisely connects distant nodes to the target node, effectively reducing the over-smoothing problem and enabling the model to utilize a broader range of information. The definitions of meta-path and meta-path sampling are:

Definition 2.4. **Meta-path.** A meta-path p is a path denoted in the linear form of $o_1 \xrightarrow{l_1} o_2 \xrightarrow{l_2} \cdots \xrightarrow{l_{m-1}} o_m$, where o_i and l_i are node types and edge types, respectively, and subscripts indicate the order of appearance for nodes or edges.

Definition 2.5. **Meta-path Sampling.** In a heterogeneous graph $HG = (V, E)$, given a target node v_0 and a meta-path p, meta-path sampling identifies all sequences of nodes and edges starting from v_0 that follow the pattern specified by p. These sequences are called meta-path instances.

2.3 Complex-Path

Although meta-paths can capture long-distance linear path types (see Definition 2.4), they struggle to represent some nonlinear conditional path types, which are important in real-world rich-content commercial graphs as shown in Fig 2. Using graph query language, we extracted information corresponding to these nonlinear conditional path types and discovered a strong correlation with our

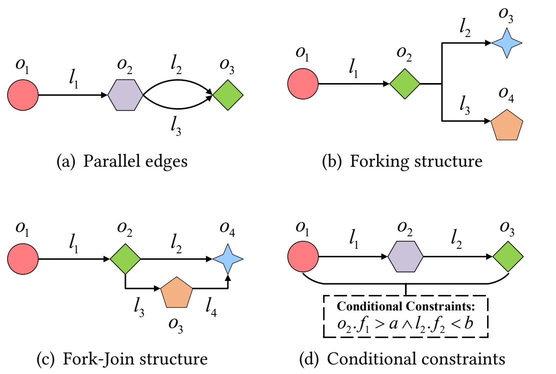


Figure 4: Four examples of nonlinear conditional paths.

labels. However, graph query languages usually represent paths using operations provided by graph databases (e.g., (), ->, Where), such as the example shown in Listing 1. Although this is very flexible, most of them need a parser to transfer the language into the path structure, while meta-paths can let HGNNs directly capture the path structure, like the order of nodes or edges and their type. Therefore, our goal is to refine the Meta-path definition to convey nonlinear conditional paths while maintaining a generalized path schema suitable for HGNNs. We proposed four improvements based on the four situations illustrated in Figure 4.

- For the situation shown in Figure 4(a), we replace the single edge type l with an edge list L: $p = o_1 \xrightarrow{L_1} o_2 \xrightarrow{L_2} \cdots \xrightarrow{L_{m-1}} o_m$, where $L_i = [l_{i,1}, l_{i,2}, \cdots]$. Then, the path in Figure 4(a) can be represented as $p = o_1 \xrightarrow{[l_1]} o_2 \xrightarrow{[l_2, l_3]} o_3$.
- For the situation shown in Figure 4(b), we switch to using tuples to represent the relations at each hop, where the first element of a tuple indicates its connection to a prior node: $p = \big[(o_1, L_1, o_2), \cdots, (o_i, L_{j-1}, o_j), \cdots, (o_n, L_{m-1}, o_m)\big]$, where $i < j$. Then, the path in Figure 4(b) can be represented as $p = [(o_1, [l_1], o_2), (o_2, [l_2], o_3), (o_2, [l_3], o_4)]$.
- For the situation shown in Figure 4(c), we define the elements of the edge list L to be either edge type l or path type p: $p = \big[(o_1, L_1, o_2), \cdots, (o_i, L_{j-1}, o_j), \cdots, (o_n, L_{m-1}, o_m)\big]$, where $L_i = [\cdots, e_k, \cdots]$, $e_k = l_k$ or p_k. Then, Figure 4(c) can be represented as $p = [(o_1, [l_1], o_2), (o_2, [l_2, [(o_2, [l_3], o_3), (o_3, [l_4], o_4)]], o_4)]$.
- For the situation shown in Figure 4(d), we add the conditional constraints C to the path: $p = \{[(o_1, L_1, o_2), \cdots, (o_i, L_{j-1}, o_j), \cdots, (o_n, L_{m-1}, o_m)] \mid C\}$. Then, the path in Figure 4(d) can be represented as $p = \{[(o_1, [l_1], o_2), (o_2, [l_2], o_3)] | o_2.f_1 > a \wedge l_2.f_2 < b\}$.

We name this optimized definition 'Complex-path'. The final definition of Complex-path is:

Definition 2.6. **Complex-path.** A complex-path is defined on a heterogeneous graph $HG = (V, E)$ and is represented as $p = \{[(o_1, L_1, o_2), \cdots, (o_i, L_{j-1}, o_j), \cdots, (o_n, L_{m-1}, o_m)] \mid C\}$. Each tuple (o_i, L_{j-1}, o_j) connects the j-th node o_j to the prior i-th node o_i through any edge or complex-path in $L_{j-1} = [\cdots, e_k, \cdots]$, where $e_k = l_k$ or p_k, $i < j$. C is the conditional constraints on the attributes of all nodes and edges.

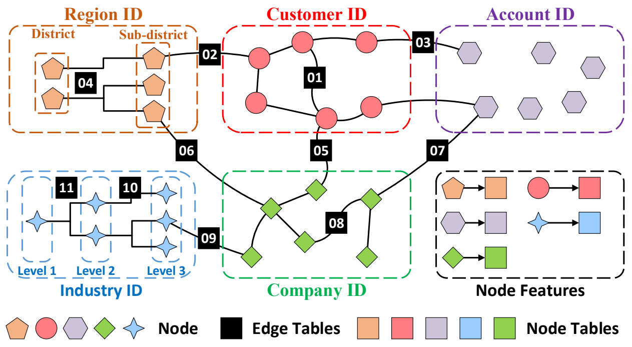


Figure 5: The overall structure of our graph.

Table 1: The feature count of each type of edge.

Edge Index	01	02	03	04	05	06	07	08	09	10	11
Feature Count	120	120	120	1	34	5	27	7	1	1	1

Table 2: The feature count of each type of node.

Node Type	Region	Customer	Account	Industry	Company
Feature Count	59	96	44	24	7

2.4 Problem Definition

Given a large-scale heterogeneous graph $HG = (V, E)$, a set of target nodes $V^{o^{\text{tgt}}}$ and a set of labeled target nodes $V^{\text{label}} \subset V^{o^{\text{tgt}}}$ with binary labels (0 for negative and 1 for positive), we aim to train a model using HG and V^{label} to rank all the target nodes $V^{o^{\text{tgt}}}$ based on the predicted positive probability. Specifically, our goal is to rank customers of a major logistics company based on their potential to become high-value customers.

3 GRAPH CONSTRUCTION

Our data, which mainly comes from a logistics company, can be represented as a heterogeneous graph with 5 types of **1 billion nodes** and 11 types of **10 billion edges**, as illustrated in Figure 5. Next, we will introduce the contents and structure of our graph.

3.1 Data Description

Our dataset contains two forms: edges (relations), and nodes (entities). We convert the associated attributes of these edges and nodes into numerical features. The feature counts for each type of edge and node are specified in Tables 1 and 2, respectively. Note that to enhance privacy, all key identifiers (e.g., customers and accounts) have been encrypted, we only keep the encrypted ID in the graph.

3.2 Graph Structure Design

Motivation. Traditional graph neural network platforms like PyG[17] and DGL[62] need to load the entire graph into memory, which cannot work well for our huge data. To address this challenge, we drew inspiration from some newer distributed platforms, such as AliGraph [69, 82] and Galileo [31], which construct graphs in the form of tables that can be accepted by distributed relational databases and then utilize operations supported by distributed systems to process large-scale heterogeneous graph data. Specifically, we used two types of tables to construct our graph: Edge Tables $\mathbf{E} = [\cdots, \mathrm{E}^{l^{\text{index}}}, \cdots]$ and Node Tables $\mathbf{V} = [\cdots, \mathrm{V}^{o^{\text{type}}}, \cdots]$, where $l^{\text{index}} \in \{l^{01}, l^{02}, \cdots, l^{11}\}$ indicates the edge type and $o^{\text{type}} \in \{o^{\text{Region}}, o^{\text{Customer}}, o^{\text{Account}}, o^{\text{Industry}}, o^{\text{Company}}\}$ indicates the node type. Below are the detailed formats of these tables:

- **Edge Table:** Each edge table, denoted as E^{l^i}, includes two columns that store the IDs of the head and tail nodes associated with each edge, identified by node types o^i and o^j. These columns are represented as $\mathrm{E}^{l^i}.o^i$ and $\mathrm{E}^{l^i}.o^j$, respectively. Additionally, the table contains d_{l^i} columns for edge features, collectively referred to as $\mathrm{E}^{l^i}.\mathbf{f}^{l^i}$, where d_{l^i} is the feature count for edge type l^i.
- **Node Table:** Each node table, denoted as V^{o^i}, comprises a column for storing the IDs of the nodes, which is referenced based on its node type o^i as $\mathrm{V}^{o^i}.o^i$. The table also includes d_{o^i} columns for node features, which are collectively referred to as $\mathrm{V}^{o^i}.\mathbf{f}^{o^i}$, where d_{o^i} is the feature count for node type o^i.

Additionally, we use a special node table $\mathrm{V}^{\text{label}}$ to store the labeled target nodes with type o^{tgt}, which is o^{customer} for our task. Each month, we used approximately 130,000 labeled nodes.

- **Labeled Node Table.** The labeled node table, denoted as $\mathrm{V}^{\text{label}}$, includes a column for storing the IDs of the labeled nodes, referred to as $\mathrm{V}^{\text{label}}.o^{\text{tgt}}$. In addition, it contains a column indicating the date when the sales contacted the customer associated with that node ID, represented as $\mathrm{V}^{\text{label}}.\mathit{date}$, and a column containing labels reported by the sales after the contact, denoted as $\mathrm{V}^{\text{label}}.\mathit{label}$. The value of $\mathrm{V}^{\text{label}}.\mathit{label}[i]$ is binary, with 1 indicating that the sales believes the i-th customer has the potential to become a high-value customer, and 0 indicating otherwise.

Data Leakage Prevention. To avoid using information obtained after sales have already contacted the customers, we constructed 12 heterogeneous graphs $HG = (\mathbf{E}, \mathbf{V})$, each based on data available up to the first day of each month, from 2022.10.01 to 2023.09.01. For every labeled node, we exclusively relied on the heterogeneous graph corresponding to the first day of the month specified by its contacted date $\mathrm{V}^{\text{label}}.\mathit{date}$ to gather information during training.

3.3 Complex-Path Implementation

Complex-Path Representation: First, based on our definition of complex-path, we can represent many nonlinear conditional path types in our graph that meta-paths cannot express. For example, the path type in Figure 2 can be represented as shown in Listing 2, where superscripts indicate the type of each node or edge, and subscripts indicate their position. For instance, o_5^{company} represents a node of type 'company' that appears fifth in the path.

Complex-Path Selection: Although our graph contains numerous complex-path types, most of them are not effective for our task. Therefore, we employed a data-driven approach to select useful complex-paths. We initially kept 534 types of complex-paths that start with the target node type o^{tgt} (o^{customer}) and have up to six edges. We then trained a model using all these complex-paths and

```
p = { [ ($o_1^{customer}$, [$l_1^{05}$], $o_2^{company}$),
        ($o_2^{company}$, [ [($o_2^{company}$, [$l_2^{09}$], $o_3^{industry}$),
                          ($o_3^{industry}$, [$l_3^{09}$], $o_5^{company}$)],
                         [($o_2^{company}$, [$l_4^{06}$], $o_4^{region}$),
                          ($o_4^{region}$, [$l_5^{06}$], $o_5^{company}$)] ], $o_5^{company}$),
        ($o_5^{company}$, [$l_6^{05}$], $o_6^{customer}$) ]
      | $o_3^{industry}$ != NULL ∧ $o_4^{region}$ != NULL
        ∧ $o_6^{customer}.f_{value}$ == 'high'}
```

Listing 2: The complex-path representation for Figure 2.

recorded the AUC results on the test set. Next, we iteratively removed each type of complex-path and retrained the model. If the AUC decreased by more than 0.001 after removal, we added the complex-path back. This process yielded a final set of 53 complex-path types that maintained the optimal performance.

Sampling Result Format: In later computations, we will sample numerous instances of each complex-path type. Each instance will be stored in a row of the path table corresponding to its specific complex-path type. The detailed format of path table is as follows:

- **Path Table.** Each path table P contains three types of columns: node ID, node features and edge features, which correspond to the nodes and edges appearing in the path. We use $\mathrm{P}.o^i$ to denote the node ID column corresponding to each node involved in a complex-path, $\mathrm{P}.\mathbf{f}^{o^i}$ to denote the feature columns from the node table $\mathrm{V}^{o^i}.\mathbf{f}^{o^i}$, $\mathrm{P}.\mathbf{f}^{l^i}$ to denote the feature columns from the edge table $\mathrm{E}^{l^i}.\mathbf{f}^{l^i}$. For example, for the complex-path represented as Listing 2, the columns in corresponding path table P will be $[\mathrm{P}.o_1^{\text{customer}}, \mathrm{P}.\mathbf{f}^{o_1^{\text{customer}}}, \mathrm{P}.\mathbf{f}^{l_1^{05}}, \mathrm{P}.o_2^{\text{company}}, \mathrm{P}.\mathbf{f}^{o_2^{\text{company}}}, \mathrm{P}.\mathbf{f}^{l_2^{09}}, \mathrm{P}.o_3^{\text{industry}}, \mathrm{P}.\mathbf{f}^{o_3^{\text{industry}}}, \mathrm{P}.\mathbf{f}^{l_3^{09}}, \mathrm{P}.\mathbf{f}^{l_4^{06}}, \mathrm{P}.o_4^{\text{region}}, \mathrm{P}.\mathbf{f}^{o_4^{\text{region}}}, \cdots]$.

4 METHOD

In this section, we show the detailed design of CompNode. As shown in Figure 6, CompNode includes three stages. In Stage 1, based on selected complex-paths, we aggregate the same type of complex-paths by their starting nodes to reduce the computational complexity. Then, in Stage 2, we train a node classification model using the aggregated complex-path features from Stage 1 and labeled target nodes. Finally, in Stage 3, we use the aggregated complex-path features from Stage 1 and the model trained in Stage 2 to rank all the target nodes based on their predicted positive probability.

4.1 Complex-path Pre-aggregation

Motivation. In recent years, most HGNNs [29, 39, 52, 65] have adopted a two-stage approach to aggregate path information: intra-path level aggregation, which aggregates node information within the same path type, and inter-path level aggregation, which aggregates information from all paths. The primary reason for the high memory and time consumption of HGNNs is the intra-path level aggregation. This is because each starting node can correspond to a large number of paths of the same type, and the number of paths grows exponentially with increasing path length [81]. However, recent studies [19, 24, 70] have shown that sometimes averaging the nodes features of the same type within a path does not significantly

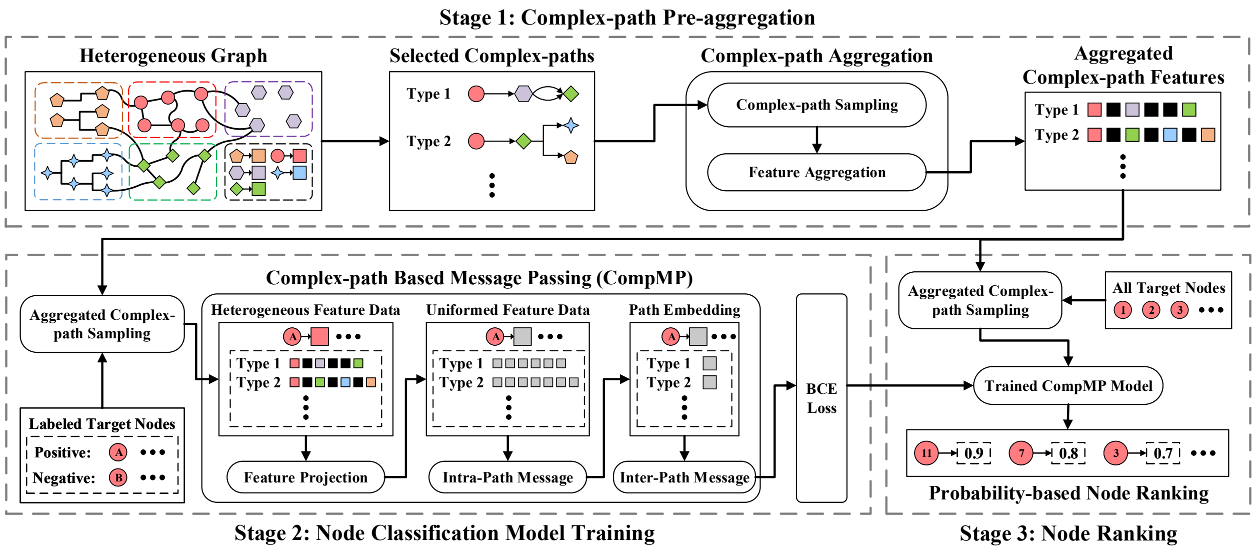


Figure 6: Three stages in the workflow of our CompNode framework.

impact performance. This approach saves memory and sampling time because the number of paths of the same type no longer affects the required computational resources.

Inspired by this idea, we designed a pre-aggregate stage that takes place before model training and prediction. In this stage, the information of nodes and edges of the same type within each complex-path is aggregated based on the path's starting node, using pre-defined aggregation functions. However, even with this approach, the aggregation process still consumes a significant amount of time and memory. A common solution to this challenge is the use of distributed systems [49], which leverage the resources of multiple computers to overcome memory and computation time constraints. In this section, we introduce a method called 'Complex-path Aggregation,' which efficiently completes pre-aggregation of complex-paths using only operations between tables that are compatible with distributed systems designed for relational databases. Next, we will introduce two key modules of this method.

Complex-path Sampling. To obtain all instances corresponding to each complex-path $p \in [p_1, p_2, \cdots, p_{N_p}]$, where N_p is the number of selected complex-paths, we propose 'Complex-path Sampling'. This method, based on our graph structure, is compatible with distributed systems. Algorithm 1 presents the pseudo-code of Complex-path Sampling, with inputs and outputs described below:

$$\mathrm{P} = \mathrm{CompSampling}(HG, p), \tag{1}$$

where HG is our heterogeneous graph and P is the path table corresponding to each complex-path p, as described in Section 3.3.

Our overall approach in Algorithm 1 is as follows: We first sequentially retrieve the edge table E_{k_p} corresponding to each L_{k_p} (Line 2-12), then join the corresponding node tables V_o with each edge table E_{k_p} (Line 13-16). Next, we join all the edge tables E_{k_p} to generate the path table P (Line 17). Finally, we apply conditional constraints, denoted as $p.C$, on the node and edge attributes along the path (Line 19).

Algorithm 1 Complex-Path Sampling (CompSampling).

```
Require: Heterogeneous Graph $HG = (\mathbf{E}, \mathbf{V})$; complex-path $p$.
Ensure: Path table P.
 1: function CompSampling($HG, p$)
 2:     for $k_p = 1$ to Length($p$) do
 3:         $o_i \leftarrow p[k_p, 0], L_{k_p} \leftarrow p[k_p, 1], o_{k_p+1} \leftarrow p[k_p, 2]$    ▷ $i \le k_p$
 4:         for $k_L = 1$ to Length($L_{k_p}$) do
 5:             $e_k \leftarrow L_{k_p}[k_L]$
 6:             if $e_k = l_k$ then
 7:                 $\mathrm{L} \leftarrow \mathrm{E}^{l_k}$
 8:             else if $e_k = p_k$ then
 9:                 $\mathrm{L} \leftarrow$ CompSampling($HG, p_k$)
10:             end if
11:             $\mathrm{E}_{k_p} \leftarrow \mathrm{L}$ if $k_L = 1$ else $\mathrm{E}_{k_p} \underset{\mathrm{E}_{k_p}.o_i=\mathrm{L}.o_i \cap \mathrm{E}_{k_p}.o_{k_p+1}=\mathrm{L}.o_{k_p+1}}{⟗} \mathrm{L}$
                                                          ▷ ⟗ means outer join
12:         end for
13:         if $k_p = 1$ then
14:             $\mathrm{E}_{k_p} \leftarrow \mathrm{E}_{k_p} \underset{\mathrm{E}_{k_p}.o_i=\mathrm{V}^{o_i}.o_i}{\bowtie} \mathrm{V}^{o_i}$    ▷ ⋈ means inner join
15:         end if
16:         $\mathrm{E}_{k_p} \leftarrow \mathrm{E}_{k_p} \underset{\mathrm{E}_{k_p}.o_{k_p+1}=\mathrm{V}^{o_{k_p+1}}.o_{k_p+1}}{\bowtie} \mathrm{V}^{o_{k_p+1}}$
17:         $\mathrm{P} \leftarrow \mathrm{E}_{k_p}$ if $k_p = 1$ else $\mathrm{P} \underset{\mathrm{P}.o_i=\mathrm{E}_{k_p}.o_i}{\bowtie} \mathrm{E}_{k_p}$
18:     end for
19:     $\mathrm{P} \leftarrow$Limit($\mathrm{P}, p.C$)
20:     return P
21: end function
```

Feature Aggregation. Using Equation 1, we can obtain the path table P corresponding to each type of complex-path p as described in Section 3.3. We then aggregate all node feature columns $\mathrm{P}.\mathbf{f}^{o_i}$

and edge feature columns $\mathrm{P}.\mathbf{f}^{l_i}$ in each path table using the path start node column $\mathrm{P}.o_1$ as the key. $\mathrm{P}.o_1$ is the same as $\mathrm{P}.o^{\mathrm{tgt}}$, as we described in Section 3.3. The aggregation function is shown below:

$$\mathbf{V}^p = \mathrm{Aggregate}(\mathrm{P}, f_{\mathrm{agg}}^p), \tag{2}$$

where f_{agg}^p is the aggregation function we choose for each complex-path p ($f_{\mathrm{agg}}^p \in$ {'AVG', 'SUM', 'MAX', 'MIN', 'STD'}). $\mathbf{V}^p = [\mathrm{V}_{o_1}^p, \mathrm{V}_{l_1}^p, \cdots, \mathrm{V}_{o_m}^p]$ is a list of aggregated feature tables for each complex-path p. The length of this list, N_L^p, is the total number of edges and nodes in the path. Each aggregated feature table $\mathrm{V}_{o_i}^p$ or $\mathrm{V}_{l_i}^p$ has an o^{tgt} node ID column and aggregated feature columns $\mathrm{V}_{o_i}^p.\mathbf{f}^{o_i}$ or $\mathrm{V}_{l_i}^p.\mathbf{f}^{l_i}$, which originate from $\mathrm{P}.\mathbf{f}^{o_i}$ or $\mathrm{P}.\mathbf{f}^{l_i}$, respectively.

Based on Equation 1 and 2, the input and output of the entire Complex-path Aggregation method can be represented as follows:

$$\mathbf{V}^p = \mathrm{CompAgg}(HG, p, f_{\mathrm{agg}}^p). \tag{3}$$

By aggregating the instances corresponding to each complex-path $p \in [p_1, p_2, \cdots, p_{N_p}]$ using Equation 3, we obtain the aggregated feature tables $\mathcal{V} = [\mathbf{V}^{p_1}, \cdots, \mathbf{V}^{p_{N_p}}]$ for all complex-paths.

4.2 Node Classification Model Training

Motivation. In this stage, we aim to train a node classification model (Complex-path Based Message Passing) based on labeled nodes, enabling us to rank target nodes according to their predicted probabilities of being positive. Through Equation 3, we have aggregated the instances of each complex-path type p to their corresponding start node o^{tgt}. This process results in each target node having only one path instance for each complex-path type, significantly reducing the required system memory and sampling time for the model based on it. Therefore, we first design a corresponding sampling method (Aggregated Complex-path Sampling) for the aggregated complex-path features. Then, to more accurately complete node classification, we follow the design of many HGNNs [19, 32, 65] while considering our own situation. We first map the various features to the same dimension, and then use the attention mechanism to consider the mutual influence of nodes and edges on the same path and the interactions between different paths. Finally, we synthesize these results to make the ultimate prediction. Next, we will introduce the key components of our model.

Aggregated Complex-path Sampling. We first designed a method to sample the aggregated complex-path features of target nodes on each complex-path. The input and output of this method can be represented as:

$$\mathrm{F}^{\mathrm{sample}}, \mathcal{F}^{\mathrm{sample}} = \mathrm{AggCompSample}(\mathcal{V}, \mathrm{V}^{\mathrm{sample}}), \tag{4}$$

where $\mathrm{F}^{\mathrm{sample}}$ denotes the original features of the sampled node $\mathrm{V}^{\mathrm{sample}}$, and $\mathcal{F}^{\mathrm{sample}} = [\mathbf{F}^{p_1}, \cdots, \mathbf{F}^{p_{N_p}}]$ represents the aggregated features of $\mathrm{V}^{\mathrm{sample}}$ corresponding to each complex-path. For each $\mathbf{F}^p \in \mathcal{F}^{\mathrm{sample}}$, $\mathbf{F}^p = [\mathrm{F}_{o_1}^p, \mathrm{F}_{l_1}^p, \cdots, \mathrm{F}_{o_m}^p]$, $\mathrm{F}_{o_i}^p$ and $\mathrm{F}_{l_i}^p$ represent the aggregated features from the node type o_i and edge type l_i of the complex-path p.

For $\mathrm{F}^{\mathrm{sample}}$, each type of $\mathrm{F}_{o_i}^p$ and $\mathrm{F}_{l_i}^p$, the corresponding features are obtained as follows:

$$\begin{aligned}
\mathrm{F}^{\mathrm{sample}} &= \mathrm{V}^{\mathrm{sample}} \underset{\mathrm{V}^{\mathrm{sample}}.o^{\mathrm{tgt}}=\mathrm{V}^{o^{\mathrm{tgt}}}.o^{\mathrm{tgt}}}{⟕} \mathrm{V}^{o^{\mathrm{tgt}}}.\mathbf{f}^{o^{\mathrm{tgt}}} \\
\mathrm{F}_{o_i}^p &= \mathrm{V}^{\mathrm{sample}} \underset{\mathrm{V}^{\mathrm{sample}}.o^{\mathrm{tgt}}=\mathrm{V}_{o_i}^p.o^{\mathrm{tgt}}}{⟕} \mathrm{V}_{o_i}^p.\mathbf{f}^{o_i} \\
\mathrm{F}_{l_i}^p &= \mathrm{V}^{\mathrm{sample}} \underset{\mathrm{V}^{\mathrm{sample}}.o^{\mathrm{tgt}}=\mathrm{V}_{l_i}^p.o^{\mathrm{tgt}}}{⟕} \mathrm{V}_{l_i}^p.\mathbf{f}^{l_i},
\end{aligned} \tag{5}$$

where ⟕ means left join. Through left join, we can obtain the corresponding features for each target node and the result preserves the order of the nodes in $\mathrm{V}^{\mathrm{sample}}.o^{\mathrm{tgt}}$. The original features of the target node o^{tgt} is represented as $\mathrm{V}^{o^{\mathrm{tgt}}}.\mathbf{f}^{o^{\mathrm{tgt}}}$ as introduced in Section 3, and $\mathrm{V}_{o_i}^p.\mathbf{f}^{o_i}$ and $\mathrm{V}_{l_i}^p.\mathbf{f}^{l_i}$ denote the aggregated features of node type o_i and edge type l_i on each complex-path p obtained through the first stage.

Based on Equation 4 and labeled node table $\mathrm{V}^{\mathrm{label}}$, we can obtain the corresponding original features $\mathrm{F}^{\mathrm{label}}$ and all the aggregated complex-path features $\mathcal{F}^{\mathrm{label}}$.

$$\mathrm{F}^{\mathrm{label}}, \mathcal{F}^{\mathrm{label}} = \mathrm{AggCompSample}(\mathcal{V}, \mathrm{V}^{\mathrm{label}}) \tag{6}$$

Simultaneously, we extract the label column from the labeled node table to serve as the labels for the node in each row.

$$\mathrm{Y}_{\mathrm{label}} = \mathrm{V}^{\mathrm{label}}.label \tag{7}$$

where $\mathrm{Y}_{\mathrm{label}} \in \mathbb{R}^{N_{\mathrm{label}} \times 1}$, N_{label} is the number of labeled nodes. $\mathrm{F}^{\mathrm{label}}$ and any $\mathrm{F}_{o_i}^p, \mathrm{F}_{l_i}^p \in \mathcal{F}^{\mathrm{label}}$ also contain N_{label} rows of data.

Feature Projection. Since different node types and edge types have different features, as shown in Table 1 and 2, the features in $\mathrm{F}^{\mathrm{label}}$, $\mathrm{F}_{o_i}^p$, $\mathrm{F}_{l_i}^p$, which come from different types of nodes and edges, respectively, also have different dimensions. We first use several MLP modules to transform $\mathrm{F}^{\mathrm{label}}$, $\mathrm{F}_{o_i}^p$, $\mathrm{F}_{l_i}^p$ to the same dimension:

$$\mathrm{H}^{\mathrm{label}} = \mathrm{MLP}^{o^{\mathrm{tgt}}}(\mathrm{F}^{\mathrm{label}}), \mathrm{H}_{o_i}^p = \mathrm{MLP}^{o_i}(\mathrm{F}_{o_i}^p), \mathrm{H}_{l_i}^p = \mathrm{MLP}^{l_i}(\mathrm{F}_{l_i}^p), \tag{8}$$

where $\mathrm{H}^{\mathrm{label}}$, $\mathrm{H}_{o_i}^p$, and $\mathrm{H}_{l_i}^p \in \mathbb{R}^{N_{\mathrm{label}} \times d_h}$ represent the transformed features (embedding) with standardized embedding length d_h.

Then, we stack the embedding from the same complex-path p and add a position embedding to indicate their relative positions.

$$\mathbf{H}^p = \mathrm{Stack}(\mathrm{H}_{o_1}^p, \mathrm{H}_{l_1}^p, \cdots, \mathrm{H}_{o_m}^p) + \mathrm{Expand}(\mathrm{h}_{\mathrm{pos}}^p, N_{\mathrm{label}}) \tag{9}$$

where $\mathbf{H}^p \in \mathbb{R}^{N_{\mathrm{label}} \times N_L^p \times d_h}$, $\mathrm{h}_{\mathrm{pos}}^p \in \mathbb{R}^{N_L^p \times d_h}$, $\mathrm{h}_{\mathrm{pos}}^p$ is randomly initialized for each complex-path p, and will be trained along with other parameters. Expand($\cdot$) replicates $\mathrm{h}_{\mathrm{pos}}^p$ for N_{label} rows, resulting in an embedding of dimension $\mathbb{R}^{N_{\mathrm{label}} \times N_L^p \times d_h}$.

At this point, we have the transformed original features (embedding) $\mathrm{H}^{\mathrm{label}}$ of the labeled nodes and the stacked aggregated complex-path features (embedding) $\mathcal{H}^{\mathrm{label}} = [\mathbf{H}^{p_1}, \cdots, \mathbf{H}^{p_{N_p}}]$.

Intra-Path Message Passing. For the embedding of each complex-path $\mathbf{H}^p$, we first use the QKV attention mechanism [58] to understand the mutual influence among nodes and edges on the path. The detailed formula is:

$$\begin{aligned}
\mathbf{Q} = \mathbf{W}_Q^p \mathbf{H}^p, \mathbf{K} = \mathbf{W}_K^p \mathbf{H}^p, \mathbf{V} = \mathbf{W}_V^p \mathbf{H}^p \\
\mathbf{H}_{\mathrm{QKV}}^p = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} + \mathbf{H}^p.
\end{aligned} \tag{10}$$

where $\mathbf{W}_Q^p, \mathbf{W}_K^p, \mathbf{W}_V^p \in \mathbb{R}^{d_h \times d_h}$ are trainable parameters for each complex-path, $\mathbf{H}_{\text{QKV}}^p \in \mathbb{R}^{N_{\text{label}} \times N_L^p \times d_h}$ is the updated embedding for the complex-path p.

Then, we flatten the updated embedding $\mathbf{H}_{\text{QKV}}^p$ into a two dimensional embedding and use an MLP to transform the embedding of different complex-paths into the same dimension.

$$\mathrm{H}_p^o = \mathrm{MLP}^p(\mathrm{Flatten}(\mathbf{H}_{\text{QKV}}^p)) \tag{11}$$

where $\mathrm{H}_p^o \in \mathbb{R}^{N_{\text{label}} \times d_h}$ represents the embedding for each complex-path. Flatten$(\cdot)$ will transfer the dimension of $\mathbf{H}_{\text{QKV}}^o$ to $\mathbb{R}^{N_{\text{label}} \times (N_L^p \times d_h)}$.

Inter-Path Message Passing. To integrate the embedding $\mathrm{H}^{\text{label}}$ and the embedding H_p^o corresponding to each complex-path, we first stack them together.

$$\mathbf{H}^o = \mathrm{Stack}(\mathrm{H}^{\text{label}}, \mathrm{H}_{p_1}^o, \cdots, \mathrm{H}_{p_k}^o, \cdots, \mathrm{H}_{p_{N_p}}^o) \tag{12}$$

where $\mathbf{H}^o \in \mathbb{R}^{N_{\text{label}} \times (N_p+1) \times d_h}$

Then, we employ another QKV attention mechanism to learn the interactions between target nodes and different complex-paths.

$$\begin{gathered}\mathbf{Q} = \mathbf{W}_Q^o\mathbf{H}^o, \mathbf{K} = \mathbf{W}_K^o\mathbf{H}^o, \mathbf{V} = \mathbf{W}_V^o\mathbf{H}^o, \\ \mathbf{H}_{\text{QKV}}^o = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} + \mathbf{H}^o\end{gathered} \tag{13}$$

where $\mathbf{W}_Q^o, \mathbf{W}_K^o, \mathbf{W}_V^o \in \mathbb{R}^{d_h \times d_h}$ are trainable parameters, $\mathbf{H}_{\text{QKV}}^o \in \mathbb{R}^{N_{\text{label}} \times (N_p+1) \times d_h}$ is the updated embedding.

Then, we flatten the updated embedding $\mathbf{H}_{\text{QKV}}^o$ into a two dimensional embedding and use an MLP for prediction.

$$\mathrm{Y}_{\text{pred}} = \mathrm{MLP}^o(\mathrm{Flatten}(\mathbf{H}_{\text{QKV}}^o)) \tag{14}$$

where $\mathrm{Y}_{\text{pred}} \in \mathbb{R}^{N_{\text{label}} \times 1}$ represents the predicted probability for each labeled node. Each element in Y_{pred} is between 0 and 1. Flatten$(\cdot)$ will transfer the dimension of $\mathbf{H}_{\text{QKV}}^o$ to $\mathbb{R}^{N_{\text{label}} \times ((N_p+1) \times d_h)}$.

The input and output of the entire Complex-path Based Message Passing method from Equation 8 to 14 can be represented as:

$$\mathrm{Y}_{\text{pred}} = \mathrm{CompMP}(\mathrm{F}^{\text{label}}, \mathcal{F}^{\text{label}}), \tag{15}$$

where $\mathrm{Y}_{\text{pred}} \in \mathbb{R}^{N_{\text{label}} \times 1}$.

Loss. We train the CompMP model with BCE loss.

$$\mathcal{L} = -[\mathrm{Y}_{\text{label}}^T \ln \mathrm{Y}_{\text{pred}} + (1 - \mathrm{Y}_{\text{label}})^T \ln(1 - \mathrm{Y}_{\text{pred}})] \tag{16}$$

4.3 Node Ranking

Through the first stage, we have already obtained the aggregated features $\mathcal{V}$ of all the complex-paths. Then, by applying Equation 4, we can acquire the original features F^{tgt} and all the complex-path aggregated features $\mathcal{F}^{\text{tgt}}$ corresponding to all target nodes $\mathrm{V}^{o^{\text{tgt}}}$.

$$\mathrm{F}^{\text{tgt}}, \mathcal{F}^{\text{tgt}} = \mathrm{AggCompSample}(\mathcal{V}, \mathrm{V}^{o^{\text{tgt}}}) \tag{17}$$

Through the second stage, we have successfully trained a node classification model CompMP represented by Equation 15. Now, we can directly use it to complete the prediction for all target nodes.

$$\mathrm{Y}^{\text{tgt}} = \mathrm{CompMP}(\mathrm{F}^{\text{tgt}}, \mathcal{F}^{\text{tgt}}) \tag{18}$$

where $\mathrm{Y}^{\text{tgt}} \in \mathbb{R}^{N_{\text{tgt}} \times 1}$, where N_{tgt} is the number of all target nodes.

Based on the prediction results Y^{tgt}, we can rank all target nodes in descending order of their predicted probabilities, such that nodes with higher rankings are more likely to be positive.

5 EXPERIMENT

In this section, we carefully design our experiments by considering the following key research questions:

- **RQ1:** Is our framework more effective than other baselines?
- **RQ2:** Is it necessary to use such a large amount of evolving training data and so many types of complex-paths?
- **RQ3:** Is our framework more efficient than other methods?
- **RQ4:** Does our pre-aggregation stage affect model performance?
- **RQ5:** Are all the design elements of our model effective?
- **RQ6:** Is each type of node in the graph necessary?
- **RQ7:** Are complex-paths also effective for other tasks?

5.1 Experimental Setup

Hardware. For the pre-aggregation stage, we utilized a Spark [49] cluster consisting of 200 executors, each equipped with 4 cores and 16GB of memory. For stages 2 and 3, we used 4 P40 GPUs, each with 24GB of memory, along with 180GB of system memory.

Datasets. Details for the raw dataset used can be found in Section 3. We constructed 12 heterogeneous graphs based on data prior to the first day of each month from 2022.10.01 to 2023.09.01 and obtained over 1.5 million labeled nodes based on feedback from sales between 2022.10.01 and 2023.09.30. The labeled nodes were split chronologically into three datasets according to their associated contact time ($\mathrm{V}_{\text{label}}.date$), with each dataset containing nodes contacted in 4 consecutive months. The corresponding contact time periods for datasets 1-3 are 2022.10.01 to 2023.01.31, 2023.02.01 to 2023.05.31, and 2023.06.01 to 2023.09.30, respectively.

Within each dataset, we further divided the labeled nodes chronologically based on their associated contact time into training, validation, and test sets. The first three months of labeled nodes in each dataset were used for training, while the fourth month was split into validation and test sets, with the first half of the month used for validation and the second half for testing. For example, in dataset 3, labeled nodes from 2023.06.01 to 2023.08.31, were used as the training set; labeled nodes from 2023.09.01 to 2023.09.15, were used as the validation set; and labeled nodes from 2023.09.16, to 2023.09.30, were used as the test set. Furthermore, each sample was only associated with the heterogeneous graph corresponding to the first day of its contact month. For instance, a node contacted on 2023.09.10 would only utilize the graph constructed on 2023.09.01.

Baselines. We compared our approach with six state-of-the-art HGNNs, which are: (i) Relational Graph Convolutional Networks (R-GCNs) [52]; (ii) Heterogeneous Graph Attention Network (HAN) [65]; (iii) Heterogeneous Graph Transformer (HGT) [29]; (iv) Meta-path Aggregated Graph Neural Network (MAGNN) [19]; (v) Simple-HGN [39] and (vi) Simple and Efficient Heterogeneous Graph Neural Network (SeHGNN) [70].

These HGNNs fall into two categories: those based on k-hop sampling (R-GCNs, HGT, Simple-HGN) and those based on meta-path sampling (HAN, MAGNN, SeHGNN). For models based on k-hop sampling, we sample the 2-hop subgraph for each target node and use these sampling results as input. For models based on meta-path sampling, we filtered the 53 complex-paths that were used in CompNode based on the definition of Meta-path, which

Table 3: Accuracy comparison between baselines and our framework on potential high-value customer classification.

Model	Dataset 1					Dataset 2					Dataset 3				
	AUC	AP	P@5k	P@10k	P@20k	AUC	AP	P@5k	P@10k	P@20k	AUC	AP	P@5k	P@10k	P@20k
MLP	0.578	0.0401	4.98%	4.47%	4.20%	0.559	0.0367	4.28%	4.02%	3.80%	0.572	0.0392	4.58%	4.27%	4.12%
MetaAgg-MLP	0.597	0.0412	5.64%	5.28%	4.41%	0.573	0.0377	4.36%	4.12%	4.06%	0.592	0.0416	5.46%	4.62%	4.28%
CompAgg-MLP	0.647	0.0520	8.02%	6.73%	5.63%	0.618	0.0429	6.16%	5.60%	5.44%	0.639	0.0473	7.58%	6.70%	5.65%
R-GCNs	0.604	0.0423	5.98%	5.09%	4.65%	0.570	0.0385	5.26%	4.48%	4.22%	0.594	0.0416	5.84%	4.96%	4.43%
HAN	0.609	0.0429	5.86%	5.24%	4.82%	0.572	0.0378	4.46%	4.11%	3.97%	0.586	0.0407	4.68%	4.42%	4.21%
HGT	0.621	0.0485	6.72%	5.84%	5.31%	0.587	0.0412	5.48%	5.03%	4.91%	0.612	0.0454	6.18%	5.49%	5.13%
Simple-HGN	0.613	0.0448	6.08%	5.32%	4.76%	0.573	0.0398	5.32%	4.71%	4.36%	0.601	0.0429	5.88%	5.12%	4.62%
MAGNN	0.610	0.0431	5.96%	5.21%	4.77%	0.568	0.0393	4.94%	4.49%	4.35%	0.603	0.0418	5.54%	4.91%	4.57%
SeHGNN	0.628	0.0471	6.88%	6.02%	5.29%	0.592	0.0415	5.26%	4.83%	4.72%	0.616	0.0453	5.94%	5.37%	4.98%
CompNode	**0.676**	**0.0597**	**9.94%**	**8.95%**	**7.43%**	**0.647**	**0.0509**	**8.02%**	**7.22%**	**6.36%**	**0.664**	**0.0542**	**9.02%**	**8.15%**	**6.63%**

resulted in 8 meta-paths. We then utilized the sampling results of these 8 meta-paths as input.

In addition, we employed Multi-layer Perceptron (MLP) [50] to validate the effectiveness of our selected complex-paths and the proposed framework CompNode. Specifically, for MLP, we used three types of input: (i) the features of the target node itself (MLP), (ii) the features of the target node concatenated with the aggregated features from the 8 meta-paths used in HGNNs (MetaAgg-MLP), and (iii) the features of the target node concatenated with the aggregated features from our 53 complex-paths (CompAgg-MLP).

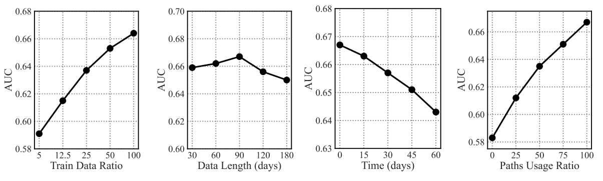
Training Configuration. For all the baselines and our model, we set all hidden layer dimensions to 64, the learning rate to 5e-5. In each batch, 4,000 samples were randomly selected for training. The model that performed the best on the validation set was chosen, and its performance on the test set was used as the final result. For models based on k-hop sampling, two graph neural network layers were employed, while default structures were applied for those using meta-path sampling. In all the HGNNs, we limited samples to 20 per edge type per hop.

Evaluation Metrics. Following existing studies [19, 52], we evaluate the performance of all the models using Area Under the Curve (AUC), Average Precision (AP), Precision @ 5000 (P@5k), Precision @ 10000 (P@10k), Precision @ 20000 (P@20k).

5.2 Overall Performance (RQ1)

To address **RQ1**, we evaluated the performance of our framework (CompNode), against other methods on three datasets using default parameters. The results are shown in Table 3.

From Table 3, we observe that MLP, when utilizing only the target nodes' features, exhibits the worst performance. When incorporating the features aggregated by the 8 meta-paths used by HGNNs, MetaAgg-MLP still performs worse than most HGNNs, which demonstrates the effectiveness of current HGNNs. However, CompAgg-MLP significantly surpasses various HGNNs when it incorporates the features aggregated by our 53 complex-paths. This outcome suggests that current HGNNs lack the capability to autonomously utilize the information from non-linear conditional paths represented by our complex-paths. Moreover, our model outperforms CompAgg-MLP, indicating that integrating design strategies from HGNNs can enhance the mining and synthesis of information across diverse complex-paths. Furthermore, the decreasing values of P@5k, P@10k, and P@20k demonstrate the effectiveness of ranking target nodes based on the prediction probabilities obtained from node classification models.



(a) Data ratio (b) Data length (c) Time interval (d) Usage ratio

Figure 7: Necessity of large-scale data and complex-paths.

5.3 Influence of Training Data (RQ2)

To address **RQ2**, we tested the influence of using different portions of the training data and different portions of complex-path types. The test results are shown in Figure 7. Next, we will discuss the results of each test individually.

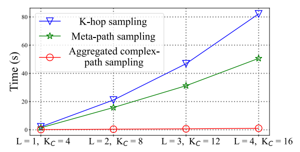
Evaluation on different proportions of training data. Firstly, we tested the impact of using different proportions of training data on the results, which are shown in Figure 7(a). We can observe that as the number of training samples increases, the model's performance improves significantly. Therefore, it is crucial to enable the model to train with large-scale data.

Evaluation on different data lengths. We also explored the impact of using samples from different data lengths. The results are shown in Figure 7(b). We observed that as the time span of the samples increased, the performance first improved, then slowly decreased. This indicates that it is better to utilize data from the past 90 days to train the model.

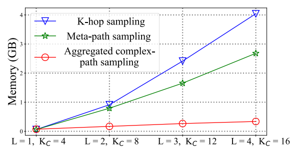
Evaluation on different lengths of interval time. To verify the necessity of continuously updating the data, we tested the model's performance on the test set under different time intervals between the validation and test sets. The results, shown in Figure 7(c), indicated that as the time interval increases, the performance decreases significantly. This phenomenon suggests that the latest data can bring a better performance.

Table 4: Theoretical analysis of our method's efficiency.

	Sampling Time	Memory Consumption
K-hop sampling	$O(N_T \prod_{l=1}^{L} \sum_{r \in R^l} \varepsilon_r)$	$O(N_T \prod_{l=1}^{L} \sum_{r \in R^l} \varepsilon_r D_{r^l})$
Meta-path sampling	$O(N_T \sum_{k=1}^{K_C} \prod_{r \in R^p} \varepsilon_r)$	$O(N_T \sum_{k=1}^{K_C} \prod_{r \in R^p} \varepsilon_r D_{r^p})$
Aggregated complex-path sampling	$O(N_T K_C)$	$O(N_T \sum_{k=1}^{K_C} D_{r^p})$



(a) Sampling time



(b) Memory consumption

Figure 8: Empirical analysis of our method's efficiency.

Evaluation on different complex-paths usage ratio. To investigate the effect of using different ratios of complex-paths, we conducted an experiment where only a portion of complex-paths was retained. The results are shown in Figure 7(d). It can be seen that as the proportion of complex-paths used increases, the performance steadily improves. This indicates that by utilizing various kinds of complex-paths, we can effectively extract latent information from the graph and improve the performance.

5.4 Efficiency Analysis (RQ3)

Based on our experimental results in Section 5.3, we need to use a large amount of data to train the model and continuously re-train it with the newest data. Therefore, our framework must be highly efficient. However, current HGNNs based on k-hop and meta-path sampling require significant time and memory to sample and store neighbor nodes and edges. To address this, our framework employs a pre-aggregation stage to pre-aggregate the features of the same type of neighbor nodes and edges to the target node. As a result, our aggregated complex-path sampling can be highly efficient. Next, we will validate its efficiency both theoretically and experimentally.

Theoretical analysis. Firstly, from a theoretical perspective, assuming the model is processing a sample containing N_T nodes and needs to sample the tail nodes of K_C meta-paths, the number of features for the tail node corresponding to each kind of edge r is denoted as D_r. If a k-hop sampling method is utilized, assuming the maximum number of hops involved across all meta-paths is L, and the relations involved in the l-th hop of these meta-paths are denoted as $\{r \in R^l\}$, each kind of edge r necessitates the addition of ε_r neighbor nodes. Thus, for each hop, we need to sample and store $\sum_{r \in R^l} \varepsilon_r$ additional nodes for each node from the last hop. Therefore, with each additional hop, the count is multiplied accumulatively, and the final result is shown in Table 4. On the other hand, for the meta-path sampling method, assuming the edges contained in each meta-path are denoted as $\{r \in R^p\}$, and ε_r neighbor nodes need to be sampled at each hop. Since each edge on the same meta-path continues sampling based on the results of the previous hop, $\prod_{r \in R^p} \varepsilon_r$ neighbor nodes need to be sampled and stored for each kind of meta-path. By summing up the sampling results of each meta-path, the final result is as shown in Table 4. In contrast, our aggregated complex-path sampling method only necessitates one round of sampling for each kind of meta-path during the sampling stage, and the memory required for each node is merely the sum of the feature lengths corresponding to the tail nodes of all meta-paths.

Empirical analysis. We also validated our results through experiments. We randomly selected 10 groups, each with 20,000 nodes, and chose 16 meta-paths with lengths of 1, 2, 3, and 4 respectively. When we sequentially utilized meta-paths with lengths less than or equal to 1, 2, 3, and 4, the changes in average sampling time and memory consumption are illustrated in Figure 8. It can be observed that, with the increase in the number of hops, both the sampling time and memory required for the two traditional methods increase quickly, whereas our method can maintain stable time and memory consumption. Therefore, k-hop and meta-path sampling based methods take over 30 days to predict for all our target nodes, while our framework only requires 1 day.

Table 5: Evaluating the influence of aggregation methods.

Model	Dataset 1	Dataset 2	Dataset 3
Complex-path + GCN	0.609	0.589	0.591
Complex-path + GAT	**0.618**	0.588	0.599
Complex-path + Aggregation	0.614	**0.596**	**0.605**

Table 6: Ablation study.

Ablated Design	AUC	AP	P@5k	P@10k	P@20k
None	0.662	0.0549	8.99%	8.11%	6.81%
Position Embed	0.654	0.0525	8.36%	7.24%	6.49%
Intra-path Message	0.649	0.0523	8.31%	7.46%	6.37%
Inter-path Message	0.643	0.0508	8.06%	6.91%	5.88%

5.5 Aggregation Influence Analysis (RQ4)

While aggregated complex-path sampling is highly efficient, would our method decrease the model performance? To answer this question, we aggregated the features of nodes of the same type within the same complex-paths using three different methods: Aggregation functions (f_{agg}^{p} in Equation 2), GCN, and GAT. The aggregated features were then fed into an MLP model for prediction. Due to system memory constraints, we could not simultaneously aggregate all complex-paths using GCN and GAT. Therefore, we randomly divided all complex-paths into four groups and used the average results of the four groups as the final performance. The performance of these three approaches on the three datasets is shown in Table 5. We can observe that our method can achieve better results in most cases. We believe this is because the complex-path has already accurately selected neighbor nodes of the same type, reducing the significance of the dynamic allocation of weights to neighbor nodes.

Table 7: Impact of node type removal.

Node	AUC	AP	P@5k	P@10k	P@20k
Region	0.629	0.0477	7.18%	6.17%	5.38%
Customer	0.625	0.0465	7.32%	6.22%	5.46%
Account	0.638	0.0486	8.02%	6.92%	5.83%
Industry	0.644	0.0502	7.88%	6.83%	6.04%
Company	0.642	0.0491	7.42%	6.69%	5.78%

5.6 Ablation Study (RQ5)

To address **RQ5**, we conducted an ablation study to validate the various designs presented in Section 4.2. The average results on 3 datasets are shown in Table 6, where the first row corresponds to the performance without ablating any modules. In Section 4.2, we propose three main designs. First, we add position embeddings to the features of nodes and edges based on their positions in the complex-path, allowing the model to consider the different influences of the same features at different positions. Removing this step, as shown in the second row of the Table 6, slightly decreases the model's performance. Second, in the Intra-path Message Passing module, we model the mutual influence between different nodes and edges in the same complex-path using Equation 10. As shown in the third row of Table 6, ablating this equation and directly transforming the features using Equation 11 also decreases the model's performance. Third, in the Inter-path Message Passing module, we model the mutual influence between different complex-paths using Equation 13. As shown in the fourth row of Table 6, ablating this equation and directly making the prediction using Equation 14 significantly decreases the model's performance.

5.7 Significance of Diverse Source Data (RQ6)

To address **RQ6**, we remove each type of node from the graph sequentially to observe their impact on model performance. The results are shown in Table 7. From the results, it is evident that the model's final performance decreases significantly when any type of information is missing. Also, we employed Shap-Value [38] to analyze the importance of features [16, 73] from various sources. We extracted the top 100 features based on their importance and identified their sources: Customer (12%), Region (66%), Account (9%), Company (7%), and Industry (6%). These findings indicate that features from all sources benefit the final prediction performance. Notably, features related to the Region are considerably more important, suggesting a significant correlation between a customer's potential value and the region in which the customer lives.

5.8 Evaluation on External Datasets (RQ7)

To verify the effectiveness of complex-paths for other node prediction tasks, we conducted experiments on paper category prediction using the Microsoft Academic Graph [60]. This is also a billion-scale heterogeneous graph, which contains various node types such as papers and authors, along with edge types representing citation and authorship relations. Given the strong correlation between neighboring and target papers' categories, we used our Complex-path Aggregation (Equation 3) to count categories of neighboring papers connected through various paths. If we use the most frequently counted category as the prediction result, counting results based on meta-paths like "paper $\xrightarrow{citation}$ paper" or "paper $\rightarrow$ author $\rightarrow$ paper" achieved prediction accuracy of 69.83% and 70.58%, respectively. Our complex-path definition, however, allowed for better selection of relevant neighboring papers through nonlinear paths. For example, by considering papers that share any author and are cited by the target paper, the prediction accuracy rose to 72.86%.

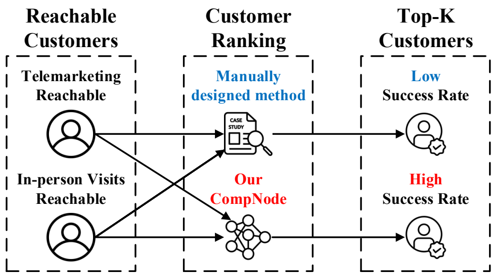


Figure 9: Real-world usage of our framework.

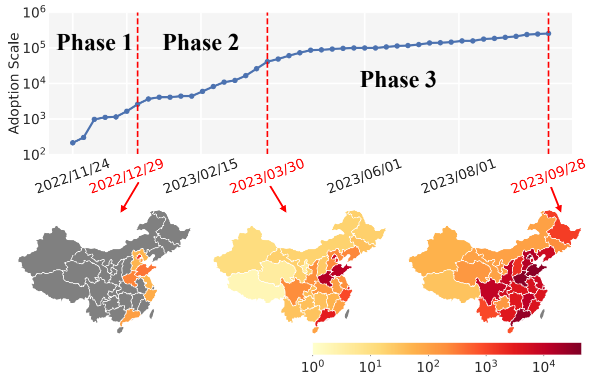


Figure 10: Three-phase real-world deployment results.

6 REAL-WORLD DEPLOYMENT

To further evaluate the impact of our framework, we deploy it at a major logistics company. Our framework serves as the key component to rank customers based on their potential to place more orders with this company. This company contacts customers through two channels: *telemarketing* and *in-person visits*. During these interactions, customers are asked about their intention to place orders, and those who express interest will be offered discounts if their future order volume exceeds a certain threshold. As shown in Figure 9, our framework outperforms the previous approach in identifying customers who are interested in placing more orders with a higher success rate, which can help this company expand its business. In the following sections, we will discuss the implementation results of our framework in both telemarketing and in-person visit scenarios.

Table 8: Real world A/B test result.

Model	Total Calls	Interested Calls	Success Rate
Random sampling	714	7	0.98%
Previous method	2,067	50	2.42%
Ours	2,391	204	**8.53%**

6.1 Upgrading Customers via Telemarketing

Background. Telemarketing means contacting customers via phone calls by telemarketers and is widely adopted in industries due to its broad customer reach and low cost. Therefore, we deploy our framework to rank customers according to their potential to become high-value customers, prioritizing calls to customers with a higher potential of being high-value, which can help telemarketers contact customers more effectively

3-phase deployment. From December 2022 to September 2023, we continuously made calls and scaled up each week with the help of telemarketers, making over 200,000 calls and achieving a 6.1% success rate. The overall situation is depicted in Figure 10. **In Phase 1**, we first deployed our model in a few cities and achieved a more than 10% success rate. Motivated by the promising results, we further expanded the deployment to more than 360 cities across China in **Phase 2**. Before large-scale implementation nationwide in **Phase 3**, we compare our model with 1) random sampling and 2) the previous method, which is the best method implemented by graph database used by the company's business department. As shown in Table 8, our model significantly outperforms both methods in identifying potential high-value customers. Given the superior performance, we implemented the model nationwide for identifying top potential high-value customer in Phase 3.

6.2 Upgrading Customers via In-Person Visits

Background. While telemarketing is an effective approach for upgrading customers, some high-value customers may be difficult to reach through telemarketing. Therefore, in-person visits [26–28] to potential customers by professional staff serve as another important approach for customer acquisition.

Deployment. We first conducted site visits to over 1,500 companies across three distinct delivery regions in Beijing City, identifying 243 companies that expressed strong interest in placing more orders with a reasonable discount. We then checked the ranking results of these interested companies among all visited companies and found that 233 of them ranked in the top 500. Building on this result, we expanded the application of our framework to 10 delivery regions. Collectively, our methodology achieved a success rate of 40% among all visited companies, significantly benefiting the platform.

7 RELATED WORK

7.1 Graph Database

Graph databases have been gaining popularity in recent years [51]. Modern graph query languages, such as Neo4j's Cypher [12], Tigergraph's GSQL [22], and Oracle's PGQL [47], are seeing rapid adoption in industry. Path queries, which form the core of these languages, have been studied in database research since the late 1980s [5, 6, 8, 11, 42, 43]. As data grows in size and complexity, new research topics have emerged. For example, Martens et al. introduced path multiset representations, which can represent path multisets with exponential succinctness [41]. Gou et al. proposed a novel algorithm for persistent regular path queries on streaming graphs [21]. Zhang et al. proposed a new extension-based approach for shortest path queries to reduce the required space cost while still guaranteeing query time [76]. However, for node ranking problems, approaches relying solely on graph databases require manually designed algorithms to aggregate results from various paths [23, 40]. When the importance value of a node cannot be explicitly expressed, such methods struggle to achieve high accuracy.

7.2 Heterogeneous Graph Neural Network

Recent advancements in HGNNs [56, 67, 68] have provided a new direction for solving problems that are difficult to address with manually defined rules. HGNNs [54, 66] can automatically synthesize information on a heterogeneous graph based on labeled data and neighborhood information to solve many downstream tasks, such as node classification [25, 59, 71], link prediction [75, 77, 78] , and community search [30, 34, 36]. As neural networks require input in a fixed format, a unified sampling of the graph structure information is needed to design models based on this. Currently, there are two main sampling methods: k-hop sampling and meta-path sampling. K-hop sampling approaches [29, 52] collect all neighborhood nodes within k hops from the target node without explicitly filtering node and edge types. For instance, Schlichtkrull et al. proposed Relational Graph Convolutional Networks (RGCN) for modeling heterogeneous graphs [52]. RGCN maintains a unique linear projection weight for each edge type. However, these methods struggle to leverage long-distance information due to the over-smoothing issue. To address this, meta-path sampling approaches were introduced, defining a sequence of edges based on prior knowledge and enabling the model to utilize a broader range of information [65, 70]. For example, Wang et al. [65] used graph attention networks to learn the semantic information from different meta-paths.

8 CONCLUSION

In this paper, we proposed a framework for discovering potential high-value customers in billion-scale heterogeneous graphs. Specifically, we first introduced the concept of Complex-path to describe the complex information in heterogeneous graphs. Based on Complex-path, we design our framework CompNode, which models large-scale data effectively and efficiently. Results on real-world datasets show that our framework achieves superior performance compared to multiple state-of-the-art methods. Moreover, our framework is deployed at a major logistics company for nationwide high-value customer discovery.

ACKNOWLEDGMENTS

We thank all the reviewers for their insightful feedback to improve this paper. This work is partially supported by the National Science and Technology Major Project under Grant 2021ZD0114200, the National Natural Science Foundation of China (NSFC) under Grant 61925202, and the Jiangsu Provincial Key Research and Development Program under Grant BE2022065-1, BE2022065-3.

REFERENCES

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter A. Boncz, George H. L. Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan F. Sequeda, Oskar van Rest, and Hannes Voigt. 2018. G-CORE: A Core for Future Graph Query Languages. In *International Conference on Management of Data (SIGMOD)*. 1421–1432.

[2] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savkovic, Michael Schmidt, Juan F. Sequeda, Slawek Staworko, and Dominik Tomaszuk. 2021. PG-Keys: Keys for Property Graphs. In *International Conference on Management of Data (SIGMOD)*. ACM, 2423–2436.

[3] Marcelo Arenas, Sebastián Conca, and Jorge Pérez. 2012. Counting Beyond a Yottabyte, or How SPARQL 1.1 Property Paths Will Prevent Adoption of the Standard. In *International Conference on World Wide Web (WWW)*. 629–638.

[4] Guillaume Bagan, Angela Bonifati, and Benoît Groz. 2013. A Trichotomy for Regular Simple Path Queries on Graphs. In *Symposium on Principles of Database Systems (PODS)*. 261–272.

[5] Guillaume Bagan, Angela Bonifati, and Benoît Groz. 2013. A Trichotomy for Regular Simple Path Queries on Graphs. In *Symposium on Principles of Database Systems (PODS)*. 261–272.

[6] Pablo Barceló. 2013. Querying graph databases. In *Symposium on Principles of Database Systems (PODS)*. 175–188.

[7] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Lake Tahoe, Nevada) *(NIPS'13)*. Curran Associates Inc., Red Hook, NY, USA, 2787–2795.

[8] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 1999. Rewriting of Regular Expressions and Regular Path Queries. In *ACM Symposium on Principles of Database Systems*. ACM Press, 194–204.

[9] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. 2023. Heterogeneous Graph Contrastive Learning for Recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining* (, Singapore, Singapore,) *(WSDM '23)*. Association for Computing Machinery, New York, NY, USA, 544–552. https://doi.org/10.1145/3539597.3570484

[10] Zixuan Chen, Panagiotis Manolios, and Mirek Riedewald. 2023. Why Not Yet: Fixing a Top-k Ranking that Is Not Fair to Individuals. *Proceedings of the VLDB Endowment* 16, 9 (2023), 2377–2390.

[11] Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. 1987. A Graphical Query Language Supporting Recursion. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 323–330.

[12] cypher. [n.d.]. Cypher Query Language. https://neo4j.com/developer/cypher/.

[13] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD '17*. ACM, 135–144.

[14] Zheng Dong, Xin Huang, Guorui Yuan, Hengshu Zhu, and Hui Xiong. 2021. Butterfly-Core Community Search over Labeled Graphs. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2006–2018.

[15] Chenguang Du, Kaichun Yao, Hengshu Zhu, Deqing Wang, Fuzhen Zhuang, and Hui Xiong. 2023. Seq-HGNN: Learning Sequential Node Representation on Heterogeneous Graph. In *SIGIR*.

[16] Alexandre Duval and Fragkiskos D. Malliaros. 2021. GraphSVX: Shapley Value Explanations for Graph Neural Networks. In *ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part II (Lecture Notes in Computer Science)*, Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and José Antonio Lozano (Eds.), Vol. 12976. 302–318.

[17] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR*.

[18] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *Proceedings of the VLDB Endowment* 12, 5 (2019), 461–474.

[19] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW* (Taipei, Taiwan). 2331–2341.

[20] Di Ge, Zheng Dong, Yuhang Cheng, and Yanwen Wu. 2024. An enhanced spatio-temporal constraints network for anomaly detection in multivariate time series. *Knowledge-Based Systems* 283 (2024), 111169.

[21] Xiangyang Gou, Xinyi Ye, Lei Zou, and Jeffrey Xu Yu. 2024. LM-SRPQ: Efficiently Answering Regular Path Query in Streaming Graphs. *Proceedings of the VLDB Endowment* 17, 5 (2024), 1047–1059.

[22] GSQL. [n.d.]. GSQL. https://www.tigergraph.com/gsql/.

[23] A. Gulino, S. Ceri, G. Gottlob, E. Sallinger, and L. Bellomarini. 2021. Distributed company control in company shareholding graphs. In *ICDE*. IEEE.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NeurIPS* 30 (2017).

[25] Jinquan Hang, Zheng Dong, Hongke Zhao, Xin Song, Peng Wang, and Hengshu Zhu. 2022. Outside In: Market-aware Heterogeneous Graph Neural Network for Employee Turnover Prediction. In *WSDM '22*. ACM, Virtual Event, 353–362.

[26] Zhiqing Hong, Guang Wang, Wenjun Lyu, Baoshen Guo, Yi Ding, Haotian Wang, Shuai Wang, Yunhuai Liu, and Desheng Zhang. 2022. CoMiner: nationwide behavior-driven unsupervised spatial coordinate mining from uncertain delivery events. In *SIGSPATIAL* (Seattle, Washington). Article 10, 10 pages.

[27] Zhiqing Hong, Haotian Wang, Yi Ding, Guang Wang, Tian He, and Desheng Zhang. 2024. SmallMap: Low-cost Community Road Map Sensing with Uncertain Delivery Behavior. *UBICOMP* 8, 2, Article 50 (may 2024), 26 pages.

[28] Zhiqing Hong, Heng Yang, Haotian Wang, Wenjun Lyu, Yu Yang, Guang Wang, Yunhuai Liu, Yang Wang, and Desheng Zhang. 2022. FastAddr: real-time abnormal address detection via contrastive augmentation for location-based services. In *SIGSPATIAL* (Seattle, Washington). Article 64, 10 pages.

[29] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW* (Taipei, Taiwan). 2704–2710.

[30] Xin Huang, Laks V. S. Lakshmanan, and Jianliang Xu. 2017. Community Search over Big Graphs: Models, Algorithms, and Opportunities. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. IEEE Computer Society, 1451–1454.

[31] JDGalileo. 2021. Galileo Repository. https://github.com/JDGalileo/galileo

[32] Houye Ji, Xiao Wang, Chuan Shi, Bai Wang, and Philip S. Yu. 2023. Heterogeneous Graph Propagation Network. *TKDE* 35, 1 (2023), 521–532.

[33] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[34] Ling Li, Siqiang Luo, Yuhai Zhao, Caihua Shan, Zhengkui Wang, and Lu Qin. 2023. COCLEP: Contrastive Learning-based Semi-Supervised Community Search. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 2483–2495.

[35] Xiang Li, Danhao Ding, Ben Kao, Yizhou Sun, and Nikos Mamoulis. 2021. Leveraging Meta-path Contexts for Classification in Heterogeneous Information Networks. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 912–923.

[36] Boge Liu, Fan Zhang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2021. Efficient Community Search with Size Constraint. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 97–108.

[37] Qi Liu, Zheng Dong, Chuanren Liu, Xing Xie, Enhong Chen, and Hui Xiong. 2014. Social Marketing Meets Targeted Customers: A Typical User Selection and Coverage Perspective. In *ICDM*. 350–359.

[38] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *NeurIPS* (Long Beach, California, USA). 4768–4777.

[39] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are We Really Making Much Progress? Revisiting, Benchmarking and Refining Heterogeneous Graph Neural Networks. In *KDD* (Virtual Event, Singapore). 1150–1160.

[40] Davide Magnanimi, Luigi Bellomarini, Stefano Ceri, and Davide Martinenghi. 2023. Reactive Company Control in Company Knowledge Graphs. In *ICDE*. 3336–3348.

[41] Wim Martens, Matthias Niewerth, Tina Popp, Carlos Rojas, Stijn Vansummeren, and Domagoj Vrgoč. 2023. Representing Paths in Graph Database Pattern Matching. 16, 7 (mar 2023), 1790–1803.

[42] Wim Martens, Matthias Niewerth, and Tina Trautner. 2020. A Trichotomy for Regular Trail Queries. In *STACS (LIPIcs)*, Vol. 154. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 7:1–7:16.

[43] Alberto O. Mendelzon and Peter T. Wood. 1995. Finding Regular Simple Paths in Graph Databases. *SIAM J. Comput.* 24, 6 (12 1995), 1235–1258.

[44] N.G. Nwokah and J. Gladson-Nwokah. 2015. Impact of Social Network on Customer Acquisition in the Banking Industry in Nigeria. *Information and Knowledge Management* 5 (2015), 150–163.

[45] Subhransu Panda and K. Siva Nageswara Rao. 2019. Customer Acquisition and Retention in Non-Banking Finance Companies (NBFC). *Journal of mechanics of continua and mathematical sciences* 5 (2019), 601–613.

[46] Yeonhong Park, Sunhong Min, and Jae W. Lee. 2022. Ginex: SSD-enabled Billion-scale Graph Neural Network Training on a Single Machine via Provably Optimal In-memory Caching. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2626–2639.

[47] PGQL. [n.d.]. PGQL. https://pgql-lang.org/.

[48] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. 2022. xFraud: Explainable Fraud Transaction Detection. *Proceedings of the VLDB Endowment* 15, 3 (2022), 427–436.

[49] J. J. Ratey. 2013. *Spark*. Little, Brown & Company.

[50] Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* 65, 6 (1958), 386–408.

[51] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, et al. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71. https://doi.org/10.1145/3434642

[52] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer, Heraklion, Crete, Greece, 593–607.

[53] Jieming Shi, Renchi Yang, Tianyuan Jin, Xiaokui Xiao, and Yin Yang. 2019. Real-time Top-k Personalized PageRank over Large Graphs on GPUs. *Proceedings of the VLDB Endowment* 13, 1 (2019).

[54] Yizhou Sun and Jiawei Han. 2013. Mining Heterogeneous Information Networks: A Structural Analysis Approach. *SIGKDD Explor. Newsl.* 14, 2 (2013), 20–28.

[55] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Path-Sim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. 4 (2011), 992–1003.

[56] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2022. Heterogeneous Information Networks: the Past, the Present, and the Future. *Proc. VLDB Endow.* 15, 12 (2022), 3807–3811.

[57] Shulong Tan, Weijie Zhao, and Ping Li. 2022. Fast Neural Ranking on Bipartite Graph Indices. *Proceedings of the VLDB Endowment* 15, 4 (2022), 794–803.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need *(NIPS'17)*. 6000–6010.

[59] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[60] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft Academic Graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.

[61] Li Wang, Peipei Li, Kai Xiong, Jiashu Zhao, and Rui Lin. [n.d.]. Modeling Heterogeneous Graph Network on Fraud Detection: A Community-based Framework with Attention Mechanism *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 1959–1968.

[62] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv* (2019).

[63] Ping Wang, Khushbu Agarwal, Colby Ham, Sutanay Choudhury, and Chandan K. Reddy. 2021. Self-Supervised Learning of Contextual Embeddings for Link Prediction in Heterogeneous Networks *(WWW '21)*. 2946–2957. https://doi.org/10.1145/3442381.3450060

[64] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S. Yu. 2020. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *IEEE Transactions on Big Data* 9 (2020), 415–436.

[65] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW* (San Francisco, CA, USA). 2022–2032.

[66] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), 4–24.

[67] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han. 2022. Heterogeneous Network Representation Learning: A Unified Framework With Survey and Benchmark. *TKDE* 34 (2022), 4854–4873.

[68] Guang Yang, Yuequn Zhang, Jinquan Hang, Xinyue Feng, Zejun Xie, Desheng Zhang, and Yu Yang. 2023. CARPG: Cross-City Knowledge Transfer for Traffic Accident Prediction via Attentive Region-Level Parameter Generation. In *CIKM* (Birmingham, United Kingdom). 2939–2948.

[69] Hongxia Yang. 2019. AliGraph: A Comprehensive Graph Neural Network Platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 3165–3166.

[70] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. 2023. Simple and Efficient Heterogeneous Graph Neural Network. In *AAAI* (Washington, DC, USA).

[71] Wei Ye, Omid Askarisichani, Alex T. Jones, and Ambuj K. Singh. 2023. Learning Deep Graph Representations via Convolutional Neural Networks (Extended abstract). In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 3831–3832.

[72] Yuyang Ye, Zheng Dong, Hengshu Zhu, Tong Xu, Xin Song, Runlong Yu, and Hui Xiong. 2023. MANE: Organizational Network Embedding With Multiplex Attentive Neural Networks. *TKDE* 35, 4 (2023), 4047–4061.

[73] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On Explainability of Graph Neural Networks via Subgraph Explorations. In *ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research)*, Marina Meila and Tong Zhang (Eds.), Vol. 139. 12241–12252.

[74] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 793–803. https://doi.org/10.1145/3292500.3330961

[75] Jiawei Zhang, Jianhui Chen, Shi Zhi, Yi Chang, Philip S. Yu, and Jiawei Han. 2017. Link Prediction across Aligned Networks with Sparse and Low Rank Matrix Estimation. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. IEEE Computer Society, 971–982.

[76] Junhua Zhang, Wentao Li, Long Yuan, Lu Qin, Ying Zhang, and Lijun Chang. 2022. Shortest-Path Queries on Complex Networks: Experiments, Analyses, and Improvement. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2640–2652.

[77] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5171–5181.

[78] Yufeng Zhang, Weiqing Wang, Hongzhi Yin, Pengpeng Zhao, Wei Chen, and Lei Zhao. 2023. Disconnected Emerging Knowledge Graph Oriented Inductive Link Prediction. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 381–393.

[79] Yiming Zhang, Lingfei Wu, Qi Shen, Yitong Pang, Zhihua Wei, Fangli Xu, Ethan Chang, and Bo Long. 2023. Graph Learning Augmented Heterogeneous Graph Neural Network for Social Recommendation. *ACM Trans. Recomm. Syst.* 1, 4, Article 16 (oct 2023), 22 pages. https://doi.org/10.1145/3610407

[80] Tianyu Zhao, Cheng Yang, Yibo Li, Quan Gan, Zhenyi Wang, Fengqi Liang, Huan Zhao, Yingxia Shao, Xiao Wang, and Chuan Shi. 2022. Space4HGNN: A Novel, Modularized and Reproducible Platform to Evaluate Heterogeneous Graph Neural Network. In *SIGIR* (Madrid, Spain). 2776–2789.

[81] Hongkuan Zhou, Ajitesh Srivastava, Hanqing Zeng, Rajgopal Kannan, and Viktor Prasanna. 2021. Accelerating Large Scale Real-Time GNN Inference using Channel Pruning. *Proceedings of the VLDB Endowment* 14, 9 (2021), 1597–1605.

[82] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. AliGraph: a comprehensive graph neural network platform. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2094–2105.