SCALABLE DATA ANALYTICS FOR RELATIONAL DATABASES, GRAPHS AND VIDEOS

Fubao Wu
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2

Part of the Databases and Information Systems Commons, Data Science Commons, and the Theory and Algorithms Commons

Recommended Citation
https://doi.org/10.7275/28440989 https://scholarworks.umass.edu/dissertations_2/2582

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
SCALABLE DATA ANALYTICS FOR RELATIONAL DATABASES, GRAPHS AND VIDEOS

A Dissertation Presented

by

FUBAO WU

Submitted to the Graduate School of the University of Massachusetts Amherst in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY

May 2022

Electrical and Computer Engineering
SCALABLE DATA ANALYTICS FOR RELATIONAL DATABASES, GRAPHS AND VIDEOS

A Dissertation Presented

by

FUBAO WU

Approved as to style and content by:

Lixin Gao, Chair
Prof. David Irwin
Prof. Tongping Liu
Andrew Lan

Christopher V. Hollot, Department Chair
Electrical and Computer Engineering
DEDICATION

*In loving memory of my dear father*
PhD study abroad is an important and long journey in my life. Now I am reaching to the end of this journey with great memories and gratitude. First of all, I would like to express my heartfelt gratitude to my advisor, Prof. Lixin Gao. I have been working closely with her during this journey. She is a knowledgeable, insightful and responsible advisor who leads me into the essence of academic exploration. I was inspired and learned a lot from her about the passion and rigor in the exploration of new unknown areas. Moreover, I was also given a good mix of the intellectual freedom that I craved and the scientific rigor that I needed. I would wholeheartedly thank Prof. David Irwin as my PhD qualifying exam and dissertation committee member. I have obtained great inspirations from his comments and suggestions. I would be also deeply grateful for Prof. Tongping Liu and Prof. Andrew Liu who are on my dissertation committee. They have given great supports and comments through my dissertation stages.

I was honored to join in the Multimedia Networking & Internet Lab working with excellent labmates. Many thanks to Samamon Khemmarat, Jiangtao Yin, Guoyi Zhao, Xiaozhe Shao, Tian Zhou, Hao Zhang, Kaishua Liu, Qianqian Gao, Xi Wang, Li Wang, Jianzhou Chen, and Hongyu Cai. I have had a great time in studying with them, and have learned a lot from them.

I can not express my gratitude enough to my family members who are far away from me. They have given continuous supports all the way through my long journey. I would also express my endless gratitude to my life partner. I can not achieve this without her. I appreciate what she have said and done to lift me up whenever I was confused, exhausted and struggled.
ABSTRACT

SCALABLE DATA ANALYTICS FOR RELATIONAL DATABASES, GRAPHS AND VIDEOS

MAY 2022

FUBAO WU

B.Eng., NORTHEASTERN UNIVERSITY (CN)
M.Eng., UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA
M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST
Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Lixin Gao

Data analytics is to analyze raw data and mine insights, trends, and patterns from them. Due to the dramatic increase in data volume and size in recent years with the development of big data and cloud storage, big data analytics algorithms and techniques have been faced with more challenges. Moreover, there are various types of data formats, such as relational databases, text data, audio data, and image/video data. It is challenging to generate a unified framework or algorithm for data analytics on various data formats. Different data formats still need refined and scalable algorithms. In this dissertation, we explore three types of data formats, relational databases, graph databases, and video data for scalable processing. First, with the increase in the big volume of data, business organizations, governments, and other institutes need to generate insights from the data. The relationships in
the data matter more than just the individual data points. In order to leverage data relationships and easier scaling, organizations use the graph database to store the relationship information as a first-class entity. We analyze the large network management databases from Cisco, and propose a comprehensive algorithm to transfer structured data formatted in relational databases to graph data organized in a knowledge graph database. We model this problem as the field and record matching in the relational database. We then develop a matching technique for large network management databases deploying instance-level matching for effective data matching. Second, we explore the graph analytics problem for the fundamental component—graph queries that answer user queries efficiently and effectively. State-of-the-art graph query approaches mainly focus on the inference of node labels and neighborhood structures through path propagation. However, entity links in the real world also contain rich hierarchical inheritance relations. For example, the vulnerability of a product version is likely to be inherited from its older version. Taking advantage of the hierarchical inheritances can potentially improve the quality of query results. We take into account this useful dimension, hierarchical inheritance relations, to improve the state-of-the-art graph query approaches. Third, we investigate the video data from video streams for video analytics. Video analytics is to detect and track objects from video streams with many applications in traffic control, security monitoring, event analysis, etc. It involves selecting the best configuration of frame rate and resolution to achieve a certain accuracy in real-time. State-of-the-art switching approaches adjust configurations by profiling video clips on a large configuration space with much compute resources. We propose an approach that adapts the configuration by analyzing past video analytics results instead of profiling candidate configurations. Our approach adopts a lower/higher resolution or frame rate when objects move slow/fast. We train a model that automatically selects the best configuration. We evaluate our approach on two real-world video analytics applications: traffic tracking and pose es-
timation, and obtain superior performance compared to the state-of-the-art switching methods. Finally, more and more video-analytics as a service in clouds with private data draws researchers’ attention to privacy concerns. Fully homomorphic encryption (FHE) is one of the promising ways to achieve the privacy-preserving that is a hot topic in the academic and industry communities. Recent work on privacy-preserving deep learning has successfully explored the feasibility of image classification with up to 20 deep neural network layers. In our final work, we explore the feasibility of FHE on encrypted frames for the video analytics pose estimation application with more deep neural network layers. We develop a privacy-preserving pose estimation system based on the FHE SEAL library on a CPU server. It demonstrates the potential and feasibility of privacy-preserving video analytics.
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>v</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xiv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xvi</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Relational Database Integration to A Graph Database</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Graph Query for Graph Analytics</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Configuration Adaptation for Video Analytics</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Privacy-Preserving Video Analytics</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Contributions</td>
<td>12</td>
</tr>
<tr>
<td>2. SCALABLE RELATIONAL DATABASES INTEGRATION TO A GRAPH DATABASE</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Problem Description</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Matching Algorithm</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1 System Overview</td>
<td>21</td>
</tr>
<tr>
<td>2.3.2 Numerical Field Matching</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2.1 Range Difference and Bucket Dot Product Similarity Metrics</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2.2 SVM Classification-based Matching</td>
<td>27</td>
</tr>
<tr>
<td>2.3.3 Non-numerical Field Matching</td>
<td>28</td>
</tr>
<tr>
<td>2.3.3.1 Top Priority Match Metric</td>
<td>28</td>
</tr>
</tbody>
</table>
2.3.3.2 MinHash-Locality Sensitive Matching .............. 32
2.3.3.3 SVM Classification-based Matching .................. 35

2.4 Experimental Evaluation .................................. 35
  2.4.1 Dataset ............................................. 36
  2.4.2 Experimental Setup .................................. 36
  2.4.3 Evaluation based on Numerical Data ................ 37
    2.4.3.1 Evaluating of Ground Truth ..................... 37
    2.4.3.2 Top-20 Similarity Results ...................... 41
  2.4.4 Evaluation based on Non-numerical Data ............. 42
    2.4.4.1 Evaluating of Ground Truth ..................... 42
    2.4.4.2 Top-20 Similarity Results ...................... 45
  2.4.5 Evaluation of SVM Classification-based Matching .... 45
    2.4.5.1 Extending Ground Truth ......................... 45
    2.4.5.2 SVM Classification Results ...................... 47
  2.4.6 Comparisons with Other Existing Algorithms ....... 48
    2.4.6.1 Comparison of quality .......................... 49
    2.4.6.2 Comparisons of Mismatched Examples ............ 50
    2.4.6.3 Comparison of Efficiency ....................... 53

2.5 Related Work ............................................ 53
2.6 Conclusion ............................................... 55

3. SCALABLE GRAPH QUERY ON A GRAPH DATABASE WITH HIERARCHICAL INHERITANCE RELATIONS ... 57

3.1 Introduction ............................................. 57
3.2 Problem Formulation ..................................... 61
  3.2.1 Data Graph, Query Graph and Matching .............. 61
  3.2.2 Matching Score ..................................... 64
  3.2.3 Problem Statement: ................................ 66

3.3 Graph Query Algorithm with Hierarchical Inheritance Relations .... 66
  3.3.1 Matching Score for Star Query ....................... 68
  3.3.2 Bound-based Pruning for Star Query ................. 68
    3.3.2.1 Bounds of Matching Score ...................... 68
3.3.2.2 Bounds of Node Closeness Score ......................... 69
3.3.3 Top-k Selection with Bounds ................................. 69
3.3.4 Convergence of Iteration Propagation ....................... 70
3.3.5 Star Query Algorithm ........................................ 70
3.3.6 General Graph Query Algorithm ............................. 71
  3.3.6.1 Matching Score of Query Nodes ......................... 71
  3.3.6.2 Algorithm Flow ........................................ 73
  3.3.6.3 Candidate Selections with Branch and Bound
         Pruning ................................................. 73
3.4 Distributed Implementation ..................................... 74
3.5 Experimental Evaluation ....................................... 76
  3.5.1 Datasets ................................................. 76
  3.5.2 Quality of Graph Query .................................. 77
  3.5.3 Comparisons of Query Quality ............................ 77
    3.5.3.1 Overall Comparison of Query Quality ................ 78
    3.5.3.2 An Example of Query Quality Comparison with
            GStar ............................................. 78
    3.5.3.3 An Example of Query Quality Comparison with
            NeMa ............................................. 78
  3.5.4 Efficiency of Graph Query ................................. 82
3.6 Related Work ................................................ 85
  3.6.1 Structured Graph Query: ................................ 85
  3.6.2 Top-k Graph Query: ...................................... 85
3.7 Conclusion ................................................... 86

4. REAL-TIME CONFIGURATION ADAPTATION FOR VIDEO
   ANALYTICS THROUGH OBJECT TRACKING ....................... 87
  4.1 Introduction ................................................. 87
  4.2 Preliminary ................................................ 89
    4.2.1 Video analytics Pipeline for Configuration Adaptation 89
    4.2.2 Metrics for Video Analytics ............................ 90
  4.3 Motivation .................................................. 91
  4.4 System Architecture .......................................... 92
  4.5 Movement Features and Prediction Model .................... 93
5.5.1 Environment and Parameter Setting ................................. 121
5.5.2 Encryption Efficiency .................................................. 122
5.5.3 Inference Efficiency of Neural Network Layers and Activation
    Functions ................................................................. 122
5.5.4 Performance of FHE on Pose Estimation ......................... 122
5.5.5 Performance of FHE on Pose Estimation with
    Preprocessing ........................................................... 123
5.6 Related Work ............................................................... 124
    5.6.1 Secure Multi-party Computation ................................. 125
    5.6.2 Fully Homomorphic Encryption for Deep Learning ........... 125
5.7 Conclusion ................................................................. 126

6. CONCLUSIONS .................................................................... 127

BIBLIOGRAPHY .................................................................... 129
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 PRODUCT ((T_p))</td>
<td>18</td>
</tr>
<tr>
<td>2.2 INCIDENT ((T_i))</td>
<td>18</td>
</tr>
<tr>
<td>2.3 ORDER ((T_o))</td>
<td>18</td>
</tr>
<tr>
<td>2.4 Example of numerical ground truth field pairs</td>
<td>38</td>
</tr>
<tr>
<td>2.5 Top-20 similarity result of numerical field pairs</td>
<td>42</td>
</tr>
<tr>
<td>2.6 Examples of non-numerical ground truth</td>
<td>43</td>
</tr>
<tr>
<td>2.7 Sample of non-numerical record pairs</td>
<td>43</td>
</tr>
<tr>
<td>2.8 Top-20 matching results of non-numerical field pairs</td>
<td>46</td>
</tr>
<tr>
<td>2.9 SVM validation and testing results on synthetic numerical and non-numerical ground truth</td>
<td>48</td>
</tr>
<tr>
<td>2.10 Quality comparisons with other existing algorithms</td>
<td>51</td>
</tr>
<tr>
<td>2.11 Efficiency comparison with other existing algorithms</td>
<td>51</td>
</tr>
<tr>
<td>2.12 Example of field pairs matched by NEMA, but not by COMA</td>
<td>52</td>
</tr>
<tr>
<td>2.13 Example of field pairs matched by COMA, but not by NEMA</td>
<td>52</td>
</tr>
<tr>
<td>2.14 Examples of record similarity comparisons</td>
<td>52</td>
</tr>
<tr>
<td>3.1 Data set statistics</td>
<td>76</td>
</tr>
<tr>
<td>3.2 The overall query quality comparisons on the three datasets</td>
<td>79</td>
</tr>
<tr>
<td>3.3 One query result of GQH and GStar in the Cisco dataset</td>
<td>79</td>
</tr>
</tbody>
</table>
3.4 One query result of GQH and NeMa in the Cisco dataset ............ 80
4.1 Video dataset statistics ..................................................... 104
4.2 Impact of features on prediction accuracy ............................... 106
5.1 The neural network layers used for FHE on pose estimation .......... 117
5.2 Encryption parameter setting ............................................. 122
5.3 Encryption efficiency with FHE .......................................... 122
5.4 Inference time with different layers and activation function for a frame ................................................................. 123
5.5 The performance of FHE pose estimation and plaintext pose estimation ................................................................. 123
5.6 Comparison of efficiency of FHE w/ and w/o processing ............ 124
5.7 Pose estimation performance with preprocessing ....................... 124
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Graph database created from Table 2.1, 2.2 and 2.3.</td>
</tr>
<tr>
<td>2.2</td>
<td>System diagram of automatic integration for network management databases.</td>
</tr>
<tr>
<td>2.3</td>
<td>Numerical field matching flow.</td>
</tr>
<tr>
<td>2.4</td>
<td>Non-numerical field matching flow based on TPM.</td>
</tr>
<tr>
<td>2.5</td>
<td>Non-numerical field matching flow based on MH-LSH.</td>
</tr>
<tr>
<td>2.6</td>
<td>Matching with Jaccard similarity metric.</td>
</tr>
<tr>
<td>2.7</td>
<td>Matching with combined RDS and BDPS.</td>
</tr>
<tr>
<td>2.8</td>
<td>Selection of the best $T_b$ based on the matching accuracy.</td>
</tr>
<tr>
<td>2.9</td>
<td>TPM-based matching ratio score.</td>
</tr>
<tr>
<td>2.10</td>
<td>MH-LSH-based matching score.</td>
</tr>
<tr>
<td>2.11</td>
<td>RDS values for original and synthetic field pairs in numerical fields.</td>
</tr>
<tr>
<td>2.12</td>
<td>MRS values for original and synthetic field pairs in non-numerical fields.</td>
</tr>
<tr>
<td>3.1</td>
<td>A product information network schema and a query example. (a) A graph schema of a product network. (b) An example of a hierarchical connection structure for product versions. (c) A user query graph of top products. (d) A answer graph with top-5 products.</td>
</tr>
<tr>
<td>3.2</td>
<td>Hierarchical inheritance cases and scores</td>
</tr>
</tbody>
</table>
3.3 An example of candidate selection .............................................. 75
3.4 Distributed graph query architecture ........................................ 75
3.5 Star graph queries and their results (a) Hierarchical star query graph in Cisco – Query the products with Vulnerability $V_1$ and used Technology $T_1$, and its top-5 results in order. (b) Mixed star query graph in Dblp – Query authors that work on Topic $T_1$ and cooperate with Person $PP_1$, and its top-5 results in order. .................. 81
3.6 Non-star graph queries and their results (a) Mixed general query graph in Cisco – Query a product that has Vulnerability $V_1$ and $V_2$, another product that belongs to a workgroup $W_1$, and their common technology also used in a product $P_1$, and the top-1 result. (b) Mixed general query graph in Dblp – Query an author that works on Topic $T_1$ and $T_2$, and cooperates with another query person that works on Topic $T_3$, and published a paper coauthored with Person $PP_1$ in year $Di_1$, and its top-1 result. ........... 82
3.7 Efficiency and scalability on three datasets ................................. 84
4.1 video processing pipeline ............................................................ 88
4.2 A video analytics module with an adaptive configuration controller for video inference ................................................................. 90
4.3 The correlation between movement speed and the necessary frame rate/resolution for the accuracy requirement. ......................... 93
4.4 Our proposed video analytics system ........................................... 94
4.5 Object detection result estimation for skipped frames .................... 99
4.6 Comparison of MOTrack with one-time profiling and periodic profiling on traffic tracking and pose estimation ......................... 107
4.7 Accuracy and processing time for one video clip on traffic tracking .......................................................... 108
4.8 Accuracy and processing time for one video clip on pose estimation .......................................................... 108
4.9 Processing time of MOTrack and ground truth configuration for one video clip on traffic tracking and pose estimation .................. 109
4.10 Configuration adaptation time of MOTrack compared with periodic profiling ................................................ 109

5.1 The diagram of FHE ................................................................. 116

5.2 Plaintext pose estimation neural network model with multi-stages .... 117

5.3 CKKS scheme overview [66] ................................................... 118
CHAPTER 1
INTRODUCTION

With the development of internet, multimedia, and cloud storage in recent years, data are increasingly generated, copied and stored. Data come from various sources such as cameras, phones, computers, sensors and other different kinds of machines and facilities. Recent development in social media, sensor networks, cyber-physical systems, and the Internet of Things (IoT) have increased the collection of data. It is estimated that there are 2.5 quintillion bytes of data created each day in 2018 [11]. From a statistic [2], the world data volume has been increasing exponentially in the last 10 years and will keep exploding every year. From the International Data Corporation (IDC) estimation, the amount of generated data will double every 2 years [113]. Data are obviously staggeringly increasing on a daily basis. With a huge amount of data, the concept of big data has gained popular in the last 10 years. Big data [89, 91, 108, 122] represents large and complex data sets from various sources, and contains 4 main characteristics—volume, variety, velocity and veracity. As a result, this drives the development of data analytics techniques and algorithms to mine useful information from a huge amount of data.

Data analytics involves collecting, organizing and analyzing large amounts of data. The main core and goal for data analytics is to uncover hidden patterns, correlations and other insights. Data analytics has a large number of applications [54] in lots of areas of e-commerce, manufacturing, transportation, healthcare, finance, banking and insurance, and so on. For example, it could help identify consumer behaviors and recommend related products in e-commerce. In the manufacturing, it is useful to
monitor system, maintain scheduling and detect system faults. In the transportation, it could help traffic planning, scheduling and driving experience improvement, and so on. More and more applications benefit from the effective and efficient data analytics techniques and algorithms.

Data analytics for effectively and efficiently mining useful patterns and insights from large amounts of data faces many challenges. The first challenging is the exponentially increasing amount of data volume with time [123]. Mining the useful information from a huge amount of data requires more scalable algorithms and techniques and/or powerful computers and servers. The second challenge for data analytics is that various problems and topics could be targeted [142]. Query, search, prediction, detection, clustering and inference, etc. are common problems for data analytics. Different problems also require different algorithms or techniques. The third challenge comes from the data format [74]. Data could be categorized into structured data and unstructured data according to the different types of formats. Structured data involves a fixed data model, schema, or representation, commonly in the form of relational databases. Unstructured data do not have the fixed data model including text, audio, image, video, and other multimedia, etc. There is no one universal algorithm or technique that could work well on all data formats. Data analytics on these different data formats need different algorithms or techniques. The fourth important common challenges are the data governance. There are private and sensitive data concerns and securities for storage and analysis. How to protect and utilize the private data poses a big challenge to data analytics.

With the different challenges of data analytics, more and more advanced techniques and algorithms [156, 51, 18, 105, 35] are being developed recently to address these different challenges. The data analytics techniques and algorithms have been evolving with the increasing of data. Relational database technique have been widely used for decades with convenient scheme structures. Graph databases utilize the
connection of relationship in data points and have gained popularity in many areas. Moreover, videos are also becoming one of the main sources of increasing data. Even though they exhibit different data representations, there exist some commonalities in data analytics. Mining patterns and insights involves extracting the useful information from a huge amount of data. Relational databases, graphs and videos contain rich information with redundant information in them. The techniques and algorithms of data analytics in these data representations all involve removing the redundant information and obtaining the insightful information. The recent advancement of machine learning and artificial intelligence techniques has played an important role in data analytics techniques and algorithms. The machine learning and artificial intelligence algorithms could also be explored in these data representations.

In this dissertation, we propose novel machine learning and artificial intelligence techniques to mine useful patterns and insights from a huge amount of data. We utilize the redundancy of large amounts of data in different representations of relational databases, graphs and videos. Through analyzing the characteristics of each data representation, we devise novel and advanced algorithms to tackle the challenges of data analytics. Specifically, we explore four interesting aspects of data analytics.

- We integrate these relational databases by matching columns into a graph database for further graph analytics.
- We consider query answers from the graph database and propose a graph query with hierarchical inheritance relations.
- We explore the video analytics pipeline and propose an effective and efficient video analytics system with configuration adaptation through object movement tracking.
• We investigate the plausibility of privacy preserving video analytics with fully homomorphic encryption and are first to develop a pose estimation application with fully homomorphic encryption.

1.1 Relational Database Integration to A Graph Database

With more and more data generated and stored in relational databases, scalable network measurement and analysis techniques have been used in finding hidden information or patterns to help with network query, management, network monitoring and network security [164, 62, 141]. In one scenario of important network management, big network operators/vendors serving various customers own a multitude of databases on network products, configurations, incidents, troubleshooting and diagnosis information, etc. Since the databases serve different departments, they are usually separated from each other and independently managed by different departments and groups. However, network data are inherently designed to host “connections” among different devices, groups that the devices are in, and functionality the devices serve together; they are often required to be shared and put together used in many important tasks such as network prediction, semantic query, network diagnosis and fault detection. With the database connection and integration, network administrators can easily query related product configurations and performance around devices or services. This could also help them automatically identify the correlated network trouble tickets/issues and pinpoint problems [124, 3]. As many studies [34, 21, 147] point out, the discovery of matching fields is the most crucial and foremost step for the integration of databases. Moreover, the relationship between data points is more important than individual points themselves. Matching them together to utilize the data relationship in a graph database format is more efficient network query, management and analysis. In this work, we focus on matching fields in the relational network management databases and construct them as a graph database efficiently.
Lots of research on field matching and integration approaches exists for different data formats in different contexts such as relational databases, XML and object-oriented data formats [53, 132, 152, 19]. Our work focuses on the database matching. Existing database matching approaches contain two main categories of techniques. The first category is based on schema-level matching, which utilizes metadata based on schema characteristics such as field names, data types, structural properties and other schema information [13, 55, 7]. However, network management databases from different sources have different design standards and naming conventions [73, 33]. Even similar fields can have different names (e.g. “product_family” can also be named as “product series”). The second category is instance-level matching, which uses the record values of two fields to obtain the similarity and determine matched fields [26, 129, 121, 114, 140]. Most of the previously proposed schemes rely on syntactic similarities, sampling or machine learning techniques that are meant to extract common patterns from the matching data corpus. However, it is challenging to reliably construct dictionaries and corpuses with large datasets when the naming convention is not consistent and diverse. For the network management database, we need to mine and explore the characteristics of databases themselves to construct our unique approaches for matching.

The challenges for matching these network databases are as follows:

- The database design is not ideally uniform. The data tables are created in different groups and departments by different people. Therefore, it is not reliable to use schema information to match data directly and easily.

- The data is noisy and irregular. Some table fields contain unexpected records such as null, invalid values and typos. Some table records are either partly missing, incomplete or incorrect. Some fields have a large amount of records, while some have very few records.
• The table contents are complicated and heterogeneous with numerical and non-
numerical data formats.

• No thesauri or auxiliary information exist that we can rely on for matching.
The only set of observation available is the database itself.

1.2 Graph Query for Graph Analytics

Lots of real-world applications and information systems, such as enterprise net-
works, social networks, and biological networks, chemical networks, traffic network
that have the data point relationship can be modeled as heterogeneous information
networks (HIN) [37, 56, 143, 151, 46, 191, 186]. Heterogeneous information networks
(HINs) could be considered a knowledge graph database used to model information
systems which contains multiple types of objects and relations providing rich graph
analytics in semantic queries, knowledge discovery, information fusions, recommenda-
tions, predictions and inferences. Graph query, as an important technique for doing
these graph analytics, has been extensively explored recently. It mainly explores sub-
graph isomorphism algorithms to get an exact match [49, 50, 172], and also develops
subgraph matching algorithms to do an inexact/approximate match as the potential
query answers [83, 77, 157].

Current state-of-the-art research on graph query/matching mainly focuses on two
dimensions. The first dimension is the unary node-to-node property mapping. The
second dimension is edge-to-edge/path similarities. Jin et al. [77, 175, 155, 78]
consider node types and closest path propagation to get scores of query answers. Some
work [58, 83, 174, 77, 120, 100] considers similar nodes’ labels and their neighbors to
learn the path propagation to get ranked answers.

However, knowledge representation has hierarchical structures in the real world
system [106, 76, 32, 187]. Long et al. [106] state that the knowledge structure
representation can be inherited with upward and downward inheritances. Clauset et
al. [32] show that the existing knowledge of hierarchical structure can be used to predict missing connections. In addition, Jiang et al. [76] construct the hierarchical structures of entities for the large freebase knowledge base system based on real world entities and relations. One visible example in an enterprise’s product databases is that product vulnerabilities can be inherited from or passed down to different product versions.

While measuring the similarity of objects for graph matching, hierarchical inheritance relations can also play an important role for the answer ranking. Lin et al. [103] investigate the ad-hoc entity retrieval from a knowledge graph with hierarchical entity types and entity descriptions by encoding the nodes and their labels into a Markov random field-based framework via a path aware smoothing method, and improve the quality of entity retrieval from the knowledge graph. The quality of query answers can also be greatly affected by hierarchical inheritance relations. Therefore, we consider the power of hierarchical inheritance whereby a subclass inherits the properties and constraints of its parents, and more meaningful and accurate query answers are expected to be obtained.

Taking an example of an knowledge graph database with hierarchical structures with product types and other property types. The product type is connected by four property types: site, workgroup, technology and vulnerability. Product entities could have hierarchical connections with different versions of products. Some properties are inherited among different versions of products, such as vulnerability and technology properties. For example, if the user wants to query which product version has a certain specific vulnerabilities, with the hierarchical information modeled in the graph database, it would be more efficient and effective to find out the correlated product versions in hierarchical relations. In this way, the graph database modeled as hierarchical relations play an important part.
The challenges for knowledge graph query with hierarchical inheritance relations are

- The knowledge graph databases contain multiple entity types, labels, and various relations with complex neighbors or structures.
- The knowledge graph contains rich information in a substantial size, making efficient query answers more challenging.
- The user query graph is complex and variable in various formats and structures, which is also difficult to parse query intents.
- No available knowledge graphs with hierarchical inheritance relations constructed exist, in which we have created hierarchical structures for user queries.
- There are no existing query graph algorithms with hierarchical inheritance relation for reference. We have to formulate the query problems from scratch and construct the algorithms.

1.3 Configuration Adaptation for Video Analytics

Many cities like New York City, Beijing, and London have deployed millions of video cameras [1, 180, 47]. Some of the cameras are installed outside on the street for traffic control and crime, while the cameras installed inside buildings are for surveillance and business intelligence. The pervasive deployments of video cameras drive the big demand for video analytics. Object detection and tracking as the common parts of video analytics focuses on detecting and tracking objects from video streams. Many applications in traffic control, business intelligence, action/event analysis, and VR/AR are built on top of object detection and tracking [75, 23]. Recently, object detection and tracking relies on deep neural network (DNN) models for more accurate inference. Each DNN module corresponds to a frame rate and resolution, which is referred to as a configuration in this work.
The selection of a configuration impacts the accuracy and resource consumption of object detection and tracking. A configuration with high frame rate and resolution is considered as an expensive configuration, and a configuration with low frame rate and resolution is considered as a cheap configuration. For example, an expensive configuration with 25 fps and 1080p generally leads to a higher accuracy than a cheap configuration with 1 fps and 240p, but consumes more resources. The “best” configuration is the configuration that achieves a desired accuracy with the least resource consumption.

The state-of-the-art configuration adaptation falls into two classes: one-time profiling and periodical profiling. One-time profiling [59, 181, 184] aims to profile all configurations only once during the beginning of a video (e.g., 10 seconds), and then chooses the best configuration above an accuracy requirement for the video. However, it uses a fixed configuration for the whole video and neglects the intrinsic dynamics of video contents. Periodic profiling [179, 130, 75] determines the configuration for every interval of a video through profiling the first few frames of the interval. However, profiling periodically costs compute resources and incurs processing time.

Video content exhibits temporal and spatial characteristics. Due to the temporal and spatial characteristics, objects keep the same or similar movement in a short period of time. Therefore, it could be quantified with object movement in this period. Objects moving fast usually need expensive configurations to track, and cheap configurations suffice for slow objects.

The challenges for video analytics configuration adaptation are as follows:

- The video contents of a video stream change dynamically with different objects, light variation and conditions, which impacts the selection of configurations.

- The configuration space for video analytics is usually large with lots of combinations, which makes scheduling among them challenging.
• The video analytics applications are various and have different accuracy and resource consumption requirements, making the configuration adaptation complex.

1.4 Privacy-Preserving Video Analytics

Video analytics is increasingly utilized for traffic control, business intelligence, action/event analysis, human-machine interaction, VR/AR, etc. Lots of video data are increasingly generated every day [95]. More and more applications for video analytics are currently done through cloud servers with powerful computation capacities with deep learning models. This makes the video analytics as a service on cloud servers an effective and efficient option. In this case, lots of data owners’ private data have to be transmitted to the cloud servers that might be distrusted to data owners, such as sensitive patients’ data in hospitals and clients’ personal activity data in stores. The video analytics involving personal activity and tracking are private and sensitive. Therefore, to effectively use the deep learning or video analytics as a service, the demand on privacy-preserving deep learning as a service is increasing.

Privacy-preserving deep learning are a hot topic recently and mainly divided into two types. One type method is through the secure multi-party computation [40, 86], which is to create certain methods for multiple parties to jointly compute a function over their input data and also keep those input data private. The second type of method is through cryptography. With cryptography, homomorphic encryption (HE), garbled circuits, secret sharing and secure processors are the most widely used cryptographic techniques to achieve privacy-preserving deep learning [4, 128]. Especially, homomorphic encryption as a promising method and became more and more capable with its flexibility and security. Our work explores the feasibility of fully homomorphic encryption (FHE) for video analytics. FHE could allow computations on the encrypted data without disclosing the raw data, making it suitable for privacy-
preserving in the clouds. FHE for deep learning is an on-going hot research topic in recent years. There exists some work on FHE for deep learning exploring the feasibility of tasks with deep neural networks. Due to the commonly used deep neural networks for lots of applications and the highly expensive computations involved in the FHE, there are emerging research on how to apply FHE to deep neural network based tasks such as image classification. CryptoNets [52] is one of initiators that explores deep neural network with MNIST optical character recognition tasks with 5–9 layers of neural network based on Simple Encrypted Arithmetic Library (SEAL) [22] for homomorphic encryption. Juvekar et al. [79] combine homomorphic encryption and garbled circuit and experiment on the image classification on MNIST and CIFAR-10 datasets for 5 layer of neural networks. Lee et al. [93] explore FHE on more deep layers with ResNet-20 image classification with approximation and bootstrapping, which improves the efficiency based on RNS-CKKS homomorphic encryption scheme. It achieves the average classification accuracy on encrypted images with an accuracy of around 90.67% and about $10^6$ magnitude more inference time compared to the classification on raw images.

The challenges for privacy-preserving video analytics with fully homomorphic encryption are as follows:

- The higher the required security level of fully homomorphic encryption, the more computations are involved.

- Fully homomorphic encryption supports only limited arithmetic operations of addition, multiplication and scaling. Deep neural networks involve lots of non-arithmetic operations, which need the research for approximation and thus impact the accuracy.

- A deep neural network model for video analytics application generally involves deep layers. The multiplication operation between two or more ciphertexts is
bounded due to the increasing noises. This would lead to the incorrectness of decryption. Recent proposed work of fully homomorphic encryption with bootstrapping technique has been shown to work on a few deep neural layers. However, its is still unknown to a large amount of layers.

1.5 Contributions

The major contributions of this dissertation are to investigate scalable data analytics to mine useful patterns and insights with the increasing of large amounts of data. We propose novel machine learning and artificial intelligence algorithms to conquer the challenges of data analytics. From the relational databases, graphs and videos, we explore the characteristics of these types of data representations and propose corresponding algorithms to mine useful patterns from them. More specifically, our main contributions are as follows.

1. We propose an automatic matching technique for large NEtwork MAnagement databases (NEMA) to construct a graph database for network management and analysis effectively and efficiently. We propose several algorithms for numerical and non-numerical field matching respectively based on instance matching. Specifically, we propose effective range difference and bucket dot product similarity metrics to match numerical fields, and the top priority metric to match non-numerical fields. Then, to make the algorithm more scalable for large network management databases, we utilize min-hash-locality sensitive hashing algorithm for faster processing with a little scarification of accuracy. Furthermore, we propose to use the proposed similarity metric scores as features for classification to improve the reliability of our matching algorithms. Then we experimentally demonstrate the effectiveness and efficiency of our matching algorithms in the real Cisco network management databases.
2. We model the graph query with a new matching score function considering hierarchical inheritance relations for effective answers, and by proposing a bound-based technique for an efficient query. Specifically, we formulate the graph query problem with hierarchical inheritance relations to improve query quality. Then we propose a new graph query algorithm based on uniform cost search in the context of a new matching score function. We also design a bound-based method to prune search spaces to efficiently get the top-$k$ best answers. Finally, we implement our algorithm in the Spark GraphX distributed environment for large-scale knowledge graphs and evaluate the effectiveness and efficiency of our matching algorithm.

3. We conquer the video analytics problem by capturing the object movement from past video analytics results to guide the selection of frame rate and resolution to adapt configurations over time. Leveraging this, a machine learning-based classification method, MOTrack, is utilized to obtain the relationship between object movement and the best configuration. We obtain the estimated object movement and corresponding configurations as labeled training data instances to automatically learn the mapping between them. Specifically, we investigate the impact of object movement on configurations and propose to utilize the object movement to guide configuration adaptation. Then we propose a machine learning-based classification method to predict the configuration for future frames, which significantly reduces the cost of configuration adaptation. Finally, we experimentally demonstrate the effectiveness and efficiency of MOTrack on traffic tracking and pose estimation applications.

4. We further investigate the feasibility of video analytics as a service in cloud servers with privacy concern. Fully homomorphic encryption is one of currently plausible ways to achieve privacy concern. We develop a privacy-preserving
model for video analytics pose estimation on encrypted data with fully homomorphic encryption successfully. We design different approximations of deep neural network layers and activation functions. The system shows the 86.7% agreement of accuracy on encrypted data compared with the original model on raw data. Moreover, it achieves 252,201 second per frame on encrypted data with FHE. It demonstrates the potential and feasibility of privacy-preserving video analytics.

The rest of this dissertation is organized as follows. In Chapter 2, we explore the relational databases and propose the integration method of relational network management databases to a graph database. Chapter 3 presents the graph analytics algorithm for graph query algorithm with hierarchical inheritance relations. In Chapter 4, we explore the video analytics problem with configuration adaption through movement prediction. Chapter 5 shows the privacy-preserving system for video analytics. We conclude this dissertation in Chapter 6.
2.1 Introduction

With the development of big data analytics and data mining techniques, scalable network measurement and analysis techniques have been used in finding hidden information or patterns to help with network management, network monitoring, and network security [164, 62, 141]. In one scenario of important network management, big network operators/vendors serving various customers own a multitude of databases on network products, configurations, incidents, troubleshooting and diagnosis information, etc. Since the databases serve different departments, they are usually separated from each other and independently managed by different departments and groups. However, network data are inherently designed to host “connections” among different devices, groups that the devices are in, and functionality the devices serve together; they are often required to be shared and put together used in many important tasks such as network prediction, semantic query, network diagnosis and fault detection. With the database connection and integration, network administrators can easily query related product configurations and performances around devices or services. This could also help them automatically identify the correlated network trouble tickets/issues and pinpoint problems [124, 3]. As many studies [34, 21, 147] point out, the discovery of matching fields is the most crucial and foremost step for the integration of databases. For this reason, in this work we aim to automatically construct such matchings that lead to efficient network management and analysis.
There is an abundance of research on field matching and integration approaches for different data formats in different contexts such as relational databases, XML and object-oriented data formats [53, 132, 152, 19]. Existing database matching approaches include two main categories of techniques. One is based on schema-level matching, which exploits metadata using schema characteristics such as field names, data types, structural properties and other schema information [13, 55]. However, network management databases from different sources have different design standards and naming conventions [73, 33]. Even similar fields can have different names (e.g. “product_family” can also be named as “product series”). The other category is instance-level matching, which uses the record values of two fields to obtain the similarity and determine matched fields [26, 129, 121, 114]. Most of the previously proposed schemes rely on syntactic similarities, sampling or machine learning techniques that are meant to extract common patterns from the matching data corpus. However, it is difficult to directly apply these techniques to network databases or challenging to reliably construct dictionaries, corpus with large datasets when the naming convention is not consistent and diverse.

The challenges for matching these network databases are: (1) The database design is not ideally uniform. The data tables are created in different groups and departments by different people. Therefore, it is not reliable to use schema information to match data directly and easily. (2) The data is noisy and irregular. Some table fields contain unexpected records such as null, invalid values and typos. Some table records are either partly missing, incomplete or incorrect. Some fields have a large amount of records, while some have very few records. (3) The table contents are complicated and heterogeneous with numerical and non-numerical data formats. (4) No thesauri or auxiliary information exist that we can rely on for matching. The only set of observations available is the database itself.
To solve these challenges, we propose an automatic matching technique for large NEtwork MAnagement databases (NEMA) to construct a graph database for network management and analysis effectively and efficiently. We propose several algorithms for numerical and non-numerical field matching respectively based on instance matching. Our main contributions are as follows:

1. We propose effective range difference and bucket dot product similarity metric to match numerical fields, and top priority metrics to match non-numerical fields.

2. To make the algorithm more scalable for large network management databases, we utilize min-hash-locality sensitive hashing algorithm for faster processing with a little scarification of accuracy.

3. We further propose to use the proposed similarity metric scores as features for classification to improve the reliability of our matching technique.

4. We experimentally demonstrate the effectiveness and efficiency of our matching algorithms in the real Cisco network management databases.

We define the problem in Section 2.2. Section 2.3 describes NEMA matching algorithms in detail, including both numerical field matching and non-numerical field matching. Experimental evaluation is presented in Section 2.4 and related work is shown in Section 2.5. We conclude our work in Section 2.6.

2.2 Problem Description

Given structured network management databases, our goal is to create a graph database of network management by finding the most accurate matching field pairs among different tables in these databases. The matching of two fields is determined by the matching score measured by their record pair similarities. We utilize the matched
results to construct a graph database for semantic query, network analysis, network prediction, etc. [116, 107].

To illustrate our problem and algorithms clearly, we use three sample tables below as a toy example throughout the rest of this work. In Table 2.1, PRODUCT Table ($T_P$) contains 2 fields $product\_id$ (primary key), and $family$ with 7 records, respectively. In Table 2.2, INCIDENT Table ($T_I$) contains 2 fields $incident\_key$ (primary key), and $prod\_key$ with 7 records respectively. In Table 2.3, ORDER Table ($T_O$) contains 3 fields $order\_key$ (primary key), $incident\_id$, and $product\_name$ with 7 records respectively. The problem is to find whether these 7 fields match among the Table 2.1, 2.2 and 2.3 by evaluating the matching of their records, then to construct a graph database for network analysis and management.

**Table 2.1: PRODUCT ($T_P$)**

<table>
<thead>
<tr>
<th>$product_id$</th>
<th>$family$</th>
</tr>
</thead>
<tbody>
<tr>
<td>107</td>
<td>AIR series</td>
</tr>
<tr>
<td>108</td>
<td>con series</td>
</tr>
<tr>
<td>109</td>
<td>con series</td>
</tr>
<tr>
<td>150</td>
<td>47-7000</td>
</tr>
<tr>
<td>151</td>
<td>cisco0500</td>
</tr>
<tr>
<td>152</td>
<td>80-7066C</td>
</tr>
<tr>
<td>153</td>
<td>con5100</td>
</tr>
</tbody>
</table>

**Table 2.2: INCIDENT ($T_I$)**

<table>
<thead>
<tr>
<th>$incident_key$</th>
<th>$prod_key$</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>107</td>
</tr>
<tr>
<td>202</td>
<td>107</td>
</tr>
<tr>
<td>203</td>
<td>108</td>
</tr>
<tr>
<td>204</td>
<td>109</td>
</tr>
<tr>
<td>207</td>
<td>150</td>
</tr>
<tr>
<td>208</td>
<td>151</td>
</tr>
<tr>
<td>209</td>
<td>152</td>
</tr>
</tbody>
</table>

**Table 2.3: ORDER ($T_O$)**

<table>
<thead>
<tr>
<th>$order_key$</th>
<th>$incident_id$</th>
<th>$product_name$</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>201</td>
<td>AIR1212AC</td>
</tr>
<tr>
<td>302</td>
<td>201</td>
<td>AIR1002</td>
</tr>
<tr>
<td>303</td>
<td>203</td>
<td>con5122</td>
</tr>
<tr>
<td>304</td>
<td>204</td>
<td>mem-4700m-64d=</td>
</tr>
<tr>
<td>305</td>
<td>207</td>
<td>47-7066C</td>
</tr>
<tr>
<td>306</td>
<td>208</td>
<td>cisco0510</td>
</tr>
<tr>
<td>307</td>
<td>208</td>
<td>cs6012</td>
</tr>
</tbody>
</table>

**Record Matching:** Given two instances $e_1$ and $e_2$, a record matching function is defined as a 4-tuple: $\langle e_1, e_2, v, r \rangle$ where $e_1$ and $e_2$ are two field records; $v$ is a similarity
score (typically in the $[0, 1]$) between $e_1$ and $e_2$; $r$ is a relation (e.g., equivalence, part-of, etc.) between $e_1$ and $e_2$. The matching function $\langle e_1, e_2, v, r \rangle$ asserts that the relation $r$ holds between the record $e_1$ and $e_2$ with score $v$. For numerical field matching, only if two records are equal, they are considered matched. For example, records \{107, 108, 109, 150, 151, 152\} in `product_id` field in Table 2.1 are matched as the equal records in `prod_key` field in Table 2.2, respectively. For a non-numerical pair, however, if the similarity score of two records are higher than a threshold based on a similarity metric, they are considered matched. Here we consider the part-of relation in the network management databases for meaningful relations including subgroup versus group, product versus product family, subseries versus series, etc. For example, the record “con5122” in `product_name` field and the record “con5100” in `family` field can have high similarity score with part-of relation. A record pair which is matched is called a matched record pair, and it is called a non-matched record pair if the pair is not matched.

**Field Matching:** A field here means a database field indicating the names of a column and the single piece of data stored. Given two fields $f_1, f_2$ and a threshold $T$, we define $sim(f_1, f_2)$ as the matching/similarity score (e.g. Jaccard similarity [14]) between two fields $f_1$ and $f_2$. If $sim(f_1, f_2)$ value is above $T$, we call $(f_1, f_2)$ a matched field pair, otherwise it is called a non-matched field pair. In the toy example, $sim(product_id, prod_key)$ has a high matching score with Jaccard similarity, so $(product_id, prod_key)$ can be correlated and matched. Moreover, the field pair $(family, product_name)$ can also be matched in terms of many matched record pairs such as some pairs $(AIRseries, AIR1002)$, $(47 – 7000, 47 – 7066C)$ and $(con5100, con5122)$, etc.

**Graph Database:** One effective way to utilize matched results is to construct a graph database for semantic query, network analysis, network prediction, etc. [116, 107], which is also our goal. We define a graph database as a labeled, attributed and
undirected graph $G = (V, E, L_v, L_e)$ where $V$ is the node set containing all the records appearing in the fields which match, $E$ is the edge set between node pairs for node set $V$. $L_v$ is a set of label information of node set $V$, which are the index attributes for the columns of a table. $L_e$ is a set of label information of edge sets $E$, which are the relations of two records from two tables’ column attributes. Specifically, when we construct a graph database from matching of relational databases, a node $v$ consists of a field and a record value in a row; the label $L_v$ of $v$ comprises of the other field information in the same row as node $v$; an edge $e$ is the matching between two records; $l_e$ indicates the field information when two records of the fields matches. Figure 2.1 shows an example of a constructed graph database from parts of matched results in Tables 2.1, 2.2 and 2.3. For example, a specific node $v$ “product id: 107” in the Table 2.1 has a node label $l_v$ “PRODUCT” and “Family:AIR series”. Node $v$ matches with “prod_key: 107” in Table 2.2, so it is connected to the node “Incident_key:202” with a Product-incident relation, to the node “Incident_key: 201” with a Product-incident relation, and to the node “Family:AIR series” with a Product-family relation.

### 2.3 Matching Algorithm

To find whether two fields match, one simple way is to use field name matching. If the name of two fields are the same or similar, they are matched. However, this is not reliable for many database sources because they are noisy and irregular. Moreover, the network databases comprise of numerical and non-numerical fields with different attributes and matching requirements. For numerical field matching, we consider equivalence relation between record pairs. For non-numerical field matching, however, we do not directly consider the equivalence relation as the matching standard. non-numerical record values are possible to be semantically correlated with different names. For example, the non-numerical fields family and product name in the Table 2.1 and 2.3 have very few common characters on their names, but they
Figure 2.1: Graph database created from Table 2.1, 2.2 and 2.3.

are semantically correlated that product_name has a part-of relation with family. Moreover, in terms of field records, records “cisco0510” and “cisco0500” in these two fields can be considered belonging to the same family and being matched with a high similarity. However, the record pair “47-7066C” and “80-7066C” are considered to be non-matched with different families, even if the two strings have many common characters. (Details will be covered in 2.3.3). Hence, we use the record matching to decide whether two fields match to improve matching accuracy and satisfy semantic matching. Overall, we match numerical and non-numerical fields separately and design different matching algorithms respectively.

2.3.1 System Overview

The system overview is shown in Figure 2.2. We divide the structured data into numerical data with only numerical fields and non-numerical data with only non-
numerical fields. In each part, we develop an independent matching algorithm for field matching. Matching algorithms for numerical and non-numerical data are quite different, which will be introduced in section 2.3.2 and 2.3.3. The results of each part are combined together to load into a graph database.

![System diagram of automatic integration for network management databases.](image)

**Figure 2.2:** System diagram of automatic integration for network management databases.

### 2.3.2 Numerical Field Matching

Numerical fields are table fields with records which are numerical values. For example, the *incident_key* and *prod_key* in Table 2.2 are numerical fields. Their record values serve as a basis for similarity metrics of fields. We define each numerical field record values as a set. This is transferred to a problem of set similarity.

#### 2.3.2.1 Range Difference and Bucket Dot Product Similarity Metrics

There are some common methods for solving set similarity including Jaccard index, Dice index, Hamming distance, cosine similarity [20], etc. However, it is not practical to just use one method to get accurate decision bounds of matching because of the noisiness and complexity of the structured databases. We propose a synthetic
column-wise numerical field matching algorithm to get the decision bounds to determine whether two fields are matched or not. The numerical field algorithm is shown in Figure 2.3.

![Numerical field matching flow](image)

**Figure 2.3: Numerical field matching flow.**

The process of this algorithm is shown as follows:

- We preprocess the numerical data, such as removing null values, negative values and some exceptional non-numerical values in every field.

- We apply range difference similarity metric to all the preprocessed numerical field pairs between every two tables.

- After we get the range difference similarity score for each field pair, a threshold $T_r$ (which will be discussed later) will be decided to cut the filtered results.

- Finally, bucket dot product similarity metric will be applied to the range difference similarity metric’s filtered results. Then we sort the similarity scores of all the pairs to select the most correlated field pairs, that is, top-$k$ result as
matched pairs with scores above a certain threshold $T_b$ will be selected to input to a graph database.

2.3.2.1.1 Preprocessing To deal with the irregular and noisy data, we do some preprocessing of field record values before the formal similarity calculations. This includes removing null values, negative values, and considering unique values only. We do not consider negative values because they are useless and noisy data in the real databases. Almost all the table fields are about identifications, or numbers which can possibly be matched among them. In our databases, an average of 6% of field records are removed (excluding unique value reduction) which does not impact the instance-based algorithms.

2.3.2.1.2 Range Difference Similarity Metric Considering the noisy and sparse characteristics of data, Jaccard similarity [14] as a similarity metric, which measures how many common values between two sets, is not ideal for differentiating some matched pairs and non-matched pairs. For example, for a non-matched field pair with limited number of records, the number of common values in these two fields might take a large portion and hence the Jaccard similarity is very high for them. Their distribution of range, however, can be quite different, which is probably not to be matched in most cases. Therefore, we propose range difference (RD) similarity metric to measure the distribution of these field pairs first. Using RD similarity metric first, we can effectively prune lots of unwanted computations, which can also hugely reduce time consumption for further matching. Given a field set $A$, we sort the record value and then get the different percentile (10th, 20th, 30th,..., 90th percentile). The percentile $i$th value in $A$ is recorded as $A_i$. Therefore, given two field record sets, $A$ and $B$, the RD value $D_i$ for each percentile $i$ is given as

$$ D_i = \frac{|A_i - B_i|}{|A_i + B_i|} \quad (2.1) $$
We use 20th plus 30th percentile as the low range coverage, and 80th plus 90th percentile as the high range coverage, to cover the distribution of field record values. Hence the RD similarity score for a field pair \((A, B)\) is defined as:

\[
RDS(A, B) = 1 - \frac{D_{20} + D_{30} + D_{80} + D_{90}}{4}
\]  

(2.2)

To keep consistent with the general similarity metric and be convenient for comparisons, we use 1 minus the averaged RD value as the RD similarity score \(RDS\). The metric based on this similarity score is called RD similarity metric. Using this similarity metric, we can get the similar distribution for matched pairs. \(RDS\) value is in \([0, 1]\). The bigger the similarity score, the more correlated the pair. There are three cases about the similarity score here: (1) If there are no overlaps between two field ranges, \(RDS\) would be as low as the minimum value 0. (2) If two fields have similar distributions, \(RDS\) would be higher, up to 1. (3) If two field ranges overlap at the head, tail or in the middle, \(RDS\) can fall into a middle value.

In our toy example, the matched pairs \((incident\_key, incident\_id)\) and \((product\_id, prod\_key)\) have \(RDS\) values as high as 0.996 and 0.999 respectively. In contrast, the pairs \((incident\_key, product\_id)\) and \((prod\_key, incident\_id)\) have no overlaps with \(RDS\) value 0, which are not matched pairs. The more correlated the field pairs are, the higher \(RDS\) similarity score they have. Therefore, using a threshold \(T_r\) to filter results, we can almost rule out case (1) and part of case (3), then mainly consider case (2) to differentiate them further. To minimize the error of \(RDS\) similarity score in the first step, we can use a conservative threshold close to the boundary to only filter out definite non-matched pairs, which will be discussed in the section 2.4.

**2.3.2.1.3 Bucket Dot Product Similarity Metric** After we consider the distribution of field pairs with range difference similarity metric, we propose bucket dot product (BDP) similarity metric to further refine the filtered results of \(RDS\) similar-
ity metric. BDP similarity metric is to divide the whole concatenated ranges of two fields into different bucket/bins and compress each bucket as one point to calculate dot product similarity. The intuition behind this is that matched pairs generally have more common values than non-matched pairs. If we increase the bucket size up to a certain value to calculate dot product, it can make the similarities of all the non-matched pairs decrease more, and meanwhile make the similarities of all the matched pairs drop less, therefore it effectively increases the similarity gaps between matched pairs and non-matched pairs. Therefore, a good design of BDP will help significantly differentiate between matched pairs and non-matched pairs.

The general dot product similarity of two vectors \(X\) and \(Y\) with \(n\) elements is \(DP\), defined as follows: 
\[
DP(X, Y) = \sum_{i=1}^{n} X_i \cdot Y_i.
\]
We use the bucket number \(b_n\) to determine the number of buckets for calculating the dot product. Given two field record sets \(A\) and \(B\), we first derive the required vectors \(A_v\) and \(B_v\) for the input to the BDP similarity calculation. The vectors \(A_v\) and \(B_v\) derived from \(A\) and \(B\) are constructed in this way. Given two sets \(A\) and \(B\), we concatenate \(A\) and \(B\)’s value ranges as a combined set \(C\), and then divide \(C\) into several buckets according to the \(b_n\). If there is any one value in \(A\) or \(B\) falling in a bucket, the bucket point for \(A_v\) or \(B_v\)’ is 1, otherwise it is 0. Then we apply the general dot product similarity to \(A_v\) and \(B_v\). Therefore, the BDP similarity score (normalized) is defined as follows:

\[
BDPS(A_v, B_v) = \frac{\sum_{i=1}^{b_n} A_{vi} B_{vi}}{|A_v||B_v|} \tag{2.3}
\]

where \(b_n\) decides the sparsity/density of range distributions. Since sets \(A\) and \(B\) usually have different sizes with different ranges, it would make sense for \(b_n\) the same for each set. For example, we calculate the BDPS for a field pair \((\text{incident key}, \text{incident id})\) in Table 2.2 and 2.3 as \(A\) and \(B\) with \(b_n = 3\). We first concatenate these two field ranges into a set \{201, 202, 203, 204, 207, 208, 209\}. Then we construct a set \(C = \{\{201, 202, 203\}, \{204, 207, 208\}, \{209\}\}\) according to the bucket number \(b_n\). After
that, we get the vector \( A_v = \{1, 1, 1\} \) and \( B_v = \{1, 1, 0\} \). Finally, the BDPS is 0.816, which is high for matching. If we set \( b_n \) as 4, BDPS for this pair is 1, which is the highest for matching.

\( b_n \) is also an important factor to affect the quality of this metrics. According to our experimental observations, it is affected by the data range and distribution. Generally, matched pairs would have more similar ranges than non-matched pairs. A trade-off value of \( b_n \) would effectively improve matched pairs’ similarities more and also not help grow non-matched pairs’ similarities much, which can potentially increase more gaps between matched and non-matched pairs. The selection of \( b_n \) will be discussed later in the section 2.4.

### 2.3.2.1.4 Final Top-k Selection
To construct a high quality graph database, more true positive field pairs are preferable from higher similarity scores. Also, manual thresholds could lead to selection instability of field pairs with low similarity scores, so we seek top-k to further refine the quality of matching for the graph database. We sort all the candidate pairs by the similarity scores in a non-ascending order, then we verify this final field pair matching results to select top-k field pairs, which involves only a little human labor.

### 2.3.2.2 SVM Classification-based Matching

Our previous proposed similarity metric-based algorithm for numerical field matching involves three manual thresholds to determine field matching. To overcome the problem of selecting thresholds manually, we propose a classification-based learning approach to decide matching or non-matching for numerical fields here. The target label is whether a field pair matched or not matched. Hence, the matching could be modeled as a binary classification problem. Using a classification model and previously proposed similarity scores as features, the model can learn the internal thresholds and decide a given field pair is matched or not. We select support vector machine
(SVM) as our binary classification for the reason that it works well on unstructured data and scales well with high dimensional data.

**Features for Classification:** In the numerical field matching, we have previously generated RD similarity score (RDS) and BDP similarity score (BDPS). We propose to use these similarity scores as features. To generalize our model, we generate 19 different BDPS based on different bucket numbers with 19 percentiles [5th, 10th, 15th,..., 85th, 90th, 95th] of combined records from each ground truth field pair. As a consequence, we have 20 features in total for classification. With these features and data, the thresholds in previous similarity metric-based numerical field matching algorithm involved in deciding the boundary decision of RD or BDP, and bucket number $b_n$ could be internally learned through our SVM model.

### 2.3.3 Non-numerical Field Matching

Here we proposed algorithms for non-numerical field matching, including top priority match metric, minHash-locality sensitive matching, and SVM-based classification.

#### 2.3.3.1 Top Priority Match Metric

We propose an algorithm for non-numerical field matching–top priority match metric (TPM) for fast filtering, and match ratio score for final similarity computations. The diagram is shown in Figure 2.4. The main process of this algorithm is as follows:

- **Preprocess non-numerical data:** this is an important step that decides the quality of our non-numerical field matching algorithm. After splitting non-numerical data from the original databases, we use our designed natural language processing methods of segmentation, stemming and prefixing for every field record.

- **Calculate the record-wise similarities iteratively:** It is very time-consuming to apply cosine similarity to the combination of every record pair in a field pair.
Considering the scalability of large-scale data matching, we propose top priority match metric for record-wise similarity calculation. In the process of iterative computations, we check the termination condition to terminate the iterations earlier, which significantly reduces time complexity.

- Calculate match ratio score: after record-wise similarity computations for every field pair are finished, we calculate the defined matching ratio score for each field pair.

- Select top-$k$ results: we sort the field pairs by the matching ratio score in a non-ascending order, and top-$k$ field pairs are selected as the final results for a graph database.

Here, we discuss each proposed steps in detail.

2.3.3.1.1 **Preprocessing**  Non-numerical field matching considers partial match between two strings. For example, product “cisco0510” and product “cisco0500” are in the same series, which is considered as a partial match. Hence, we propose
the following preprocessing method. (1) Parse every record string $A$, remove null value, separate alphabetic and numerical characters into different new substrings, and tokenize the string words. (2) Stem the alphabetic strings of the original record and new substrings. (3) Reserve the prefixes with a certain length of the original numerical strings and the new substrings if they are digital substrings. The prefix length is 2 here according to our experiments.

Each record string is preprocessed in those three steps above. For example, we have an original field record $R$ 

\{'mem-4700m-64d='\} in Table 2.3. We can obtain a string collection $X$ \{'4700', '64', '4700m', 'd', 'm', 'mem 4700m 64d', '64d', 'mem', '47xx'\} after preprocessing $R$.

2.3.3.1.2 Top Priority Match Metric for Record-wise Similarity

One intuitive way is to preprocess all the combination of record pair comparisons and calculate the similarity of each record-wise pairs. That would be very time-consuming or even unfeasible when the data are large. Specially, if two fields are not correlated as a matched pair, it would be costly for useless computations. Therefore, we propose a fast record-wise matching algorithm called top priority match (TPM) metric for record-wise similarity to fulfill this. Intuitively, if two fields $A$ and $B$ are correlated, there will be a high percentage of record-wise pairs that have higher similarities. The probability of a matched record pair encountered is higher than non-matched record pairs. Therefore, we first sort all the preprocessed records in each two fields $A$ and $B$, then we compute how many of records in $A$ are matched with records in $B$ from top to bottom, and vice-versa. The comparisons can hence be terminated as long as the current record pair similarity achieves below the similarity threshold $T_{rn}$ we set for deciding the matching of a record pair, which greatly reduce the times of comparisons with combinations.
2.3.3.1.3 Record Pair Similarity  In this fast record-wise comparisons, the record pair similarity used is cosine similarity between two record collections after preprocessing two records. It decides how and when to reduce the comparisons of matching in a fast and effective way. A threshold $T_{rn}$ is to decide how similar a record pair is as a matched record pair, which can also be adjusted by users.

Given a preprocessed string collection $X$ and another preprocessed string collection $Y$, we remove duplicated elements and transfer them into a set $XY$ ($X$ union $Y$). Next we generate a binary vector $V_x$ and a binary vector $V_y$ according to the value distribution of $X$ and $Y$ in $XY$, then we calculate the cosine similarity $\text{sim}(V_x, V_y)$ between $V_x$ and $V_y$ by getting their dot product divided by their magnitude multiplication.

$$\text{sim}(V_x, V_y) = \frac{V_x \cdot V_y}{|V_x||V_y|} \tag{2.4}$$

For example, we have a preprocessed string collection $X \{\text{cisco}, 0510, \text{cisco0510}, 05xx\}$, and $Y \{\text{cisco}, 05xx, 0500, \text{cisco0500}\}$, we transfer them into a set of $X$ union $Y$, $XY \{\text{cisco}, 0510, \text{cisco0510}, 05xx, 0500, \text{cisco0500}\}$. Then the binary vectors generated according to $X$, $Y$ and $XY$ are $V_x \{1, 1, 1, 0, 0\}$ and $V_y \{1, 0, 0, 1, 1, 1\}$. Finally, we calculate the cosine similarity of $V_x$ and $V_y$ as the similarity of $X$ and $Y$, that is, $\text{sim}(X, Y) = (1+1)/(2*2) = 0.5$.

2.3.3.1.4 Matching Ratio Score  Matching ratio score is proposed to calculate the final similarity for a field pair. After we have gone through the reducing comparisons for record-wise similarities, we select the number of record pairs that have similarity scores above $T_{rn}$. A matching ratio score as a final field pair similarity is the average of ratios of top matched record pairs calculated as follows:

Given two non-numerical sets $A$ and $B$, there are $m$ items $\{a_1, a_2, ..., a_m\}$ in $A$ and $n$ items $\{b_1, b_2, ..., b_n\}$ in $B$. The matching ratio score (MRS) between $A$ and $B$ is defined as follows.
\[ MRS(A, B) = \frac{1}{2} * \left( \frac{\sum_{i=1}^{m} A_i}{m} + \frac{\sum_{j=1}^{n} B_j}{n} \right) \] (2.5)

where

\[ A_i = \begin{cases} 
1 & \text{if } \exists b_j \in B, \ sim(a_i, b_j) \geq T_{rn} \\
0 & \text{otherwise}
\end{cases} \] (2.6)

and

\[ B_j = \begin{cases} 
1 & \text{if } \exists a_i \in A, \ sim(b_j, a_i) \geq T_{rn} \\
0 & \text{otherwise}
\end{cases} \] (2.7)

where \( sim(a_i, b_j) \) and \( sim(b_j, a_i) \) are the cosine similarities of the record pairs \((a_i, b_j)\) and \((b_j, a_i)\), respectively. MR value is in \([0, 1]\) and it is the final similarity score to decide the correlation of each field pair.

### 2.3.3.1.5 Final Top-\( k \) Results

Similar to numerical field matching, matched non-numerical field pairs in the result list are more meaningful and important than non-matched field pairs, so we select top-\( k \) results of non-numerical field pairs sorted with MRS in non-ascending order for a graph database. \( K \) value can be selected by users for deciding most effective field pairs in a graph database and limiting the size of the graph database.

### 2.3.3.2 MinHash- Locality Sensitive Matching

The proposed TPM metric can be effective to distinguish between matched and non-matched non-numerical fields. However, it possibly involves all the pairwise record combinations in the worst time complexity, which is time-consuming for large databases with millions of records. Therefore, we propose applying more scalable minHash-locality sensitive hashing algorithm (MH-LSH) \cite{96} to estimate the matching score of non-numerical fields in the databases. It can greatly reduce the comparison size and time for non-numerical record-wise pairs with little cost of matching accuracy down.
The proposed diagram for non-numerical field matching based on MH-LSH is shown in Figure 2.5. The main process of the algorithm is as follows:

- Preprocess non-numerical data: this step is the same as the preprocessing step of TPM algorithm.

- Select matched field pairs fast: we apply the locality sensitive hashing technique in the database field matching for fast selecting field pairs that are correlated.

- Field pair similarity calculation: We use minHash technique to estimate the matching score of field pairs.

- Select top-\( k \) results: Same as the operation for non-numerical field matching based on TPM, we sort the field pairs by the estimated matching score in a non-ascending order, the top-\( k \) field pairs are selected as the final results of field pairs for a graph database.

Figure 2.5: Non-numerical field matching flow based on MH-LSH.

Here, we discuss important steps in detail.
### 2.3.3.2.1 Matching Score Estimation with MinHash

The similarity score of matched field pairs can be estimated fast with minHash signatures. Given two fields \( A \) and \( B \), we can evaluate the similarity between them as follows. We choose \( n \) hash functions \( h_1, h_2, ..., h_n \). For each hash function \( h_j \), we let a signature of set \( A \) be \( sgn(A) = \min_{i; a_i \in A} h_j(a_i) \) for \( j \in n \). Let a signature of set \( B \) be \( sgn(B) = \min_{i; b_i \in B} h_j(b_i) \) for \( j \in n \). Then, the probability that the two sets have the same minHash signatures is used to estimate their similarity.

\[
\text{MHSim}(A, B) = P\{sgn(A) = sgn(B)\} \tag{2.8}
\]

Here each record in a field \( A \) or \( B \) is also preprocessed with the same preprocessing method of TPM algorithm. The preprocessed record strings are combined into a new set like a word set in a document. We also use \( k \) shingle (a substring of length \( k \)) to create a set of \( k \)-shingles strings and apply \( n \) different hash functions on the set of strings.

### 2.3.3.2.2 Field Pair Selection with Locality Sensitive Hashing

Performing pairwise similarity measurement can be time consuming with large amounts of field pairs available. In order to identify which field pairs are similar quickly, we propose using locality sensitive hashing (LSH) to select the candidate field pairs.

The values of minHash signature \( sgn(A) \) for one field \( A \) are grouped into \( b \)-tuples (referred to as sketches) with \( r \) rows. Similar field pairs have similar minHash signatures and hence have a high probability of having the same sketches. Moreover, dissimilar pairs have low chance of falling into the same sketch. The probability that two fields of \( A \) and \( B \) have at least one sketch (of size \( b \)) in common out of \( r \) is

\[
P_C(A, B) = 1 - (1 - \text{MHSim}(A, B)^r)^b \tag{2.9}
\]
\[ P_C(A, B) = 1 - (1 - \text{sim}(A, B))^r \] 

(2.10)

Therefore, we can find the candidate pairs with the designed number \( b \) and \( r \). The selection of \( b \) and \( r \) is generally decided by a threshold \( t = (1/b)^{1/r} \) shown in [96], which indicates how similar the two fields is to be considered as a candidate pair, and can also be set by users. In this way, if pairs with similarity above \( P_C(A, B) \), they will be selected as candidate pairs to be further estimated, and the matching score between them with minHash will be calculated.

### 2.3.3.3 SVM Classification-based Matching

Similarly as numerical field matching, we propose a SVM classification-based matching method to avoid the manual thresholds involved in the top-priority match (TPM) or minHash-locality sensitive (MH-LSH) metrics for non-numerical matching.

**Features for Classification:** During the calculation of TPM matching ratio score (MRS) and MH-LSH score (MHSim), we have one important record-pair threshold \( T_{rn} \) while calculating TPM matching ratio score. To avoid the record similarity threshold \( T_{rn} \), we use a broad range of 7 different \( T_{rn} \) values in [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8] to calculate different matching ratio scores as features. Combined with MHSim, 8 features of scores are obtained for each non-numerical field pair. With these features and data, the thresholds \( T_{rn} \) and the final threshold for deciding the matching boundary are internally learned through our SVM classifier.

### 2.4 Experimental Evaluation

We evaluate our technique NEMA for structured network management database matching. Specifically, we measure the effectiveness of NEMA using ground truth for numerical and non-numerical data that are annotated by humans. Meanwhile, experiments on a large dataset are also conducted, and we show the top-k effective
results of matching field pairs. Moreover, comparisons of NEMA with other existing algorithms are also shown.

2.4.1 Dataset

The structured network management databases available in the form of database tables are provided by Cisco Systems, Inc. The dataset includes heterogeneous and diversely distributed “install_base” and “service_request” databases, which are generated by various departments of the corporation.

In these databases, there are 21 tables which contain 1,458 columns. Each column has 10 million records on the average. Out of them, there are 679 numerical fields and 779 non-numerical fields. Therefore, a complete match involves the maximum 1,067,882 field matching decisions. With primary key constrains in numerical field matching, there are 5 “primary keys” on average in each table, which would reduce to 374,326 field pairs matching.

We have ground truth field pairs that are annotated by humans to be matched or non-matched field pairs for a subset of the data. There are 60 balanced ground truth field pairs in numerical dataset and 40 balanced ground truth field pairs in non-numerical dataset. For future reference, table names in service_request database start with “T.”, and start with “X.” in install_base database, respectively.

2.4.2 Experimental Setup

We implement NEMA system in Python. To evaluate the effectiveness of NEMA, we evaluate the numerical and non-numerical algorithm parts, respectively. For each part, we first evaluate its algorithms based on the ground truth data. Then all the column pairs in the large dataset are evaluated in the following experiments shown in sections 2.4.3.2 and 2.4.4.2, which shows the effectiveness of NEMA. Then we evaluate the SVM classification-based algorithms. Finally, we compare with the common matching system COMA [10] on the ground truth for both schema-level
and instance-level matching. Our quality evaluation is based on the balanced ground truth of positive and negative field pairs. We use common metrics—precision, recall, “accuracy” (ACC) and “F1” score (F1) to evaluate our field matching algorithms.

2.4.3 Evaluation based on Numerical Data

We evaluate our technique NEMA on the numerical data in two parts. Because of the limited ground truth field pairs, other than the common 80/20 splitting ratio, we randomly select 60% of matched field pairs and 60% of non-matched field pairs to make balanced ground truth from the whole ground truth field pairs to determine the thresholds of NEMA numerical algorithms. The rest 40% of them will be tested to show the quality in Section 2.4.6. The more proportion for test data also helps reduce the randomness and improve the generalization of our algorithms. The matching results of all of other numerical field pairs are also described in Section 2.4.3.2.

2.4.3.1 Evaluating of Ground Truth

We show the evaluation result of NEMA numerical algorithms and the compared baseline method-Jaccard similarity using numerical ground truth (We choose Jaccard similarity since it is an exemplar method considering common values of two sets for similarity calculation). In the dataset, there are 30 matched ground truth field pairs which are originally from fields pairs annotated by humans or from join operations in the databases and proved to be matched field pairs. Referring to the undersampling technique, we randomly sample 300 non-matched field pairs confirmed by humans and select 30 non-matched field pairs from them as non-matched ground truth to construct balanced data instances of positive and negative field pairs.

Table 2.4 shows 10 matched and 10 non-matched field pair examples of numerical ground truth. Columns “Table.field A” or “Table.field B” shows a table name and its field pair name to be compared. The matched class column indicates that the field pair is matched with value “1” or non-matched with value “0”. For example,
Table 2.4: Example of numerical ground truth field pairs

<table>
<thead>
<tr>
<th>No.</th>
<th>Table.fieldA</th>
<th>Table.fieldB</th>
<th>Matched class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T_INCI.prod.hw_key</td>
<td>T_HW.PROD.bl_prod_key</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>T_INCI.cur.ct_key</td>
<td>T_CT.bl_ct_key</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>T_INCI.up.tech_key</td>
<td>T_TECH.bl_tech_key</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>T_INCI.inci_id</td>
<td>T_INCI_I2.inci_id</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>T_INCI.bl_cot_key</td>
<td>T_COT.bl_cot_key</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>T_INCI.inci_id</td>
<td>T_OR_HD.inci_id</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>T_INCI.item_id</td>
<td>T_PROD.item_id</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>T_INCI.ins_site_key</td>
<td>T_SITE.bl_site_key</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>T_OR_LN.prod_key</td>
<td>T_PROD.bl_prod_key</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>T.OR_HD.header_id</td>
<td>T.OR_LN.header_id</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>T.OR_HD.order_dur</td>
<td>T.OR_LN.loc_key</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>T_CAL.bl_cal_key</td>
<td>T_PROD.item_id</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>T_INCI_I2</td>
<td>T.DEFT.deft_id</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>T_INCI_I2.res_time</td>
<td>T_TECH.sub_tech_id</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>X_PRO.list_price</td>
<td>T_TECH.sub_tech_id</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>T_INCI_I2.res_time</td>
<td>T_TECH.bl_tech_key</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>T.OR_HD.deliv_dur</td>
<td>T.DEFT.deft_key</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>T.OR_LN.hold_dur</td>
<td>T_TECH.sub_tech_id</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>T_IN.serlevel_key</td>
<td>T_INCI.last_dur</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>T_IN.closect_key</td>
<td>T_INCI.resp_tz</td>
<td>0</td>
</tr>
</tbody>
</table>

in the first row, “T_INCI.prod.hw_key” indicates a field “prod.hw_key” in the table “T_INCI”, and “T_HW.PROD.bl_prod_key” indicates a field “bl_prod_key” in the table “T_HW.PROD”. This field pair about products’ key is matched, indicated with “1” in the matched class value. The remaining rows share the same characteristics too.

The problem of similarity of a numerical field pair is modeled as similarity problem of a set pair. The baseline method for the similarity is the well-known Jaccard similarity which measures the similarity of two given sets.

Figure 2.6 shows the Jaccard similarity scores on these ground truth field pairs. Red circle represents matched pairs and blue star represents non-matched pairs. X axis denotes the index of these 30 matched and 30 non-matched field pairs, and Y axis indicates Jaccard similarity score. From this figure we can see that there are about half of positive and negative pairs mixed together from which are difficult to differentiate.
Figure 2.6: Matching with Jaccard similarity metric.

Figure 2.7: Matching with combined RDS and BDPS \(^1\).
Figure 2.7 shows our numerical field matching algorithm. It shows the applied result on 18 positive and 18 negative field pairs with combined RD similarity and BDP similarity metrics. For RD similarity metric, we use a threshold $T_r$ and rule out the result pairs below $T_r$ as non-matched field pairs. Then we keep the rest of field pairs to BDP similarity metric for further matching. We show this combined RDS and BDPS together for better visualizing the decision boundary of matching. The decision threshold for RDS is $T_r = 0.1$. When $0 < T_r < 0.1$, y-axis shows the RDS. When $T_r \geq 0.1$, it shows the normalized BDPS for the rest of field pairs. There are 12 field pairs which are considered as non-matched pairs and removed with $0 < RDS < 0.1$. The rest 24 field pairs (18 matched pairs and 6 non-matched pairs) with $RDS \geq 0.1$ are easily differentiated with BDP similarity metric. We can see that the BDP result could provide an excellent decision boundary among matched pairs and non-matched pairs in which the final threshold $T_b$ is chosen around 0.2. With this combined RD and BDP similarity metrics, we have greatly improved the result over the Jaccard similarity result.

The bucket number $b_n$ is the main factor affecting the BDP similarity and the final results. To determine the optimal $b_n$ value, we use a number of different $b_n$ values to apply on that 60% of ground truth field pairs to evaluate on the accuracies. The $b_n$ values are obtained from different percentiles (5th, 10th, ..., 90th and 95th percentile) of record values of each combined field pair when computing BDPS. Figure 2.8 shows the accuracy of BDP similarity metric with different percentiles ($b_n$ values) when the threshold $T_b$ is set at 0.1 and 0.2. It shows a similar summit that the accuracy is at an optimal value when the $b_n$ is around from 35th percentile. BDP accuracy goes down when the $b_n$ becomes smaller or bigger. Thus the $b_n$ is selected as 35th percentile for matching other part of our dataset in the later experiments.

BDPS values are obtained from the rest of 24 field pairs after RD similarity metrics is applied with a threshold $Tr = 0.1$ and normalized in $[0, 1]$. 

1
2.4.3.2 Top-20 Similarity Results

The matching experiment based on all the 679 numerical fields is shown here. Table 2.5 shows top-20 matching results. We use RDS threshold $T_r = 0.1$ and bucket number $b_n = 50,000$ for BDP similarity computation. All the rows are accurate matches, which are also confirmed by humans. The accuracy can be up to 100\% for the top-20 results, which shows a great potential for our numerical field matching algorithm applied on the large dataset, also reducing lots of human labor for matching.

Moreover, with our system-aided matching findings, we can find some pairs matching which are difficult to be found with human annotations such as “changewg.key” and “subregion.key”, indicating which regions that the workgroup mainly serves.

To ensure a graph database more meaningful and complete, the selection of top-$k$ results is finally decided by users. Users can observe the top-$k$ results and rule out the unwanted matches as the final input pairs to a graph database so as to introduce less “noisy” connections of the graph database.
Table 2.5: Top-20 similarity result of numerical field pairs

<table>
<thead>
<tr>
<th>Table.field A</th>
<th>Table.field B</th>
<th>BDP similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>T.DEFT.deft_key</td>
<td>T.INCI.DE.bl_def_key</td>
<td>0.844</td>
</tr>
<tr>
<td>T.OR.LN.item_id</td>
<td>X.PRO.item_id</td>
<td>0.698</td>
</tr>
<tr>
<td>T.DEFT.def_id</td>
<td>T.INCI.DE.defect_id</td>
<td>0.682</td>
</tr>
<tr>
<td>T.INCI.item_id</td>
<td>X.PRO.item_id</td>
<td>0.673</td>
</tr>
<tr>
<td>T.DEFT.def_key</td>
<td>T.PROD.item_id</td>
<td>0.64</td>
</tr>
<tr>
<td>X.INS.item_id</td>
<td>X.PRO.item_id</td>
<td>0.597</td>
</tr>
<tr>
<td>T.PROD.bl_prod_key</td>
<td>T_SUR.task_key</td>
<td>0.567</td>
</tr>
<tr>
<td>T.INCI.changewg_key</td>
<td>T.WK.subregion_key</td>
<td>0.551</td>
</tr>
<tr>
<td>T.INCI.changewg_key</td>
<td>T.WK.theater_key</td>
<td>0.551</td>
</tr>
<tr>
<td>T.INCI.changewg_key</td>
<td>T.WK.wkgrp_key</td>
<td>0.545</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.WK.subregion_key</td>
<td>0.545</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.WK.theater_key</td>
<td>0.545</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.WK.wkgrp_key</td>
<td>0.507</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.PROD.bl_prod_key</td>
<td>0.507</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.WK.subregion_key</td>
<td>0.507</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.WK.wkgrp_key</td>
<td>0.507</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.PROD.bl_prod_key</td>
<td>0.507</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.WK.wkgrp_key</td>
<td>0.507</td>
</tr>
<tr>
<td>T.INCI.currentwg_key</td>
<td>T.WK.wkgrp_key</td>
<td>0.507</td>
</tr>
</tbody>
</table>
| 2.4.4 Evaluation based on Non-numerical Data

We evaluate our NEMA non-numerical algorithms based on TPM and Hashing on the non-numerical data in two parts as well. We use non-numerical ground truth data to evaluate the effectiveness of our algorithms. The matching results of all the other non-numerical field pairs are then described.

2.4.4.1 Evaluating of Ground Truth

There are 20 positive ground truth field pairs which are annotated by humans. Also, 20 negative ground truth field pairs are randomly selected from the dataset and verified to make balanced data instances of positive and negative field pairs. Similarly to numerical evaluation, we randomly select 60% matched field pairs and 60% non-matched field pairs to make balanced ground truth from the whole ground truth field pairs to determine the thresholds of NEMA non-numerical algorithms. The rest 40% will be tested to show the quality in Section IV-F. Part of field pair examples are shown in table 2.6.
Table 2.6: Examples of non-numerical ground truth

<table>
<thead>
<tr>
<th>No.</th>
<th>Table.field A</th>
<th>Table.field B</th>
<th>Matching class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T.PROD.item_name</td>
<td>X.INS.item_name</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>T.COT.cpr.country</td>
<td>T.SITE.country</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>T.CT.temp.desc</td>
<td>T.PROD.item_desc</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>T.CT.ctserv_line</td>
<td>X.SAH.servline_name</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>T.INCL.curr_wg_name</td>
<td>T.WK.wkgp_name</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>T.CT.temp.desc</td>
<td>X.SAH.temp_name</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>T.SITE.cust_state</td>
<td>X.SAH.billto_state</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>T.CT.temp_name</td>
<td>X.SAH.temp_desc</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>T.PROD.prod_family</td>
<td>T_HW.PROD.family</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>T.PROD.prod_family</td>
<td>T_HW.PROD.erp_family</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>T.INCL.init_gp_name</td>
<td>T.SITE.address</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>T.DEFT.deft_submitter</td>
<td>T.SITE.email_addr</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>T.COT.cpr_country</td>
<td>T.INCL.summary</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>T.SITE.address1</td>
<td>X.PRO.prod_family</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>T.PROD.prod_family</td>
<td>X.SAH.hdrcust_name</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>T.INCL.tacpica_ct</td>
<td>T_HW.PROD.family</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>T.WK.wkgp_desc</td>
<td>X.PRO.physins_loc</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>T.SITE.county</td>
<td>T_SUR.batchcot_name</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>T.INCL.customersw_ver</td>
<td>T.SITE.state</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>T.OR.LN.partsloc_code</td>
<td>X.INS.item_name</td>
<td>0</td>
</tr>
</tbody>
</table>

We first analyze the ground truth record-pairs and show the viability for the record pair similarity threshold $T_{rn}$. Table 2.7 shows the record pair similarity scores of 9 different record pairs in a field pair (“T.PROD.prod_subgrp”, “T_HW.PROD.platform”). The first 7 rows of pairs with high similarity scores are matched record pairs. The last 2 rows are not matched record pairs with lower score of 0.333. They have a decision boundary of score around 0.4. Also, based on the database matching standards of prefixing and our experimental observations on ground truth field records, we set $T_{rn} = 0.4$ as the record similarity threshold.

Table 2.7: Sample of non-numerical record pairs

<table>
<thead>
<tr>
<th>T.PROD.prod_subgrp</th>
<th>T_HW.PROD.platform</th>
<th>Record similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>c900 series</td>
<td>c900 series</td>
<td>1</td>
</tr>
<tr>
<td>c2950 series</td>
<td>c2916 series</td>
<td>0.8</td>
</tr>
<tr>
<td>1601r series</td>
<td>1601 series</td>
<td>0.775</td>
</tr>
<tr>
<td>css2950</td>
<td>css2916</td>
<td>0.667</td>
</tr>
<tr>
<td>C2960</td>
<td>C2960CX</td>
<td>0.577</td>
</tr>
<tr>
<td>C3560CX</td>
<td>C3560X</td>
<td>0.5</td>
</tr>
<tr>
<td>AIR35CE</td>
<td>AIR35SE</td>
<td>0.4</td>
</tr>
<tr>
<td>ts900</td>
<td>cs900</td>
<td>0.333</td>
</tr>
<tr>
<td>c800</td>
<td>s800</td>
<td>0.333</td>
</tr>
</tbody>
</table>

43
We demonstrate the effectiveness of non-numerical algorithms of TPM and MH-LSH by calculating matching ratio score based on TPM and estimated matching score based on MH-LSH on non-numerical ground truth. Figure 2.9 shows matching ratio scores of this ground truth data matching in a non-ascending order on TPM. The
matching ratio scores of almost all the matched pairs are above the non-matched pairs’s. If we use the threshold 0.1 or select top-20 results from this, the accuracy can achieve about 95%, which shows the effectiveness of NEMA based on TPM. Figure 2.10 shows the matching scores of these ground truth in a non-ascending order based on MH-LSH. Although the decision boundary is not as good as the TPM-based result, the accuracy can achieve 90% when the threshold is 0.1 or top 19 results are selected.

2.4.4.2 Top-20 Similarity Results

There are 779 non-numerical fields in the large dataset. Considering that almost all the primary keys in a table are numerical fields, we do not consider primary key constraint matching method for non-numerical field matching. The record similarity threshold $T_{rn}$ is set to be 0.4 here based on the analysis of Table 2.7. The final top list of matching ratio scores are obtained based on TPM algorithm from all the non-numerical field pairs.

Table 2.8 shows the top-20 results of field pair matching based on TPM. We can see that all the field pairs are matched pairs, and they are also confirmed by humans later, which shows the effectiveness of NEMA based on TPM algorithm.

2.4.5 Evaluation of SVM Classification-based Matching

2.4.5.1 Extending Ground Truth Data

To address the problem of limited availability of the positive ground truth in our dataset, we refer to a sampling method by H. Kohler [88] to synthesize more positive ground truth field pairs as “synthetically positive field pairs”. It preserves the “synchronization property” (to preserve the Jaccard similarity of original sets). Given an original ground truth field pair $(A, B)$, we sample a new field pair based on this original pair. The sampling process is to make sure if a particular record sample $e$ is sampled in $A$ and $e$ is also in $B$, then it is also sampled from $B$. If the original ground truth pair is positive, the synthetic field pair is considered as positive
Table 2.8: Top-20 matching results of non-numerical field pairs

<table>
<thead>
<tr>
<th>Table.field1</th>
<th>Table.field2</th>
<th>Matching ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_INCI.I2.currentwg_key</td>
<td>T_WK.wkgp_name</td>
<td>0.637</td>
</tr>
<tr>
<td>T_INCI.I2.currentwg_key</td>
<td>T_WK.wkgp_desc</td>
<td>0.632</td>
</tr>
<tr>
<td>T_INCI.initwg_name</td>
<td>T_WK.wkgp_name</td>
<td>0.63</td>
</tr>
<tr>
<td>T_INCI.initwg_name</td>
<td>T_WK.wkgp_desc</td>
<td>0.628</td>
</tr>
<tr>
<td>T_WK.wkgmgr_email</td>
<td>T_SUR.eval_email</td>
<td>0.626</td>
</tr>
<tr>
<td>T_INCI.creatorwg_name</td>
<td>T_WK.wkgp_name</td>
<td>0.624</td>
</tr>
<tr>
<td>T_INCI.creatorwg_name</td>
<td>T_WK.wkgp_desc</td>
<td>0.622</td>
</tr>
<tr>
<td>T_INCI.curr wg_name</td>
<td>T_WK.wkgp_name</td>
<td>0.611</td>
</tr>
<tr>
<td>T_INCI.curr wg_name</td>
<td>T_WK.wkgp_desc</td>
<td>0.607</td>
</tr>
<tr>
<td>X_SAH.billto_state</td>
<td>T_SITE.state</td>
<td>0.504</td>
</tr>
<tr>
<td>X_SAH.billto_state</td>
<td>T_SITE.cust_state</td>
<td>0.499</td>
</tr>
<tr>
<td>T_INCI.initwg_name</td>
<td>T_INCI.I2.curr wg_name</td>
<td>0.462</td>
</tr>
<tr>
<td>T_INCI.initwg_name</td>
<td>T_INCI.I2.wkgp_name</td>
<td>0.457</td>
</tr>
<tr>
<td>T_COT.cpr_country</td>
<td>T_SITE.country</td>
<td>0.453</td>
</tr>
<tr>
<td>T_SITE.cust_country</td>
<td>T_COT.cpr_country</td>
<td>0.453</td>
</tr>
<tr>
<td>T_INCI.I2.wkgp_name</td>
<td>T_INCI.curr wg_name</td>
<td>0.451</td>
</tr>
<tr>
<td>T_COT.cpr_country</td>
<td>T_SITE.cust_country</td>
<td>0.447</td>
</tr>
<tr>
<td>T_INCI.curr wg_name</td>
<td>T_INCI.I2.curr wg_name</td>
<td>0.442</td>
</tr>
<tr>
<td>T_SITE.country</td>
<td>T_COT.cpr_country</td>
<td>0.44</td>
</tr>
<tr>
<td>T_HW_PROD.erpplatform</td>
<td>X_SCDC.products_sub_grp</td>
<td>0.431</td>
</tr>
</tbody>
</table>

as well. We synthesize \( x \) (e.g. 100) more field pairs out of each ground truth field pair. Here 20% of records values from each field are sampled. In the original dataset there are 30 positive ground truth field pairs in numerical matching. For each one of the ground truth field pairs, we synthesize 100 field pairs out of it, thus 3,000 synthetically positive field pairs are created. Also, we randomly select 3,000 negative ground truth field pairs from the original dataset. Therefore, we create balanced data instances of 6,000 field pairs where 3,000 positive field pairs and 3,000 negative field pairs for classification. Similarly, for non-numerical fields, we create data instances of 6,000 field pairs where 3,000 synthetically positive field pairs from the 20 positive field pairs and randomly select 3,000 negative ground truth field pairs as the classification dataset.

We show some comparisons of original ground truth dataset and synthetic dataset here. There are 30 positive ground truth field pairs from original dataset for numerical fields and 20 positive ground truth field pairs for non-numerical fields. For each field pair we compare with 3 randomly selected synthetic field pairs labeled as S1, S2, and S3 respectively. Figure 2.11 shows the comparison of range difference score (RDS)
for these synthetic dataset with original dataset in numerical field matching. Figure 2.12 shows the comparison of these matching ratio score (MRS) for these synthetic dataset with original dataset in non-numerical field matching. Both results show that synthetic field pairs have similar scores with original field pairs. It further implies that synthetic dataset has similar range and distribution with the original dataset.

2.4.5.2 SVM Classification Results

Here we show the experimental results of numerical and non-numerical field matching with cross-validation and testing. For each classification, we randomly split our large synthetic dataset into 80% as training dataset, and 20% as test dataset according to the common splitting ratio based on the Pareto principle [158]. In the training stage, 5-fold cross-validation is used for validation. We run the same procedure of the whole experiment 20 times and obtain the average measures of precision, recall,
ACC and F1 for validation and testing shown in the Table 2.9. It shows our SVM classification-based matching has high and similar performance compared to previous similarity metric-based algorithms.

Table 2.9: SVM validation and testing results on synthetic numerical and non-numerical ground truth

<table>
<thead>
<tr>
<th>Fields</th>
<th>Data</th>
<th>Precision</th>
<th>Recall</th>
<th>ACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerical</td>
<td>Validation</td>
<td>0.997</td>
<td>0.994</td>
<td>0.995</td>
<td>0.995</td>
</tr>
<tr>
<td></td>
<td>Test (20%)</td>
<td>0.973</td>
<td>0.972</td>
<td>0.972</td>
<td>0.972</td>
</tr>
<tr>
<td>Non-numerical</td>
<td>Validation</td>
<td>0.949</td>
<td>0.958</td>
<td>0.953</td>
<td>0.953</td>
</tr>
<tr>
<td></td>
<td>Test (20%)</td>
<td>0.938</td>
<td>0.955</td>
<td>0.946</td>
<td>0.947</td>
</tr>
</tbody>
</table>

2.4.6 Comparisons with Other Existing Algorithms

We compare our technique NEMA with other existing algorithms-COMA system [42] and rule-based Regex [114] here. COMA is a state-of-the-art and popular hybrid
matching tool and system supporting both schema-level and instance-level matching. Regex is an instance-level rule-based matching method based on regular expressions. We test and compare their matching results on the rest 40% of numerical and non-numerical ground truth field matching, respectively.

2.4.6.1 Comparison of quality

We measure the quality of precision, recall, ACC, and F1 and compare the COMA in the schema level and instance level and Regex in the instance level for numerical and non-numerical data matching. On the schema level matching, COMA uses the best field matching similarity "0" (which has no corresponding line in the COMA system) as a threshold in the schema-level matching. On the instance level matching, COMA has one similar instance-level matching that uses aggregated maximum record-wise similarities to obtain the final field pair similarities. The record-wise similarity is based on common similarity metrics such as edit distance [136] and trigram [9]. Edit distance is to measure how dissimilar two strings are to one another by counting the minimum number of operations required to transform one string into the other. Trigram is to split a string into triples of characters and comparing those to the trigrams of another string. The field matching similarity between two fields $A$ and $B$ in COMA is defined as follows:

$$
\text{sim}(A, B) = \frac{1}{m + n} \cdot \left( \sum_{i=1}^{m} \max_{j=1,...,n}(\text{sim}(a_i, b_j)) + \sum_{j=1}^{n} \max_{i=1,...,m}(\text{sim}(b_j, a_i)) \right)
$$

(2.11)

Regex is a matching method based on regular expression by creating patterns from sampling instances of one field and then match against instances of another field to decide matching.
Table 2.10 shows the quality comparisons among COMA, NEMA and Regex. COMA-SCH is COMA with schema-level matching algorithm. COMA-ED and COMA-TRG means that COMA uses edit distance and trigram to measure the record similarity in instance-level matching, respectively. Regex is the matching based on regular expression. NEMA-(RD+BDP) is NEMA using combined RD and BDP in numerical field matching, while NEMA-TPM is NEMA using TPM in non-numerical field matching. NEMA-MH_LSH indicates that NEMA uses minHash-locality sensitive hashing in non-numerical field matching. The accuracies and F1 scores of NEMA-(RD+BDP) and NEMA-TPM in numerical and non-numerical data matching can be up to 95%, as high as COMA’s non-numerical data matching, but having better performance than COMA-SCH and Regex matching. For numerical field matching, the accuracies of COMA-ED, COMA-TRG and Regex are 7%-15% lower than NEMA-(RD+BDP) because of the ineffectiveness to identify non-matched pairs of numerical ground truth. For non-numerical field matching, the highest accuracies and F1-scores of COMA-ED and COMA-TRG are only 1-2% higher than NEMA-TPM. However, the field matching score of COMA is measured based on its general string similarity matching, which is not well applied to the network management database matching for record pair similarity requirements. A large number of pairs with high record similarities in COMA are not thought of as matches in the network management databases, which shows the usefulness of the NEMA non-numerical algorithm.

2.4.6.2 Comparisons of Mismatched Examples

We further analyze the differences of COMA and NEMA in matching the ground truth field pairs. Table 2.12 shows the field pairs in every row and its similarity scores by COMA and NEMA. These field pairs are found to be matched pairs by NEMA with relatively high similarity scores, but COMA shows no similarities with score 0. For COMA, the field names have very few common characters in spelling, even though
Table 2.10: Quality comparisons with other existing algorithms

<table>
<thead>
<tr>
<th>Field</th>
<th>Algorithms</th>
<th>Precision</th>
<th>Recall</th>
<th>ACC</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerical</td>
<td>COMA-SCH</td>
<td>0.867</td>
<td>0.838</td>
<td>0.85</td>
<td>0.852</td>
</tr>
<tr>
<td></td>
<td>COMA-ED</td>
<td>0.848</td>
<td>0.933</td>
<td>0.883</td>
<td>0.889</td>
</tr>
<tr>
<td></td>
<td>COMA-TRG</td>
<td>0.765</td>
<td>0.867</td>
<td>0.8</td>
<td>0.813</td>
</tr>
<tr>
<td></td>
<td>Regex</td>
<td>0.833</td>
<td>0.833</td>
<td>0.833</td>
<td>0.833</td>
</tr>
<tr>
<td></td>
<td>NEMA-(RD+BDP)</td>
<td>0.966</td>
<td>0.933</td>
<td>0.95</td>
<td>0.949</td>
</tr>
<tr>
<td>Non-numerical</td>
<td>COMA-SCH</td>
<td>0.781</td>
<td>0.833</td>
<td>0.8</td>
<td>0.806</td>
</tr>
<tr>
<td></td>
<td>COMA-ED</td>
<td>1.0</td>
<td>0.933</td>
<td>0.967</td>
<td>0.966</td>
</tr>
<tr>
<td></td>
<td>COMA-TRG</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
</tr>
<tr>
<td></td>
<td>Regex</td>
<td>0.867</td>
<td>0.897</td>
<td>0.883</td>
<td>0.881</td>
</tr>
<tr>
<td></td>
<td>NEMA-TPM</td>
<td>0.936</td>
<td>0.967</td>
<td>0.95</td>
<td>0.951</td>
</tr>
<tr>
<td></td>
<td>NEMA-MH_LSH</td>
<td>0.933</td>
<td>0.848</td>
<td>0.883</td>
<td>0.889</td>
</tr>
</tbody>
</table>

Table 2.11: Efficiency comparison with other existing algorithms

<table>
<thead>
<tr>
<th>Non-numerical Algorithms</th>
<th>Time Mean (s)</th>
<th>Time SD (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMA-SCH</td>
<td>251</td>
<td>21</td>
</tr>
<tr>
<td>COMA-ED</td>
<td>13,322</td>
<td>1,289</td>
</tr>
<tr>
<td>COMA-TRG</td>
<td>12,890</td>
<td>1,329</td>
</tr>
<tr>
<td>Regex</td>
<td>425</td>
<td>89</td>
</tr>
<tr>
<td>NEMA-TPM</td>
<td>2,832</td>
<td>159</td>
</tr>
<tr>
<td>NEMA-MH_LSH</td>
<td>939</td>
<td>62</td>
</tr>
</tbody>
</table>

the semantic commonality exists. NEMA does not rely on the inaccurate schema-level properties, but it uses the record instance for the decisions of field matching, which indirectly considers the semantic correspondences. If the record instances for some matched pairs are incomplete or missing, however, the similarity scores for these field pairs are also low. Table 2.13 shows the two field pairs that have low similarities in NEMA. Although the field names in each pair express the same thing semantically, the record instances in the fields are actually incomplete and have very few in common between each other. However, in our databases, the missing or incomplete field pairs are very few compared to the large number of field pairs, which does not affect the overall performance for the numerical field matching with NEMA.
Table 2.12: Example of field pairs matched by NEMA, but not by COMA

<table>
<thead>
<tr>
<th>Table.fieldA</th>
<th>Table.fieldB</th>
<th>COMA-ED</th>
<th>NEMA-TPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_INCI.ins_site_key</td>
<td>T_SITE.partysite_id</td>
<td>0</td>
<td>0.208</td>
</tr>
<tr>
<td>T_OR.HD.creator_id</td>
<td>T_INCI.lastup_by</td>
<td>0</td>
<td>0.643</td>
</tr>
<tr>
<td>T_CT.temp_desc</td>
<td>T_PROD.item_desc</td>
<td>0</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 2.13: Example of field pairs matched by COMA, but not by NEMA

<table>
<thead>
<tr>
<th>Table.fieldA</th>
<th>Table.fieldB</th>
<th>COMA-ED</th>
<th>NEMA-TPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_INCI.bl_cot_key</td>
<td>T_CT.bl_cot_key</td>
<td>0.750</td>
<td>0.091</td>
</tr>
<tr>
<td>T_SUR.bl_surv_key</td>
<td>T_SUR_ANS.bl_surv_key</td>
<td>0.76</td>
<td>0.009</td>
</tr>
</tbody>
</table>

Here we analyze the specific record pair examples of non-numerical instance-level matching. COMA uses standard edit distance and trigram to calculate the similarities of records, which is not quite suitable for the matching requirement of network management databases. Table 2.14 below shows 9 examples of records pairs and three different kinds of similarities (NEMA-TPM, COMA-ED, COMA-TRG). The first 7 rows as one group are thought of as matched record pairs, the last two rows in the other group are non-matched record pairs. We can see from that the similarity of matched pairs based on COMA are quite similar around 0.7 for these two groups, from which is not easy to differentiate. While the similarities by NEMA have good differences (0.333 for non-matched pairs, 0.4 above for matched pairs). This further demonstrates that NEMA is more suitable for the network database matching.

Table 2.14: Examples of record similarity comparisons

<table>
<thead>
<tr>
<th>Record 1</th>
<th>Record 2</th>
<th>NEMA-TPM</th>
<th>COMA-ED</th>
<th>COMA-TRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>c900 series</td>
<td>c900 series</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c2950 series</td>
<td>c2916 series</td>
<td>0.8</td>
<td>0.833</td>
<td>0.6</td>
</tr>
<tr>
<td>1601r series</td>
<td>1601 series</td>
<td>0.775</td>
<td>0.909</td>
<td>0.738</td>
</tr>
<tr>
<td>css2950</td>
<td>css2916</td>
<td>0.667</td>
<td>0.714</td>
<td>0.6</td>
</tr>
<tr>
<td>C2960</td>
<td>C2960CX</td>
<td>0.577</td>
<td>0.6</td>
<td>0.775</td>
</tr>
<tr>
<td>C3560CX</td>
<td>C3560X</td>
<td>0.5</td>
<td>0.833</td>
<td>0.671</td>
</tr>
<tr>
<td>AIR35CE</td>
<td>AIR35SE</td>
<td>0.4</td>
<td>0.857</td>
<td>0.6</td>
</tr>
<tr>
<td>c800</td>
<td>s800</td>
<td>0.333</td>
<td>0.75</td>
<td>0.5</td>
</tr>
<tr>
<td>ts 900</td>
<td>cs900</td>
<td>0.333</td>
<td>0.8</td>
<td>0.667</td>
</tr>
</tbody>
</table>

52
2.4.6.3 Comparison of Efficiency

Considering the expensive time consumption for non-numerical field pair matching, we test the efficiency based on the whole non-numerical ground truth. We run the experiment 20 times on the same machine with the same data to calculate the average computation time and standard deviation (SD) without data loading time. Table 2.11 shows the total computation time spent for COMA-SCH, COMA-ED, COMA-TRG, Regex, NEMA-TPM and NEMA-MH_LSH. Among them, COMA-SCH and Regex are two fastest among all the algorithms, but the accuracies are the lowest. COMA-ED is the slowest, taking about 13,322 seconds. NEMA-TPM takes 2,832 seconds, about 5x speedup over COMA-ED. NEMA-MH_LSH takes 939 seconds, which is about 14 times faster than COMA-ED in the cost of 8% accuracy lost. NEMA-TPM is slower than COMA-SCH and Regex algorithms, but with about 8% higher accuracy. Therefore, NEMA-TPM outperforms most of other existing algorithms and reaches the best trade-off of quality and efficiency among them.

2.5 Related Work

The structured data matching is an old and important research topic but unsolved and ever-growing problem, which has a wide range of applications in database integration, migration, semantic query, etc. [19]. In the survey paper [144], the authors propose a solution taxonomy differentiating between element and structure level, schema and instance level, language and constraint-based matching techniques. Furthermore, P. Shivaiko et al. [145] review the state-of-the-art matching systems which were based on strings, structure, data instance and semantics matching techniques using different schema formats such as database, XML, OWL, RDFS, etc. In database schema matching, previous common matching systems in schema-level are introduced in several prototypes such as Similarity Flooding (SF) [115], Coma [10], etc.
SF [115] is a matching algorithm that models two structured columns to be compared as two directed labeled graphs. It makes use of field data, key properties and the string-based alignment (prefix and suffix test) to obtain the alignments between two nodes of the graph. The similarity is calculated from similar nodes to adjacent neighbors through propagation. Our NEMA only relies on the data instance values to infer the matching of fields, which does not utilize the structured properties and data types. However, SF uses a metric for matching quality based on the intended matching results, which is similar to our accuracy metric based on top-k results.

Coma [10] is a composite matching system providing extensible library and framework for combining obtained results. It contains mainly 6 elementary matchers using string-based techniques, 5 hybrid matchers using names and structural paths, and one reuse-oriented matcher based on previous matching results. The composite matcher effectively improves the match quality over a single matcher using the default combination strategy. Compared to SF, the overall average matching quality are the best among them [10]. The extended version Coma++ [42] utilizes the shared taxonomy and pivot schema to further improve the overall matching quality. In our evaluation, we compare with the Coma++ method using the default combination strategy and find our technique NEMA overall outperforms than COMA in schema-level matching.

Except from the previous matching approaches using field and structural information matching, data instance-based approaches [121, 109, 26, 176] use the similarity metric or machine learning or rule-based methods to determine the similarity of fields. In [109], the authors utilize a corpus that contains schema and mappings between some schema pairs, and learn the constraints from schema statistics to help generate more matching pairs. In [26], the authors use the mutual information of statistics to measures the similarity of schema instances between two columns to decide the matching, which shows an effective method based on instances. Also, the authors in [129] propose a new sample-driven approach which enables the end-users to easily
construct their own data to match the source and target schema. [114] proposes a
rule-based method by creating regular expression pattern to match against columns.
[176] uses matching-learning technique of training neural networks for getting candidate pairs and then filters the pairs with a rule-based algorithm. Our NEMA uses proposed similarity metrics as features to train a SVM to classify for matching effectively and efficiently. COMA [42] proposes two instance-level matching methods based on the constraint of instance data and the content-based matching to measure field matching. The constraint-based method relies on the general, numerical and pattern constraint which has specific limitation to the specific data which is not suitable for the network databases. The content-based matching depends on the aggregation of similarity scores of instance contents and it is kind of similar to our NEMA technique on content-based similarity measurement.

To sum up, most of currently popular matching approaches and systems focus on schema-level information matching. The data instances level matching approaches using field record values are mostly based on some statistical models and machine learning from corpus. We further explore the database instance matching by comparing field records using different metrics and propose effective and overall matching algorithms considering the characteristic of network database matching for a graph database construction for efficient data query, analysis and management.

2.6 Conclusion

In this work we propose a systematic technique NEMA to match databases for network management. Different from previous database matching approaches, we design a technique to match numerical and non-numerical fields in instance-level respectively, which can effectively be integrated into a graph database for network management and analysis. For numerical field matching, we propose range difference similarity and bucket dot product similarity metrics. For non-numerical field match-
ing, we design top priority match metric and also propose applying minHash-locality sensitive hashing algorithm, which reduces the matching time for large databases. To address the drawback of manual thresholds, an effective classification-based method is also proposed based on the proposed similarity metrics. NEMA are experimentally demonstrated with best trade-off of qualify and efficiency among other existing algorithms.

With the explosion of big data and popularity of distributed graph processing systems, this work has the potential to significantly reduce the human work involving identifying the matching fields for a large graph database construction and also be applied for large-scale data matching. A majority of partial matching pairs can be found by our matching algorithms which are not easily detected by humans.

This work primarily discusses the big database integration and lays the foundation for network management on a graph database. It has direct implications for network management to help network operators/administrators with network query, network diagnosis, fault detection, network performance monitoring, etc. One specific example for network query is that it is more efficient to use graph traversal algorithms to find out which network routers communicated with CiscoASR9010 have the most frequent incidents in the last year. Another example is that if a network ticket/incident occurs, with the help of graph clustering or propagation models, the administrators could easily locate the network failure and other affected networks, and analyze its root cause. The system can also automatically suggest a potential solution to the network failure based on previous histories of tickets/incidents.
CHAPTER 3

SCALABLE GRAPH QUERY ON A GRAPH DATABASE
WITH HIERARCHICAL INHERITANCE RELATIONS

3.1 Introduction

Many real-world systems, such as enterprise networks, social networks, and biological networks, can be modeled as heterogeneous information networks (HIN) [37, 56, 143, 151]. HIN contains multiple types of objects and relations providing rich semantic queries, knowledge discoveries, information fusions, recommendations and predictions. Graph query, as an important technique for solving these tasks, has been extensively explored recently. It mainly explores subgraph isomorphism algorithms to get an exact match [49, 50], and also to develop subgraph matching algorithms to do an inexact/approximate match as the potential query answers [83, 77].

Current research on graph query/matching mainly focuses on two dimensions. The first dimension is the unary node-to-node properties mapping. The second dimension is edge-to-edge/path similarities. Jin et al. [77, 78] consider node types and closest path propagations to get scores of query answers. Some work [83, 174, 77, 58] considers similar nodes’ labels and their neighbors to learn the path propagation to get ranked answers.

However, knowledge representation has hierarchical structures in the real world system. Long et al. [106] state that the knowledge structure representation can be inherited with upward and downward inheritances. Clauset et al. [32] show that the existing knowledge of hierarchical structure can be used to predict missing connections. In addition, Jiang et al. [76] construct the hierarchical structures of entities for
the large freebase knowledge base system based on real world entities and relations. One visible example in an enterprise’s product database is that product vulnerabilities can be inherited from or passed down to different product versions. While measuring the similarity of objects for graph matching, hierarchical inheritance relations can also play an important role for the answer ranking. The quality of query answers is also greatly affected by hierarchical inheritance relations. Therefore, we consider the power of hierarchical inheritances whereby a subclass inherits the properties and constraints of its parents, and more meaningful and accurate query answers are expected to be obtained.

Taking an example of an information network with hierarchical structure, we consider a schema of an enterprise’s product information network shown in Figure 3.1(a). Every node represents a type of entity at the schema level. The product type is connected by four property types: site, workgroup, technology and vulnerability. Product entities have hierarchical connections with different versions of the products shown in Figure 3.1(b). Some properties are inherited among different versions of the products, such as vulnerability and technology properties (in red bold lines in Figure 3.1(a)).

Given the information network schema with inherited relations, we show a user query example here. Assume a user wants to find the top-5 related products affected with a given vulnerability $V_1$ (Cisco WebEx meetings server information disclosure vulnerability) and employed with a given technology $T_1$ (voice - communications manager additional apps and plugins), which is constructed as a user query graph shown in Figure 3.1(c). Figure 3.1(d) shows the top-5 subgraph answers of the query in this answer graph. The given $V_1$ and $T_1$ node in user query graph is exactly matched with the $V_1$ and $T_1$ node, respectively, in the answer graph, and there are 5 product nodes which are potential answers to the query of the product node.
For general methods, if we consider the closest node types and shortest distances to measure the similarity of answers for matching, we obtain the following ranking order of answer scores, \( P_1 \) (Cisco WebEx meetings server versions 0.1.0), \( P_2 \) (Cisco WebEx meetings server versions 0.2.0), \( P_3 \) (Cisco WebEx meetings server versions 1.1), \( P_4 \) (Cisco WebEx meetings server versions 2.1) and \( P_5 \) (Cisco Jabber for Windows), that is, the ranking order of answer scores is \( s(P_1) > s(P_2) = s(P_3) > s(P_4) > s(P_5) \).

However, the vulnerability property can be inherited from different prior versions of products. Here \( P_1 \) is the prior (parent) version of \( P_3 \), and \( P_3 \) is the parent version of \( P_4 \) as the arrows indicate. Hence, \( P_1 \)'s vulnerabilities can pass down to the product \( P_3 \) or \( P_4 \), and \( P_4 \)'s vulnerability can come from the upper \( P_3 \) or \( P_1 \). With the hierarchical inheritances, the answer scores can be obtained with a more accurate ranking order \( s(P_1) \approx s(P_3) \approx s(P_4) > s(P_2) > s(P_5) \), which is very important for engineers’ troubleshooting and customers’ queries.

Due to the complexity and heterogeneousness of large networks, designing an effective and efficient algorithm with additional hierarchical features is challenging. In this paper, we conquer this problem by modeling graph queries with a new matching score function with hierarchical inheritance relations for effective answers, and by proposing a bound-based technique for an efficient query. The main contributions are as follows:

- We formulate the graph query problem with hierarchical inheritance relations to improve the query quality.

- We propose a new graph query algorithm based on uniform cost search in the context of a new matching score function.

- We design a bound-based method to prune the search spaces to efficiently get the top-\( k \) best answers.
Additional apps and plugins

- We implement our algorithm in the Spark GraphX distributed environment for large-scale networks. Experiments are done to evaluate the effectiveness and efficiency of our matching algorithm.
The rest of this paper is organized as follows. Section 3.2 describes the problem and formulates graph queries with hierarchical inheritance relations. The proposed algorithm for graph queries and its bound-based pruning technique are presented in Section 3.3. Section 3.4 discusses the distributed implementation. In Section 3.5, we present the evaluation of our algorithms. The related work and conclusion are shown in Sections 3.6 and 3.7, respectively.

3.2 Problem Formulation

3.2.1 Data Graph, Query Graph and Matching

We consider a HIN that contains hierarchical inheritance relations among nodes as a hierarchical heterogeneous information network (HHIN). HHIN is modeled as a partially undirected, labeled data graph \( G(V, E, L_v, H_e) \) with a node set \( V \), edge set \( E \), node label set \( L_v \) and hierarchical inheritance relations \( H_e \) with directions, where (1) each node \( v \in V \) represents an entity in \( G \), (2) each edge \( e \in E \) represents the relationship between two entities, and each edge weight is considered to be 1. Only an edge between two hierarchical entities has a direction. (3) each node \( v \) has a label information \( L_v \), including at least a node type and a keyword description, (4) for hierarchical entities, each edge \( e \in H_e \) between them indicates a hierarchical inheritance relation. Each edge weight between two hierarchical entities is \( |H_e| = 1 \).

There exists upward and downward hierarchical inheritance relations in \( G \). We call a node with label information that is inherited among other hierarchical entities as an “attaching” node, such as a vulnerability node in Figure 3.1(a) whose label information that could be inherited among product nodes. A node with a node type that has hierarchical levels is called an “inherited” node, such as a product node in Figure 3.1(a). If an attaching node’s label information passes down to its inherited node’s lower level entity, we call it downward inheritance. Conversely, if an attaching node’s label information can pass up to its inherited entity’s higher level entity, it
is called upward inheritance. The attaching node and one of its inherited nodes are formed as a “property inheritance pair”. For example, the vulnerability’s label information in the vulnerability entity can be downward or upward inherited from product entities in higher or lower levels as shown in red bold line in Figure 3.1(a). The label information of the workgroup or site is not inherited among product nodes as shown in the black line in Figure 3.1(a).

Given an hierarchical edge $h(u_1, u_2)$ between node $u_1$ and $u_2$, the hierarchical level difference is 1 for upward inheritance when $u_1$ is in the higher level than $u_2$, -1 for downward inheritance when $u_1$ is in the lower level than $u_2$, and 0 for the same hierarchical level or non-hierarchical relations. Then, given a node pair in the query graph $Q$, there are possible pairs of nodes $(S, V_{i,j})$ in data graph $G$ which match with that node pair, where $S$ is an “attaching node” and $V_{i,j}, i, j \in 1, 3$ indicates different “inherited” candidate nodes shown in Figure 3.2. There are basically the following cases if we only consider the matching based on hierarchical inheritance relations.

1. When $S$ inherits upward from $V_{3,1}$ to $V_{1,1}$, the node similarity score $r$ of $S$ to $V_{1,1}$ is a little smaller than the score of $S$ to $V_{3,1}$, that is, $r(S, V_{1,1}) \lesssim r(S, V_{3,1})$ shown in Figure 3.2a.

2. When $S$ inherits upward from $V_{3,1}$ to $V_{1,1}$ and downward from $V_{1,1}$ to $V_{3,2}$, it is expected that the node similarity score $r(S, V_{3,2}) \gtrsim r(S, V_{1,1})$ shown in Figure 3.2b.

3. When $S$ inherits downward from $V_{1,1}$ to $V_{3,1}$, it is expected that the node similarity score $r(S, V_{3,1}) \gtrsim r(S, V_{1,1})$ shown in Figure 3.2c.

4. When $S$ inherits downward from $V_{1,1}$ to $V_{3,1}$ and upward from $V_{3,1}$ to $V_{1,2}$, it is expected that the node similarity score $r(S, V_{3,1}) \gtrsim r(S, V_{1,2})$ shown in Figure 3.2d.
5. When $S$ inherits upward from $V_{1,1}$ to $V_{3,1}$, and there are multiple shortest paths from $S$ to $V_{1,1}$ with different hierarchical level differences, the maximum hierarchical difference is defined as: $max(h(S, V_{1,1})) = -2$ shown in Figure 3.2e.

6. When $S$ inherits downward from $V_{1,1}$ to $V_{3,1}$, and there are multiple shortest paths from $S$ to $V_{3,1}$ with different hierarchical level differences, the maximum hierarchical difference is defined as: $max(h(S, V_{3,1})) = 2$ shown in Figure 3.2f.

Figure 3.2: Hierarchical inheritance cases and scores
A query graph \( Q(V_Q, E_Q, L_v) \) is modeled as an undirected and labeled graph. \( V_Q \) contains a set of specific nodes \( V_Q^S \) and a set of query nodes \( V_Q^U \) with types \( \tau_Q^U \), which are provided by users. A specific node is defined as an instantiated node in \( Q \) that has a fixed node type and node label information, and it is also matched to a node in \( G \). A query node is defined as a node in \( Q \) that only its node type is given, and we want to find its matched nodes in \( G \). According to one classification category based on query node number in \( Q \), if the query node number \( |V_Q^U| = 1 \), we denote the query graph \( Q \) as a star query graph. If the query node number \( |V_Q^U| > 1 \), \( Q \) is called a general (non-star) query graph. According to another classification category based on hierarchical inheritance relations, if every one of the specific nodes in \( V_Q^S \) can form a property inheritance pair with its query node in \( V_Q^U \), we call \( Q \) a hierarchical query graph. If there exists no property inheritance pairs, we call \( Q \) non-hierarchical query graph. Otherwise, it is called a mixed hierarchical query graph. For example, Figure 3.1(c) shows a hierarchical star query graph where node \( V_1 \) and node \( T_1 \) comprise specific nodes, and the node marked with the “?” in the product type represents a query node.

Given a query graph \( Q \) and a data graph \( G \), we need to map each query node to a data node. This transfers to a subgraph matching problem. We denote as \( M \) an already matched subgraph in \( G \) to \( Q \). Then a subgraph matching is a many/one-to-one mapping function \( \phi: V_Q \rightarrow V \), such that, for each query node \( v \in V_Q \), \( \phi(v) \in M \). The problem here is to find such top-\( k \) potential mapping functions given a query graph \( Q \) and a data graph \( G \).

### 3.2.2 Matching Score

If nodes are close in a query graph, their mapping nodes in a data graph are also close based on node neighbors and hierarchical inheritance relations. Given a query...
graph \( Q \) containing a node pair \((u, v) \in V_Q\) that is connected, a matched subgraph \( M \) in \( G \) has mapped nodes \((\phi(u), \phi(v))\).

To measure how close is a query node in \( Q \) to its mapping node in \( G \), we define “node closeness score” based on whether hierarchical inheritances exist in \( Q \).

(1) When \( u \) and \( v \) in \( Q \) do not form a property inheritance pair, the closeness score of \((\phi(u), \phi(v))\) is defined similar to [78]. It considers the shortest distance in the graph with an exponential monotonic descending relationship. The shorter of the distance, the higher the closeness score.

\[
    r(\phi(u), \phi(u)) = \begin{cases} 
    1 & \text{if } \phi(u) = \phi(v) \\
    \alpha^{l(\phi(u), \phi(v))} & \text{otherwise}
\end{cases} \tag{3.1}
\]

where \( l(\phi(u), \phi(v)) \) is the shortest distance from \( \phi(u) \) to \( \phi(v) \). \( \alpha \) is a constant propagation factor in \([0, 1]\) that controls the decreasing rate of node closeness.

(2) When \( u \) and \( v \) can form a property inheritance pair, we consider both shortest distance and hierarchical inheritance relations. The inheritance relation has positive impact on the query result and has decreased the short distance to some extent. The inheritance level difference will be deducted to some extent from the shortest distance. Therefore, we define the closeness score of \((\phi(u), \phi(v))\) as:

\[
    r(\phi(u), \phi(v)) = \begin{cases} 
    1 & \text{if } \phi(u) = \phi(v) \\
    \alpha^{l(\phi(u), \phi(v)) - \beta \cdot |h(\phi(u), \phi(v))|} & \text{otherwise}
\end{cases} \tag{3.2}
\]

where \( l(\phi(u), \phi(v)) \) is the shortest distance from \( \phi(u) \) to \( \phi(v) \). \( \alpha \) is a constant propagation factor in \([0, 1]\) that controls the decreasing rate of the node closeness. \( \beta \) is defined as the hierarchical level propagation factor in \((0, 1)\), which indicates the importance of hierarchical level propagation when an attaching node’s label information inherits between different hierarchical levels of an inherited node. \( \beta \) is expected
to be smaller than $\alpha$ because hierarchical inheritance is more reliable than shortest distances when traversing long hops. $h(\phi(u), \phi(v))$ indicates the hierarchical level difference from $\phi(u)$ to $\phi(v)$. Vice versa, the hierarchical level difference from $\phi(v)$ to $\phi(u)$ is indicated as $h(\phi(v), \phi(u))$, and $h(\phi(v), \phi(u)) = -h(\phi(u), \phi(v))$.

Based on the node closeness score, the matching score of $M$ is defined as the summation of mapping nodes $(\phi(u), \phi(v))$ for all connected edges $(u, v)$ in $Q$.

$$F(\phi) = \sum_{(u,v) \in E_Q} r(\phi(u), \phi(v))$$  \hspace{1cm} (3.3)

### 3.2.3 Problem Statement:

Given a query graph $Q$ and a data graph $G$, we want to find the top-$k$ subgraph answers in $G$, that is, to find a set of $k$ subgraphs $M_k$ in $G$, such that for any nodes $\phi(V_Q) \in M_k$ and for all nodes $\phi'(V_Q) \notin M_k$, the matching score $F(\phi) > F(\phi')$. Specific nodes $V_Q^S$ in $Q$ are identified for exactly one-to-one mapping to matched nodes $\phi(V_G^A)$ in $G$ (we call them anchor nodes $V_G^A$), which are easily to be found. Therefore, we consider the top-$k$ sets of candidate nodes in $M_k$ for a set of query nodes based on hierarchical inheritance relations and graph structures.

Formally, given a query graph $Q(V_Q, E_Q, L_v)$, the top-$k$ subgraphs $M_k(V', E', L'_v)$ have the following mapping function with $Q$. For each $v \in V_Q$, there is a one-to-one mapping $\phi(v) \in V'$: $v \rightarrow \phi(v)$ based on the matching score $F$. Our problem considers exact and approximate matches to output the top-$k$ matching answers, so the edge of $e \in E_Q$ does not need to have a one-to-one mapping to the edge $e' \in E'$.

### 3.3 Graph Query Algorithm with Hierarchical Inheritance Relations

It is time-consuming to get all potential subgraphs from a large-scale data graph with a big query graph. Moreover, for a general query graph with multiple query
nodes, it is proved to be an NP-hard problem even for subgraph isomorphism [92]. Yang et al. [173] divide a query graph into star queries and then utilize the top-k star-join method, using the similar relational database HRJN [68]. Inspired by the structure of our general query graph with multiple query nodes, we propose a general graph query algorithm comprising three phases as follows:

**Phase 1 (Query Decomposition):** A general query graph contains some specific nodes and one or more query nodes. Considering the characteristics of our query graph, the decomposing policy of a general query graph is not as complex as the decomposing method considered in [173], as we don’t use the join for final combinations of star queries. Therefore, a simple and effective policy is to use the number of query nodes as the number of star query graphs. Each query node is the center query node for each star query, every specific node that is connected to the center query node is a specific node for its star query.

**Phase 2 (Star query):** We propose to use uniform cost search and bound-based pruning to derive top-k candidates for each star query. Selecting the top-k candidates for each star query can effectively serve the final top-k candidate results for a general graph query (Section 3.3.1–3.3.5).

**Phase 3 (Candidates selection):** We consider the top-k star query candidates together and get the optimal edge/path matching scores for query node combinations. Different from top-k join strategy with a common node in [173], here query node candidates can be 1 or more hops connected in G without a common node for joining. Therefore, graph traversals are needed among these star query node candidates to find the final top-k candidate sets for query nodes. When there are \(|V_Q^U|\) query nodes, this involves exponential \(|V_Q^U|^{k_s}\) computations, which is highly expensive if \(|V_Q^U|\) and \(k_s\) are large. We propose to use a branch and bound technique to greatly reduce search spaces by filtering out unexpected candidate sets (Section 3.3.6.3).
If the input query graph $Q$ is a star query graph, then we only do phase 2 (star query) to get the answer. If the input query graph $Q$ is a general query graph, it will involve the three phases. As query decomposition is easy to accomplish, we will mainly discuss star query algorithm and candidate selection for general query graph algorithm.

### 3.3.1 Matching Score for Star Query

Given a star query graph $Q$ with a set of specific nodes $V_Q^S$ and a query node $v$, specific nodes $V_Q^S$ have mapped to anchor nodes $V_G^A$ in $G$, so we only need to find the top-$k$ mapping nodes $\phi(v)$ for $v$. We traverse from every anchor node $\phi(v^s)$ in $V_G^A$ as one source with uniform cost search to all other nodes in the data graph, and obtain their node closeness score to the source. We denote $S(\phi(v))$ as the matching score of one $\phi(v)$ based on the aggregated results of node closeness scores from all the nodes in $\phi(V_Q^S)$:

$$S(\phi(v)) = \sum_{v^s \in V_Q^S} r(\phi(v^s), \phi(v))$$  \hspace{1cm} (3.4)

### 3.3.2 Bound-based Pruning for Star Query

For each different anchor node, there is a propagation path to each candidate node in $G$. It is time-consuming to do all node traversals if $G$ is very large. We use bound-based pruning technique to effectively reduce the search space for star queries. We trace the lower bound in the top-$k$ answers and infer the upper bound of unseen nodes to effectively filter these nodes while traversing.

#### 3.3.2.1 Bounds of Matching Score

In the top-$k$ answer list of query nodes, every node is maintained with a upper bound of matching score and a lower bound of matching score for a query node. We refine the upper bound $\overline{S}(\phi(v))$ and lower bounds $\underline{S}(\phi(v))$. 

68
The matching score bound depends on the upper bound of node closeness score \( r \) and lower bound of closeness score \( r \). Next, we show how to get these bounds of node closeness score.

### 3.3.2.2 Bounds of Node Closeness Score

The lower bound and upper bound are obtained online while the graph traversal is operated. We show the lower and upper bound refinement in the different iterations of graph traversal. We denote \( t \) as the iteration number of uniform cost search from an anchor node \( s \) to a candidate node \( u \).

1. The initial bounds are set as \( r^0(s, u) = 1 \) and \( r^0(s, u) = 0 \).
2. In each of the next iterations, every node \( u \) is updated with its lower bound using the information from its previous iteration result when it is not visited yet. The lower bound is computed as follows:

\[
\tau^t(s, u) = \begin{cases} 
\tau^{t-1}(s, u) & \tau^{t-1}(s, u) > 0 \\
\alpha^{1-\beta|\delta(u, u)|} \cdot \tau^{t-1}(s, u_{prev}) & \text{otherwise}
\end{cases}
\]

where \( u_{prev} \) is the parent node of \( u \) when traversing from \( s \) along a path to \( u \).

The upper bound in iteration \( t \) is computed as follows:

\[
\tau^u(s, u) = \begin{cases} 
\tau^t(s, u) & \tau^t(s, u) > 0 \\
\alpha^{1-\beta|\delta(s, u)|} & \text{otherwise}
\end{cases}
\]

where \( u_{prev} \) is the parent node of \( u \) when traversing from \( s \) along a path to \( u \).

### 3.3.3 Top-k Selection with Bounds

How to effectively update potential candidate results and select the final top-\( k \) results during iterations is crucial for computation performance. Here we use a top-\( k \)
selection policy based on the upper and lower bounds of matching scores referred as the top-k emergence test in [85]. We maintain a top-k candidate result in a priority queue \( P \). Each candidate node \( u \) contains its lower bound \( S_G(u) \), and upper bound \( S_G(u) \). We define \( S_{kth} \) as the smallest lower bound in \( P \). The process for selecting and updating \( P \) during the iterations is shown as follows:

1. Find the top-k potential answer nodes and put in \( P \).
2. Calculate the kth smallest lower bounds \( S_{kth} \) in \( P \).
3. If the upper bound \( S_G(u) \) of an incoming node \( n \) is less than the \( S_{kth} \), we prune the node \( u \) and the nodes with bigger distance than \( u \) from the starting source. These nodes' matching scores are lower than any node's matching score in \( P \), so they are not qualified for top-k final results.
4. Continue the previous steps until the convergence condition is reached, which is shown in Section 3.3.4.

### 3.3.4 Convergence of Iteration Propagation

The iteration propagation in essence is a graph traversal problem, the convergence of this graph query is equivalently bounded by the traversal of all the required nodes or no update of the propagation cost. Therefore, two types of iteration conditions are identified to terminate the graph propagation to obtain the final top-k answers.

1. When all the nodes with designated query node types have been explored or pruned by the bound-based pruning technique (Section 3.3.3), all the visited candidate nodes have obtained the necessary matching scores.
2. When no message is updated for the next propagation, that is, all the candidate nodes' matching scores keep the same as the last iteration.

### 3.3.5 Star Query Algorithm

According to the proposed star query matching score and bounding-based pruning, we show our star query with hierarchical inheritance relation algorithm (SQH) in Algorithm 1. First, we obtain anchor nodes \( \phi(V_Q^S) \) in \( G \) for each \( V_Q^S \) in \( Q \), which
are specific one-to-one mappings in $G$ (in Line 1). Then we aggregate node messages to do propagation simultaneously from each anchor node with a uniform cost search (in Line 8). Search cost of each node in the uniform cost search is indicated by the inverse of its matching score here. In each iteration of propagation, the candidate node closeness and matching score, lower bounds and upper bounds are updated (in Line 9–10). Candidate nodes and the queue are continuously updated (in Line 11–15). The specific top-$k$ selection and update are shown (in Line 17–28). Iterations continue until we find the final top-$k$ candidate result. The worst time complexity is $O(|V| \times |V_Q^S|)$, where $|V|$ is the node number of $G$. With the pruning of potential unmatched nodes, the average time complexity is reduced to $O(M \times |V_Q^S|)$, where $M$ is the number of visited nodes with type $\tau$ and $M \ll |V|$.

### 3.3.6 General Graph Query Algorithm

The general graph query problem involves three phases described in the earlier part of Section 3.3: decomposing query (phase 1), star query (phase 2), and candidate selection (phase 3). The previous 2 phases have been described before. For phase 3, how to effectively and efficiently select the top matching candidate sets from star query results involves effective candidate selections. We propose to find the top matching scores of query node combinations by propagations. First, we define the matching score of query nodes for general graph query.

#### 3.3.6.1 Matching Score of Query Nodes

Based on the definition of the matching score for star queries in Section 3.3.1, we define the matching score for a set of query nodes $V_Q^U$ as:

$$F_G(V_Q^U) = \sum_{v \in \phi(V_Q^U)} S_G(v) + \sum_{(v_i, v_j) \in E(V_Q^U)} E_G(\phi(v_i), \phi(v_j))$$  (3.9)
Algorithm 1: Top-$k$ star query (SQH)

**Data:** Data graph $G(V, E, L_v, H_e)$, Query Graph $Q(V^*_Q, \tau)$, Matching number $k$

**Result:** Top-$k$ match set $P_k$

1. Get anchor nodes set $\phi(V^*_Q)$ for $V^*_Q$ ;
2. Initialize empty match set $P_k$ (size $k$) Initialize node closeness score $r(s, u), r(s, u)$ and $r(s, u)$ ;
3. Initialize matching score $(S_G(u), \overline{S}_G(u), \underline{S}_G(u))$ ;
4. Initialize $L \leftarrow \{v|type(v) = \tau \& v \in V\}$ ;
5. $t \leftarrow 0$ ;
6. while $L$ not empty and message exists do
7. \hspace{1em} $M, L \leftarrow$ ITERATEPROPAGATIONUPDATE ;
8. \hspace{2em} Aggregate node $u$ from each anchor nodes with uniform cost search ;
9. \hspace{2em} Update $(r(s, u), \overline{r}(s, u), \underline{r}(s, u))$ by equation 3.7 and 3.8 ;
10. \hspace{2em} Update $(S_G(u), \overline{S}_G(u), \underline{S}_G(u))$ by equations 3.5 and 3.6 ;
11. \hspace{2em} if $\overline{S}_G(u) - \underline{S}_G(u) <= 0$ then
12. \hspace{3em} $L \leftarrow L - u$ ;
13. \hspace{2em} else
14. \hspace{3em} $P_k = P_k + u$ ;
15. \hspace{3em} $P_k, L \leftarrow$ TOPKUPDATEBOUNDPRUNE ;
16. \hspace{2em} $t \leftarrow t + 1$ ;
17. Procedure TOPKUPDATEBOUNDPRUNE() 
18. \hspace{1em} $S_{kth} \leftarrow k_{th}$ smallest $S_G(u)$ for $u \in P_k$ ;
19. \hspace{1em} node $n, S_{kth} \leftarrow k_{th}$ smallest $S_G(u)$ for $u \in P_k$ ;
20. forall $u \in P_k$ do
21. \hspace{2em} if $size(P_k) < k$ then
22. \hspace{3em} $P_k \leftarrow P_k + u$ ;
23. \hspace{2em} else if $\overline{S}_G(u) > S_{kth}(n)$ and $S_G(u) > S_{kth}$ then
24. \hspace{3em} $P_k \leftarrow P_k - n$ ;
25. \hspace{3em} $P_k \leftarrow P_k + u$ ;
26. \hspace{2em} else if $\overline{S}_G(u) < S_{kth}(n)$ then
27. \hspace{3em} $L \leftarrow L - u$ ;
28. \hspace{2em} return $P_k, L$ ;
The summation comprises of two parts. The first part is the summation of matching scores of decomposed star queries. The second part is the summation of matching scores of edges/paths among the candidates of query nodes.

### 3.3.6.2 Algorithm Flow

We show our general query with hierarchical inheritance relations (GQH) in Algorithm 2. Phase 1 for decomposing query is shown in Line 3. Phase 2 for star query is shown in Line 5–7. The candidate selection (in Line 9–13) continues propagating by uniform cost search from top candidates nodes and pruning with branch and bound until the top-$k$ candidate node set is found. The worst time complexity is $O(|V| \times |V_Q^S| + |V| \times |V_Q^U|^{k_s})$, where $|V_Q^S|$ is the maximum number of specific nodes for each query node in a query graph, and $k_s$ is the number of top-$k_s$ candidate results from each star query result. In our experiment, $k_s \in [k, 2k]$ is a good trade-off for efficiency and effectiveness. $|V|$ is the number of nodes in $G$. With the pruning of potential unmatched nodes for phase 2 and phase 3, the worst time complexity is reduced to $O(M \times |V_Q^S| + N \times |V_Q^U|^{k_s})$, where $M$ and $N$ are the numbers of visited nodes with type $\tau$ for phase 2 and phase 3, respectively.

### 3.3.6.3 Candidate Selections with Branch and Bound Pruning

The output of a star query graph is top-$k_s$ candidate nodes for each query node. The problem is how to efficiently connect the candidate nodes of star query results and pick the top-$k$ answers. If all candidate nodes are explored for each candidate combination, the time complexity would be exponential. We consider the branch and bound pruning technique [31] while traversing among these candidate nodes. To ensure the best quality of candidate selections, we sort each top-$k_s$ result of the star query in Phase 2 in a non-descending order in separate lists. Then we search through each list from the top to do a uniform cost search and construct a search tree. Each path along the root to the leaf node is a matched candidate set for query nodes. While
Algorithm 2: Top-k general query (GQH)

**Data:** Data graph $G(V, E, L_v, H_e)$, Query Graph $Q(V^S_Q, \tau)$, Matching final
number $k$

**Result:** Top-$k$ matched candidate sets $Mt$

1. Initialize top-$k$ matched candidate sets $Mt \leftarrow \phi$ (size $k$) ;
2. Initialize star query result list $stResList \leftarrow \phi$ ;
3. Star query graph set $StarGraphSet \leftarrow$ Query Graph $Q$ ;
4. $k_s \leftarrow [k, 2k]$ ;
5. **forall** $starGraph \in StarGraphSet$ **do**
6.     Top-$k_s$ candidate result $starCand \leftarrow$ SQH $(G, starCand, k)$ with
     Algorithm 1 ;
7.     $stResList \leftarrow stResList + starCand$ ;
**end**
8. $i \leftarrow 0$ ;
9. **while** $i < \text{len}(stResList) - 1$ **do**
10.     Traverse from $stResList[i] \rightarrow stResList[i + 1]$ ;
11.     Pruning nodes and path with bounds until top-$k$ candidate node sets are
     found ;
12.     $i \leftarrow i + 1$ ;
**end**

searching from root to leaf, we check the aggregated matching scores, lower and upper
bounds along the path. Assume there are top-$k$ candidate node sets with the smallest
lower bound score $F_{kh}$, by searching the next candidate node and getting its upper
bound lower than $F_{kh}$, the node candidate and all the nodes of its subtree can be
pruned.

### 3.4 Distributed Implementation

To support large information networks, we implement our graph query algorithm
in the framework GraphX, which is a distributed graph analytics platform built on
Apache Spark [170]. Figure 3.4 shows our architecture of distributed graph query
system. We define a global data structure, Global Vertex State Table (GT) for each
vertex stored in the Spark RDD data structure. GT is a user-defined class type which
can store the following hash mapping for each anchor node $v^a$: node type $\tau$, shortest
Figure 3.3: An example of candidate selection

Figure 3.4: Distributed graph query architecture

distance $sd$, hierarchical level difference $hd$, node closeness score $r$, closeness score lower bound $r_{\text{lb}}$, closeness score upper bound $r_{\text{ub}}$, etc. GT values are updated in each iteration of propagation to efficiently decide the bounds for effective pruning of many useless node propagations.
3.5 Experimental Evaluation

The experiments are designed to answer the questions as follows: (1) **Effectiveness**: How is the quality of our query algorithm for hierarchical query graph or mixed query graph? How is the query with hierarchical inheritance relations compared with state-of-the-art methods? (2) **Efficiency**: How is the efficiency and scalability of our algorithm on one machine and multiple machines?

3.5.1 Datasets

We use synthetic data graph, Cisco product data graph, and extended DBLP data graph. Table 3.1 shows the data statistics for our experiments. (1) Synthetic data graph: we randomly generate data graph and create 7 types of nodes. There are 2 attaching node types, 2 inherited node types and 3 other node types. (2) Cisco data graph: we extract the data from its official and related support websites about devices and device properties, etc. The constructed graph schema is shown in Figure 3.1(a). “Vulnerability” and “Technology” are the attaching node types. “Product” is the inherited node type. (3) Extended DBLP data graph: it is the DBLP database [97] extending the topics extracted from lists of computer science conferences and journal websites. “Topic” is the attaching node type that is inherited among the conference/journal, paper and people node types. In our experiments, we use $\alpha = 0.8$

| Dataset          | $|V|$   | $|E|$    | Avg. degree | No. of Vertex Types (Attaching + Inherited + Other) |
|------------------|--------|---------|-------------|--------------------------------------------------|
| Synthetic Graph  | 10M    | 6.54M   | 10          | 2+2+3                                            |
| Cisco Product    | 111347 | 666992  | 12          | 2+1+4                                            |
| Extended DBLP    | 1.28M  | 35.1M   | 58          | 1+2+9                                            |

(1) Synthetic data graph: we randomly generate data graph and create 7 types of nodes. There are 2 attaching node types, 2 inherited node types and 3 other node types. (2) Cisco data graph: we extract the data from its official and related support websites about devices and device properties, etc. The constructed graph schema is shown in Figure 3.1(a). “Vulnerability” and “Technology” are the attaching node types. “Product” is the inherited node type. (3) Extended DBLP data graph: it is the DBLP database [97] extending the topics extracted from lists of computer science conferences and journal websites. “Topic” is the attaching node type that is inherited among the conference/journal, paper and people node types. In our experiments, we use $\alpha = 0.8$
for the big decreasing degree of exponential function. Moreover, $\beta = 0.5$ is used for the half of the importance of the shortest distance to demonstrate our evaluation. These evaluated could be adjusted by users for their preferences.

### 3.5.2 Quality of Graph Query

As mentioned earlier, a query graph can be classified as a hierarchical, mixed hierarchical or non-hierarchical query graph considering inheritance relations, and a star query or general query graph based on query node numbers. We show the results of hierarchical star query graphs and mixed general query graphs here. In each real dataset, one star query example and non-star query example results are shown in Figure 3.6. Figure 3.6(a) shows the hierarchical star query with all specific nodes as attaching nodes and the query node as an inherited node, and the top-5 query results are found in Cisco data. As seen in the results, different inherited versions of Cisco WebEx meeting server products are queried with higher matching scores. Figure 3.6(b) displays different authors with publication papers in a journal and working on the same topic, which is verified to be reasonable online. As the more complex non-star queries shown in Figure 3.6(c) and 3.6(d) with each top-1 result, our algorithm GQH can also provide the most relevant query answers.

### 3.5.3 Comparisons of Query Quality

Existing state-of-the-art algorithms for graph queries generally only consider the shortest paths and neighbor propagation. Our paper proposes the query with hierarchical inheritance relations improving the recent query algorithm from Jin et al. [78] (GStar Query). NeMa [83] is a classical method for neighborhood-based query. Therefore, we compare the effectiveness of our query algorithm with the graph query based on GStar query and NeMa query algorithms.
3.5.3.1 Overall Comparison of Query Quality

We compare GQH with GStar and NeMa Query algorithm by comprehensive queries examples on Cisco data and Extended Dblp data. We design a metric called inheritance coverage ratio (ICR) to measure the overall quality. ICR is the coverage ratio of matched nodes with inheritance relations over the total matched nodes in $[0, 1]$. For example, given a query graph, a user wants to find a top-5 result of matched nodes. If there are 3 nodes that in the hierarchical relations, then $ICR = 0.6$. The higher the value, the more matched nodes with inherited relations would be found out. We create 100 random query graphs to obtain the top-2, top-5 and top-10 results, and then calculate the ICR results. We show the average ICR results on the three datasets. It shows the GQH algorithm has better ICR results on all the cases on the three datasets than GStar and NeMA, in which the query quality has been improved to different extents comprehensively.

3.5.3.2 An Example of Query Quality Comparison with GStar

We compare it with our algorithm GQH based on the example of the query in Figure 3.6(a) and show the result. Table 3.3 shows top-5 results of comparison with GStar’s Query algorithm. GQH shows the possible “Cisco WebEx meeting server version” inheritances as more potential candidates than GStar’s query algorithm, with three different numbers of matched candidates. This is because GStar’s query algorithm only considers the node types and shortest distances as metrics for ranking.

3.5.3.3 An Example of Query Quality Comparison with NeMa

NeMa in [83] uses nodes’ label and neighborhood similarity in small hops to find the top matched subgraphs. We compare the query quality with our algorithm GHQ for the query in Figure 3.6(a), and shows the result in Table 3.4. It shows top-5 results of comparison with GStar Query algorithm. NeMa uses matching cost which measures the cost of matched subgraphs with the query graph. The smaller the
cost, the better the matching. “—” indicates no matching result is found, only top-3 results are returned. It also does not return the hierarchical “Cisco WebEx meeting server” answers. This is because it limits the maximum hops of its visits and does not consider the hierarchical inheritance, which leads to a smaller structural difference but fewer potential matches.

Table 3.2: The overall query quality comparisons on the three datasets

<table>
<thead>
<tr>
<th></th>
<th>Synthetic</th>
<th>Cisco</th>
<th>Dblp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GStar</td>
<td>NeMa</td>
<td>GQH</td>
</tr>
<tr>
<td>Top-2</td>
<td>0.323</td>
<td>0.389</td>
<td>0.467</td>
</tr>
<tr>
<td>Top-5</td>
<td>0.415</td>
<td>0.432</td>
<td>0.533</td>
</tr>
<tr>
<td>Top-10</td>
<td>0.435</td>
<td>0.439</td>
<td>0.545</td>
</tr>
</tbody>
</table>

Table 3.3: One query result of GQH and GStar in the Cisco dataset

<table>
<thead>
<tr>
<th>Rank</th>
<th>Node</th>
<th>Score</th>
<th>Node</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cisco WebEx meetings server versions</td>
<td>1.7906</td>
<td>Cisco WebEx meetings server versions</td>
<td>1.7186</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td></td>
<td>1.x</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cisco WebEx meetings server versions</td>
<td>1.7100</td>
<td>Cisco WebEx meetings server versions</td>
<td>1.7186</td>
</tr>
<tr>
<td></td>
<td>1.x</td>
<td></td>
<td>2.x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cisco WebEx meetings server versions</td>
<td>1.7100</td>
<td>Easy vpn</td>
<td>1.7186</td>
</tr>
<tr>
<td></td>
<td>2.x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Cisco 12000 series spa interface processors running Cisco ios software</td>
<td>1.7015</td>
<td>Cisco unified ip phone</td>
<td>1.7015</td>
</tr>
<tr>
<td>5</td>
<td>Cisco xr 12000 series engine 3 line cards</td>
<td>1.7015</td>
<td>Catalyst 6000 supervisor module</td>
<td>1.7015</td>
</tr>
</tbody>
</table>
Table 3.4: One query result of GQH and NeMa in the Cisco dataset

<table>
<thead>
<tr>
<th>Rank</th>
<th>Node</th>
<th>Score</th>
<th>Node</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cisco WebEx meetings server versions 0.2</td>
<td>1.7906</td>
<td>Cisco WebEx meetings server version 0.2</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>Cisco WebEx meetings server versions 1.x</td>
<td>1.7100</td>
<td>Cisco ASA Series show running-config prior to 7.2.1</td>
<td>2.95833</td>
</tr>
<tr>
<td>3</td>
<td>Cisco WebEx meetings server versions 2.x</td>
<td>1.7100</td>
<td>Cisco ASA Series show running-config between 7.2.1 and 8.4</td>
<td>2.95833</td>
</tr>
<tr>
<td>4</td>
<td>Cisco 12000 series spa interface processors running Cisco ios software</td>
<td>1.7015</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>Cisco xr 12000 series engine 3 line cards</td>
<td>1.7015</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
(Vulnerability) (Technology) (Product)

$V_1$: Cisco asa software ikev1 and ikev2 buffer overflow vulnerability
$T_1$: Voice - communications manager additional apps and plugins

(a) Hierarchical star query graph in Cisco – Query the products with Vulnerability $V_1$ and used Technology $T_1$, and its top-5 results in order.

(b) Mixed star query graph in Dblp – Query authors that work on Topic $T_1$ and cooperate with Person $PP_1$, and its top-5 results in order.

Figure 3.5: Star graph queries and their results
Figure 3.6: Non-star graph queries and their results (a) Mixed general query graph in Cisco – Query a product that has Vulnerability $V_1$ and $V_2$, another product that belongs to a workgroup $W_1$, and their common technology also used in a product $P_1$, and the top-1 result. (b) Mixed general query graph in Dblp – Query an author that works on Topic $T_1$ and $T_2$, and cooperates with another query person that works on Topic $T_3$, and published a paper coauthored with Person $P_{p_1}$ in year $D_{i_1}$, and its top-1 result.

### 3.5.4 Efficiency of Graph Query

Our GQH algorithm mainly focuses on improving the quality of query, and we also use a different implementing platform and programming language from GStar and NeMa algorithms, thus comparing the running time directly with them is not meaningful. Therefore, we test the efficiency of our GQH algorithm itself. We evaluate the efficiency with different top-$k$ values, query graph size, and data graph size. For each different test, we keep one testing parameter varied and the other unchanged.
Each experiment is repeated 20 times and we obtain the average runtime with different parameters.

**Varying $k$:** To check how our algorithm scales with different querying $k$, we examine the average runtime for different top-$k$ values from 1, 2, 5 to 30 in Figure 3.7(a)–(c). Three different query sizes $[2, 1]$, $[4, 2]$, and $[6, 3]$ are fixed. It shows the runtime is basically sublinear no matter the $k$ value. This is because the complexity degrees of graphs lead to more than designated top-$k$ answered before the termination of iterations. We only fetch the top-$k$ candidates from all the obtained candidates.

**Varying Query Graph Size** To check how our algorithm scales with different query graph sizes, we examine the average runtime for different query sizes. The query size is defined as a tuple (specific node number, query node number). We select (2, 1), (4, 2) to (10, 10) shown in Figure 3.7(d)–(f) with top-$k$ value 2, 5, and 10 used. It shows that the running time is basically sublinear with the increasing of query size.

**Varying Data Graph size:** We test the query time with varying data graph size. We randomly and accumulatively extract subgraphs from the original data graph for different node numbers, covering 10%, 20%, 50%, 80%, and 100%. We measure 3 different query sizes in this scene to obtain the top-20 query time for different graph data size. As shown in Figure 3.7(g)–(i), the query time also increases sublinearly with the increasing of graph data size.

**Scalability on Multiple Machines:** To test the scalability of our GQH algorithm on multiple machines, we test it on Google Cloud Platform to see the average runtime trends with different worker machines deployed. We use 3 different query size, with one master and increase worker machines from 2, 4, to 32. Figure 3.7(j) shows that the running average time decreases sublinearly with the increasing number of workers.
Figure 3.7: Efficiency and scalability on three datasets
3.6 Related Work

There exist several classification categories for graph query. Based on user inputs, it can be classified as keyword query and structured query [81, 60, 83, 138, 48]. Based on query answers, it includes exact match and inexact match [82, 92, 117]. Based on matching techniques, it mainly contains indexing-based query and graph-traversal-based query for distance, neighbor and random walk [49, 190, 36, 134, 167, 189]. Our algorithm focuses on the top-\(k\) inexact match for structured graph queries with hierarchical inheritance relations.

3.6.1 Structured Graph Query:

Various techniques have been proposed for structured graph query. Recent work allows users to express their own input query as a structured query graph and do the graph traversal based on node and path similarity for matching. For example, NeMa [83] and SLQ [174] consider different similarity transformations for node to match the query graph and subgraphs in a data graph. Su et al. [148] consider the graph query based on user relevances to further improve the query quality. Some works [138, 38] consider the multiple attributes of nodes for graph query. Jin et al. [77] propose a specified ranking function for structured graph query with specific nodes to find answer nodes. Most of them use indexing which takes large spaces, or graph traversal with only two dimensions of node and edge similarities. We consider one more dimension of hierarchical inheritance relations for effective queries.

3.6.2 Top-\(k\) Graph Query:

Top-\(k\) graph query tries to obtain top-\(k\) matched answers for the graph query. The common practice for top-\(k\) search is to use threshold algorithms to find the top matches by traversing a sorted node/edge list [43, 173, 165, 192]. They require precomputed and sorted lists to derive the bounds. Recent top-\(k\) query has been studied in [28, 57]. Yang et al. [173] consider the STAR-query structure and top-\(k\)
ranked join for a general graph query, but the matches are limited to answer subgraphs with paths of bounded length. Jin et al. [77] consider the shortest path propagation with a specified matching score function for structured graph query to find answer nodes. Our algorithm extends the work and considers top-$k$ general graph with an efficient ranking score and bounded-based solution without the limitation of path lengths for hierarchical relation inheritance.

3.7 Conclusion

We consider an additional dimension of hierarchical inheritance relations on real-world heterogeneous information networks for graph query. The problem is reformulated with hierarchical inheritance relations, and we propose a graph query algorithm based on that for star-query and general graph query. With the bounding-based techniques, our algorithm can effectively capture hierarchical inheritance relations on information networks for better query answers and competitive performances are also achieved.
CHAPTER 4

REAL-TIME CONFIGURATION ADAPTATION FOR VIDEO ANALYTICS THROUGH OBJECT TRACKING

4.1 Introduction

The proliferation of cameras deployed in many enterprises and cities like New York City, London and Beijing [1, 47, 180] drives the demand for video analytics. Object detection and tracking as the common parts of video analytics focuses on detecting and tracking objects from video streams. Many applications in traffic control, business intelligence, action/event analysis, human-machine interaction and VR/AR are built on top of object detection and tracking [75, 23]. Video analytics usually involves a pipeline of video processing modules. For example, the classical pipeline was shown in Figure 4.4. The video analytics involves the decoding of the video from cameras, then we may change to a different frame rate and resolution and apply to an inference model for object detection and tracking. Recently, object detection and tracking relies on deep neural network (DNN) models for more accurate inference. Each component is called a knob, which corresponds to many different values. For example, there are many different frame rate, resolutions, or pose estimation models. The selection of a combination of specific knob value is called a configuration. Frame rate and resolution are the two important knobs of the configuration. In this work, we consider the frame rate and resolution as a configuration.

Different configurations would lead to different accuracy and resource consumption. The selection of a configuration impacts the accuracy and resource consumption of object detection and tracking. A configuration with high frame rate and resolution
is considered as an expensive configuration, and a configuration with low frame rate and resolution is considered as a cheap configuration. For example, an expensive configuration with 25 fps and 1080p generally leads to a higher accuracy than a cheap configuration with 1 fps and 240p, but consumes more resources. The “best” configuration is the configuration that achieves a desired accuracy with the least resource consumption. Also different applications have different requirements for the accuracy or resource consumption. For example, traffic vehicle counting [183] can function well with moderate accuracy while amber alert detection [182] generally requires very high accuracy. Therefore, it plays an important part for user experience and requirements with different applications in selecting the “best” configuration for adaptation during the video analytics process.

The state-of-the-art configuration adaptation research for the video analytics are through profiling approaches. The profiling is to obtain the accuracy and resource requirement for all the configurations in the configuration space or parts of the configuration space. The configuration adaptation approach falls into two classes: one-time profiling and periodical profiling. One-time profiling [59, 181, 184] aims to profile all the configurations only once during the beginning of a video (e.g., 10 seconds), and then chooses the best configuration above an accuracy requirement for the rest of the video. However, it uses a fixed configuration for the whole video and neglects the intrinsic dynamics of video contents. For the periodic profiling [179, 130, 75], a video is divided into a fixed period of intervals. In each interval, we profile configurations for the first several amount of frames a policy that determines the configuration for ev-
very interval of a video through profiling the first few frames of the interval. However, profiling periodically costs compute resources and incurs processing time.

Video contents contains the temporal and spatial characteristics. For object tracking, lots of work utilizes these characteristics to model the similarity scores between tracklet and object, or model neural network architectures [154, 171, 23, 185]. Due to the temporal and spatial characteristics, objects keep the same or similar movement consistency in a short period of time. Therefore it could be quantified with the object movement in this period. An object moving fast usually prefers expensive configurations and the slow object could lead to select cheap configurations.

In this work, we capture the object movement from past video analytics results to guide the selection of frame rate and resolution to adapt configurations over time. Leveraging this, a machine learning-based classification method, MOTrack, is utilized to obtain the relationship between object movement and the best configuration. We obtain the estimated object movement and corresponding configurations as labeled training data instances to automatically learn the mapping between them. Through extensive experiments on traffic tracking and pose estimation applications on large video datasets, we demonstrate the superiority and effectiveness of MOTrack. Compared with two state-of-the-art configuration adaptation approaches, MOTrack could achieve the accuracy threshold goal with 8–17x less computation resource.

4.2 Preliminary

We begin with some background on the configuration adaptation of DNN-based video analytics, and the metrics used for video analytics.

4.2.1 Video analytics Pipeline for Configuration Adaptation

Considering various neural network detection models with similar performances, we show a generalized video analytics framework with adaptive configuration to ex-
plore the dynamic update of configuration with frame sampling and resizing in video analytics. As shown in Figure 4.2, video frames are fed into a module of video analytics system with a general configuration adaptation. The video frames are first fed into a module of resizing and selecting(sampling) frames according to a configuration controller and then fed into a DNN model for video inference. The controller iteratively guides the selection of frame sampling rate and resolution, and then video inference is done over and over again.

Figure 4.2: A video analytics module with an adaptive configuration controller for video inference

4.2.2 Metrics for Video Analytics

**Configuration** A configuration is a combination of jumping frame interval and resolution value. The configuration space comprises all the possible combinations of these knob values.

**Quality metrics** In the video analytics, different video analytics applications have different measurements of inference quality (accuracy). It measures how accurate the detected object or action is deviated from the ground truth. For example, for object detection application, the bounding box is utilized to measure the performance of each detected object. We thus calculate the normalized intersection over union of
detected bounding box to the ground truth bounding box as the accuracy. Some other action detection applications such as pose estimation, the object keypoint similarity (OKS) [137] is used to calculate the accuracy.

**Resource cost measurement** DNN model dominantly uses GPU to process most of the workloads recently. We use the average video processing time per frame in second unit on the GPU (with 100% utilization) as the measurement of the computation resource cost named as second per frame (SPF) similar to [75]. The higher video processing time per frame, more resources are consumed.

**Ground truth** It is expensive to label the ground truth for every frame of each video and also it is impractical to obtain the ground truth for online video analytics. Same as other video analytics references [75, 179], the most expensive configuration (also called golden configuration) with the lowest jumping frame interval and highest resolution is used as the ground truth with highest accuracy.

### 4.3 Motivation

Object detection and tracking in video analytics is to detect object positions and track objects for every frame. Traditionally, every frame is expected to be processed with an expensive DNN model for detection. However, when an object moves slowly in a short period of the video, the frames in that period could be skipped for an expensive DNN model. We just need to estimate the position of the object in the skipped frames from its previous location and momentum. Similarly, a low resolution can be adopted for a slow-moving object. This could save huge computation resources while still maintaining the required accuracy. Therefore, the object movement could help make a decision for the needed frame rate and resolution.

The object movement could be quantified with the object moving velocity, which could be measured based on the distances of detected object keypoints between frames over time (we call it keypoint movement velocity). Given a minimum accuracy re-
quirement, if the movement velocity is high in a short period of the video, the high frame rate/resolution would be needed to satisfy the accuracy requirement, and vice versa.

Figure 4.3 shows the (normalized) keypoint movement speed (velocity magnitude) for every 1 second interval, and the necessary frame rate and resolution in a 4-minute dancing video clip in the pose estimation application [8]. To simplify and show the evaluation, we assume there is only one high frame rate/resolution and one low frame rate/resolution available. We calculate the object moving speed for every second interval. Given a minimum accuracy requirement for the video analytics, a frame rate/resolution will be selected for each interval. We could see that when the movement speed increases to a high value in periods of around second 15, 30, 110, 130, 180, and 230 highlighted with grey backgrounds, the high frame rates/resolutions are necessary in those periods in order to achieve the accuracy requirement. Conversely, we find most of the time the object moves at a low speed and thus a low frame rate/resolution is needed. It shows the strong correlation between the movement speed and the frame rate or resolution needed.

4.4 System Architecture

We develop a video analytics system based on our proposed configuration adaption algorithm through movement prediction. The overall architecture is as follows.

Our model utilize the machine-learning based prediction about the configuration. In the offline stage, we collect the training dataset which are the movement features and the needed configuration (label). For each video stream, we run the DNN model on all the configuration space (which are the 25 frame rate and 5 resolutions) and obtain the corresponding accuracy and processing time per frame. Then we calculate the object movement features for each frame as the current starting point, and the
needed configuration for satisfying the movement (which we will talk in detail in Section

4.5 Movement Features and Prediction Model

We propose to quantify the correlation of target objects in different periods of a video stream with some movement features. A movement feature of one target object describes how the keypoints of the object move. In addition to the movement features describing each keypoint, we also consider some dense features for the dynamics of the whole object. Then, we develop a model that learns how to dynamically adjust the configurations based on the proposed features.

We utilize the movement information of target objects to dynamically decide which frames should be analyzed in detail and which frames can be skipped. The physical movement of an object is inertial, which leads to the velocity and acceleration of the object. Then for skipped frames, we estimate the location of the target object.
We first introduce some key concepts and notations of our algorithm. Our algorithm tries to find small periods of a video for processing with different resolutions and frame rates. We attach a timestamp \( t_i \) to each frame \( i \) considering a real-time video playout rate of 25 frames/second. It can be derived easily by counting how many frames are skipped from the playout rate of the video. Similarly, our algorithm also memorizes the resolution of each frame. The width and height of a frame \( i \) are represented with \( W_i \) and \( H_i \) respectively. Therefore, a sequence of frames can be represented as follows:

\[
[(t_1, W_1, H_1), \ldots, (t_n, W_n, H_n)]
\] (4.1)
Then the position $p(i)$ of one point with coordinate $(x, y)$ at frame $i$ is represented with the following vector (normalized):

$$
\bar{p}(i) = \left[ \frac{x(i)}{W(i)}, \frac{y(i)}{H(i)} \right]^T \quad (4.2)
$$

where $x(i)$ and $y(i)$ indicate X and Y coordinates of the point in frame $i$. When there are multiple keypoints such as $a$, $b$, $c$ in one object, the corresponding positions are denoted as $p_a(i)$, $p_b(i)$ and $p_c(i)$, respectively.

### 4.5.1 Movement Features

We extract the movement information of each one object into a set of features from its past video analytics results. The features are described below.

#### 4.5.1.1 Keypoint Movement Velocity

An object in a frame can be represented as a set of keypoints detected, such as 4 bounding box corner points in an object. This keypoint movement velocity feature captures how a keypoint moves across frames in the screen space. We measure the pixel velocity of a point $p$ using consecutive frames. Assume that we have two consecutive frame $i-1$ and $i$ taking at time $t_{i-1}$ and $t_i$ respectively. Then, the velocity of this point at frame $i$ can be quantified as:

$$
\vec{v}_f(i) = \frac{\bar{p}(i) - \bar{p}(i-1)}{t_i - t_{i-1}} \quad (4.3)
$$

where $\bar{p}(i-1)$ and $\bar{p}(i)$ indicate the normalized position of the object at frame $i-1$ and frame $i$ in the screen space. Therefore, for an object with $n$ keypoints, we derive the keypoint velocity feature of the object on frame $i$ as a vector of the $\vec{v}_f$ for all keypoints as follows, where $\vec{v}_f^k(i)$ represents the velocity of keypoint $k$ at frame $i$.

$$
\mathbf{V}_f(i) = [\vec{v}_f^1(i), \cdots, \vec{v}_f^k(i), \cdots, \vec{v}_f^n(i)] \quad (4.4)
$$
4.5.1.2 Object Movement Velocity

To capture more fine-grained features, we propose another object movement feature to complement previous movement features. Previous keypoint movement velocity feature only captures the corner’s movement of an object across frames. It is based on the robust optical flow [84] calculated only with pixel values of detected objects. It could recover the object’s motion at each pixel and apparent velocities of movement of brightness pattern in a frame.

We capture the optical flow velocities of all the pixel points as our object movement velocity. To capture each pixel point’s optical flow, we consider two consecutive frames $i - 1$ and $i$ taking at time $t_{i-1}$ and $t_i$. For one pixel $k$ at location $(x, y, t)$ with its brightness $I(x, y, t)$ from frame $i - 1$ to $(x + \Delta x, y + \Delta y, t + \Delta t)$ in frame $i$ with its brightness $I(x + \Delta x, y + \Delta y, t + \Delta t)$, its brightness is assumed to keep constancy and transformed with the Taylor series, we obtain this equation,

$$\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v = -\frac{\partial I}{\partial t}$$  \hspace{1cm} (4.5)

Where $u$ and $v$ are the optical flow of point $(x, y)$ along $x$ and $y$ direction. $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the derivatives of the image intensity at $(x, y, t)$ in the corresponding directions.

To get $u$ and $v$ with the one equation, we extend with more points over a small neighborhood by introducing nearby pixels within a window and solve the multiple equations with the least-squares principle [84]. Then, the optical flow velocity for that pixel $k$ inside an object in frame $i$ is $\vec{O}^k(i) = (u, v)^k(i)$.

Therefore, for one object with $n$ pixels in the detected bounding box, we have movement feature $O(i)$ at frame $i$,

$$O(i) = [\vec{O}^1(i), \vec{O}^2(i), ..., \vec{O}^k(i), ..., \vec{O}^n(i)]$$  \hspace{1cm} (4.6)
4.5.1.3 Keypoint Relative Movement Velocity

A highly relative movement among the keypoints of an object usually indicates a drastic change. For some applications containing multiple correlated keypoints, such as pose estimation with 17 keypoints (e.g., arms, hands), the relative velocity can be adopted among different keypoints. We should use a higher resolution to check out what happens to the object or even double-check whether there is something wrong with the previous tracking result. This feature captures the change by relative keypoint movements.

The relative velocity feature $V_r$ of an object at the frame $i$ is a triangle matrix of each pair of the keypoints among $n$ keypoints as the following equation.

$$V_r(i) = \begin{bmatrix}
0 & \vec{v}_r^{1,2}(i) & \vec{v}_r^{1,3}(i) & \ldots & \vec{v}_r^{1,n}(i) \\
0 & 0 & \vec{v}_r^{2,3}(i) & \ldots & \vec{v}_r^{2,n}(i) \\
\vdots \\
0 & 0 & 0 & \ldots & 0
\end{bmatrix}$$ (4.7)

where $\vec{v}_r^{p,q}(i)$, $p, q \in [1, n]$ indicates the relative velocity from point $p$ compared to point $q$ at frame $i$. It is formulated as the following equation.

$$\vec{v}_r^{a,b}(i) = \frac{\vec{r}_{a,b}(i) - \vec{r}_{a,b}(i-1)}{t_i - t_{i-1}}$$ (4.8)

where $\vec{r}_{a,b}(i) = \vec{p}_a(i) - \vec{p}_b(i)$ indicates the relative vector distance from point $a$ to point $b$ at frame $i$. $\vec{p}_a(i)$ and $\vec{p}_b(i)$ are the coordinates of two points $a$ and $b$ on the same object at frame $i$.

For pose estimation application, we only consider the pairs in relative velocity features $V_r$ that have high impacts on relative movements, such as the wrists to shoulders, and the ankles to hips.
4.5.1.4 Object Size Change Speed

In addition to object movement velocity features, we also capture the whole object morphing by the object size change. The object size in an image is a great indicator for shape morphing and Z-axis movement of the real-world object. The change ratio of the object size represents how fast the morphing or Z-axis movement happens.

The size can be directly acquired from the video analytics result. For simplicity, we use the size of the bounding box of the target object to approximate the object size. By representing the size of the object in frame $i$ with $S_i$, we can formulate the object size change speed feature $\text{SC}(i)$ at frame $i$ as follows.

$$\text{SC}(i) = \frac{S_i - S_{i-1}}{t_i - t_{i-1}}$$

(4.9)

When the bounding box is used, the size $S_i$ can be computed as follows where $x_{\text{max}} = \max_k \vec{p}_k x$ represents the maximum X-coordinate among all the keypoints of the object.

$$S(i) = \frac{(x_{\text{max}} - x_{\text{min}}) \cdot (y_{\text{max}} - y_{\text{min}})}{W_i H_i}$$

(4.10)

Similarly, $x_{\text{min}}$, $y_{\text{min}}$ and $y_{\text{max}}$ mean the minimum X-coordinate, minimum Y-coordinate, maximum Y-coordinate respectively.

4.5.2 Feature Smoothing

The feature estimation using two frames is not reliable enough. It can be affected by some random noise or a single failure in tracking. Since physical movement in real-world is inertial, we use the exponential moving average to smooth the features as the final estimation. For a feature $\mathbf{F}$, we derive its final value $\hat{\mathbf{F}}(i)$ for frame $i$ based on its instantaneous estimation $\mathbf{F}(i)$ and the smoothed value on the previous $\hat{\mathbf{F}}(i - 1)$. We formulate this procedure as the following equation,

$$\hat{\mathbf{F}}(i) = \alpha \mathbf{F}(i) + (1 - \alpha) \hat{\mathbf{F}}(i - 1)$$

(4.11)
where $F$ can be $V_f$, $V_r$, $SC$ or $O$, and $\alpha$ is the smooth factor in $[0, 1]$. Therefore, we use $X = [\hat{V}_f(i), \hat{V}_r(i), \hat{SC}(i), \hat{O}(i)]$ as our feature.

### 4.5.3 Estimate the Results of Skipped Frames

We estimate the positions of a target object in the skipped frames through their previous locations and movement information. As shown in Figure 4.5, a frame $j$’s object position is assumed to be inferred with our DNN detector. The positions of the skipped frames in the predicted interval are then estimated as follows. Assume the skipped frame number is $S$, the keypoint $p$ in the skipped frames, $\vec{p}(j+1), \vec{p}(j+2), \ldots, \vec{p}(j+S)$, could be estimated with the movement velocity vector $\hat{F}(j)$. If the current frame $j$ has one object position with a vector coordinate $\vec{p}(j)$ in $x$ and $y$ dimension, and the estimated movement velocity for this position is $\hat{F}(j)$, then that position coordinate in the next skipped frame $k$ is estimated as:

$$\vec{p}(k) = \vec{p}(k-1) + \frac{\hat{F}(j)}{S}$$  \quad (4.12)

where $k = j + 1, j + 2, \ldots, j + S$.

![Figure 4.5: Object detection result estimation for skipped frames](image)

Figure 4.5: Object detection result estimation for skipped frames
4.5.4 Model for Configuration Prediction

We train a model that predicts which configurations are used for future frames. The frame rate determines the time interval until the next frame. Therefore, the training data is a set of features and configuration pairs generated from many video streams. The ground truth configuration should be the one that requires the lowest computing resources but still satisfies a certain accuracy requirement.

We model this problem as a multiclass and multi-output classification problem. We adopt the random forest classification model [101] to do the prediction. Random forest classification is an efficient algorithm which minimizes both bias and variance on an ensemble of decision trees. In our prediction model, frame rate and resolution are two targets, which are learned individually, similar to one target inside a classification model.

4.6 Training Data Generation

To train a model with a random forest model, we have to obtain the training data ground truth–configuration for each feature, which is not practical to manually label it.

Finding the best configuration for video analytics could be modeled as a global optimization problem with constraints. The accuracy requirement for an application is denoted as $A_{\text{min}}$. A frame rate and resolution pair is denoted as $(fr, res)$. The latency in one time interval $i$ based on $(fr_i, res_i)$ is denoted as $L(fr_i, res_i)$. The number of time intervals is $P$. Then, given $A_{\text{min}}$, the goal is to find a set of $(fr_i, res_i)$ pairs to minimize the final processing latency $L$, while satisfying that interval accuracy $Acc_i > A_{\text{min}}$. Mathematically,

That is we construct a set of frame rates and resolution $[(fr_1, res_1), (fr_2, res_2), ..., (fr_i, res_i), ..., (fr_p, res_p)]$, where $1 < i < P$. 

100
Due to the large space of frame rate, and the resolutions available, it is time-consuming to find the globally optimal configuration for each time interval. We propose a heuristic approach to determine the corresponding configuration for each time window in the object tracking to generate labeled data instances.

We use each frame as the starting frame of a predicting time interval to calculate the movement feature and then generate its corresponding configuration. The algorithm process for generating training data is as follows. We try all the frame rates from 25 to 1, and then we traverse all the resolutions from smallest to the highest to find the optimal resolution. In this way, the average accuracy is calculated to ensure above $A_{\text{min}}$ and the configuration with the least resource cost is also selected as the target configuration. To obtain the average accuracy, we use the estimated frame rate, resolution, and absolute on-frame movement velocity feature to obtain the ground truth configuration.

\textbf{Algorithm 3:} Deciding configuration starting from frame $j$

\begin{algorithm}
\textbf{Data:} Video streaming at frame $j$, minimum accuracy threshold $A_{\text{min}}$, object tracking outputs $OR_j$ up to $j$.
\textbf{Result:} Frame rate and resolution configuration $C_j$
\begin{algorithmic}[1]
\State Initialize frame rate $jr \leftarrow$ Maximum frame rate 25 ;
\State $R \leftarrow$ all the resolutions in the configuration space in ascending order ;
\State $C_j \leftarrow$ the most expensive configuration
\While{$jr \geq 1$}
\ForAll{$rs \in R$}
\State $Acc_{\text{curr}} \leftarrow$ calculate the current accuracy with movement feature under $rs, jr$ and $OR_j$ ;
\If{$Acc_{\text{curr}} \geq A_{\text{min}}$}
\State $C_j \leftarrow (jr, rs)$ // found ;
\State \textbf{return} $C_j$ ;
\EndIf
\EndFor
\State $jr \leftarrow jr - 1$ ;
\EndWhile
\State \textbf{return} $C_j$ ;
\end{algorithmic}
\end{algorithm}
4.7 Experimental evaluation

We evaluate our video analytics method, MOTrack, on two applications – pose estimation and traffic tracking, on one server with two Quadro RTX 5000 GPUs and compare it with two state-of-the-art configuration adaption methods.

4.7.1 Applications and Datasets

In this section, we show the applications and datasets used to test our algorithms in our experiment.

4.7.1.1 Applications

Pose estimation localizes anatomical keypoints or “parts” from the human body [8] based on the a model. Each object in a frame has 17 - 19 keypoints as the video analytics result.

Currently, the most effective way for traffic tracking and pose estimation is through deep neural network methods [149, 27, 150, 126, 119, 125, 87]. Recent advancements on pose estimation on neural network focus on top-down and bottom-up techniques. Top-down technique [149, 169, 27, 65, 45, 150, 126] is to detect the human body in the image first and then identify its keypoints. Bottom-up techniques [17, 119, 125, 87] directly detect the keypoints of a person from an image. Despite different deep neural network models, they have shown similar accuracies (around 5%-10% accuracy performance differences) [149]. Moreover, these techniques have the same commonality to detect all the person objects in all frames in video streams. This would require a significant amount of GPU resources for neural network inference. However, in many scenarios, videos have temporal characteristics and objects might change very slowly (e.g., the person object keeps static in the video for a few seconds). Furthermore, if the person object appears clearly and front in the camera, we only need to detect relatively large object. In this case, processing only 1 frame instead of each frame and resizing it from 960p to 480p, for example, would reduce the resource
demand by 120x with negligible accuracy decrease [75]. Frame sampling and resizing are two important knobs among the many possible knobs in pose estimation that can dramatically reduce the GPU resources cost with a little impact on accuracy. In our experiment, we use the OpenPose model [17, 112] as our Pose estimation model.

Traffic tracking localizes and tracks pedestrians and vehicles together indoors or outdoors from video streams based on detection and tracking. The video analytics result in each frame for each object is represented by a bounding box of that object.

Due to the recent development of deep neural network models and object detection with deep neural network models, object tracking has mainly dominated by the tracking-by-object paradigm [161, 44, 98]. The tracking-by-object diagram consists of two main parts. The first part is the object detection algorithm which gives the object detection results commonly in the form of the bounding box coordinates for each object. The second part is the data association algorithm to decide the newly detected objects are associated with the previous tracked object position. Recently, several algorithms are proposed to improve the accuracy and are reduced the time cost for the two parts [5, 15, 12, 166]. Simple online and realtime tracking (SORT) [12] propose an efficient and reliable frame-to-frame association approach, where Kalman Filter and Hungarian method are used to estimate the locations of the tracked object from last frame. This method achieves favorable performance at high frame rates. N. Wojke et al. [166] extended the SORT with a Deep association metric (Deep SORT). Deep SORT use a convolutional neural network trained on re-identification data set to extract the appearance information of the new detection and compare it with the previous tracked objects. DSLCF model from Hou et al. further extend the Deep SORT to overcome the low confidence track filtering for multiple object tracking, which further optimize the Deep Sort-based tracking model.
Considering the Deep Sort is the state-of-the-art object tracking model, in our experiment, we use the the Deep Sort-based model [12, 63] as our traffic tracking model.

4.7.1.2 Datasets

Existing video stream datasets do not have enough high resolution or length of videos available for video analytics. We create video stream datasets from Youtube.  

Traffic tracking dataset contains 60 video clips of surveillance indoors, and mixed pedestrians and vehicles on the street, in which objects move generally fast but most of the time are steady, with their moving speed difference between a consecutive second averaging 15 pixels/second (p/s). However, pose estimation dataset consists of 75 video clips of dancing, body conditioning, and other workouts, in which an object moves dramatically with its moving speed difference averaging 23 p/s between the second frames. Table 4.1 shows the dataset statistics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total length (sec)</th>
<th>Object no. (each video)</th>
<th>Keypoint no. (each object)</th>
<th>Speed difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic tracking</td>
<td>36k</td>
<td>Various</td>
<td>4</td>
<td>15 p/s</td>
</tr>
<tr>
<td>Pose estimation</td>
<td>45k</td>
<td>Single</td>
<td>17</td>
<td>23 p/s</td>
</tr>
</tbody>
</table>

To train a random forest model, we have to obtain the training data ground truth–configuration for each feature, which is not practical to label manually. We utilize a heuristic approach to find the corresponding configuration for each interval in video analytics to generate labeled data instances. We explore all the available frame rate and resolution configurations. Given a minimum accuracy threshold, we select the lowest frame rate and resolution configuration as the ground truth to achieve

---

1Datasets are public in: https://rb.gy/pofzby
the minimum processing time while maintaining the average accuracy above that
threshold in each estimated interval.

In our prediction model, we split the video dataset into a training dataset and
a test dataset for each application. Specifically, we use each one of the video clips
as the test dataset and the other video clips as the training dataset. 5-fold cross-
validation is utilized to obtain the best model and then applied it to the test dataset.
In our experiment, we adopt one minimum accuracy threshold of 0.92 to show our
results. The processing time shown in all the figures below is the average total time
for processing each one-second interval of video.

4.7.2 Metrics and Configurations

Traffic tracking uses the bounding box’s intersection over union to calculate the
accuracy [177]. Pose estimation accuracy is calculated with object keypoint similarity
metrics from COCO pose estimation dataset [137]. The ground truth for measuring
video analytics accuracy is based on the most expensive configuration with the highest
frame rate and resolution [75].

For the prediction model, we concern how accurate our prediction model predicts
the correct configuration, so we use the “accuracy” to measure the average prediction
accuracy for predicting the frame rate and resolution. There are 25 classes for 1
frame/second, 2 frames/second, ..., and 25 frames/second. The video resolutions
considered are: 1120x832, 960x720, 640x480, 480x352 and 320x240. In total, we have
125 classes in the large configuration space.

4.7.3 Impact of Features on Prediction Accuracy

Here we show the different features’ impact on the prediction accuracy. We eval-
uate the impact of each feature on the prediction accuracy by removing the features
one by one for each application. Table 4.2 shows the impact of different features on
the prediction accuracy. “All” means we use all proposed features for each applica-
tion. “x-” symbol ahead of each feature name means we remove this feature from “All”. It shows that our prediction method achieves an accuracy from 0.794 to 0.865 with all proposed features. When we remove one of the features, the accuracy has been degraded to a different extent. Each feature is indispensable and contributed in some way to predict the configuration.

Table 4.2: Impact of features on prediction accuracy

<table>
<thead>
<tr>
<th>Feature</th>
<th>Application Traffic tracking</th>
<th>Pose estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>0.865</td>
<td>0.794</td>
</tr>
<tr>
<td>x- Keypoint movement velocity</td>
<td>0.838</td>
<td>0.785</td>
</tr>
<tr>
<td>x- Keypoint relative velocity</td>
<td>N/A</td>
<td>0.783</td>
</tr>
<tr>
<td>x- Object movement velocity</td>
<td>0.803</td>
<td>0.766</td>
</tr>
<tr>
<td>x- Object size change</td>
<td>0.846</td>
<td>0.789</td>
</tr>
</tbody>
</table>

4.7.4 Video Analytics Performance

We show the overall video analytics performance and one-minute video clips performance of MOTrack, and compare it with two state-of-the-art configuration adaptation methods: one-time profiling and periodic profiling.

One-time profiling is operated only once at the beginning of video analytics [181]. The profiling interval $x$ is 10 sec. For periodic profiling [75], the profiling interval $t$ for each time window is 1 sec, and the time window is 4 sec. The same parameter settings are used for comparisons.

Figure 4.6(a) shows the accuracy and Figure 4.6(b) shows the processing time. From these results, MOTrack achieves 3%-12% higher accuracy with the same processing time compared to one-time profiling and 8–17x faster with the similar accuracy compared to periodic profiling.

To check the configuration adaptation effects in detail, we test on a one-minute video clip for each application and show the comparison results in Figure 4.7 and Figure 4.8.
Figure 4.6: Comparison of MOTrack with one-time profiling and periodic profiling on traffic tracking and pose estimation

Figure 4.7 shows the accuracy and processing time on traffic tracking. Compared to periodic profiling, we can see that MOTrack maintains a similar accuracy over time to periodic profiling, but much less processing time over time. This is because MOTrack has more frequent adaptation operations based on frames, but with an inexpensive adaption algorithm. Compared to one-time profiling, MOTrack maintains a much better accuracy over time and costs similar processing time.

For pose estimation on Figure 4.8, MOtrack also demonstrates the similar accuracy over time compared to periodic profiling and shows more frequent configuration adaptation over time due to the drastic changes of human pose movement. Meanwhile, MOTrack maintains a much better accuracy over time and also takes less processing time compared to one-time profiling.

Figure 4.9 shows how close our configuration adaption to the ground truth (configuration) on that video clip. The ground truth has the minimum processing time above the accuracy requirement. MOTrack obtains a very close result over time, with only an average of 5% more time than the ground truth.
Figure 4.7: Accuracy and processing time for one video clip on traffic tracking

Figure 4.8: Accuracy and processing time for one video clip on pose estimation

4.7.5 Time for Configuration Adaptation

Here we compare MOTrack’s configuration adaptation time with periodic profiling’s on the test datasets. MOTrack adopts machine learning-based prediction to do the configuration adaptation. Periodic profiling uses profiling configurations to adapt the configuration.
Figure 4.9: Processing time of MOTrack and ground truth configuration for one video clip on traffic tracking and pose estimation

Figure 4.10 shows the average (configuration) adaptation time on the two applications. MOTrack takes about 97%-98% less time than the periodic profiling’s adaptation time. The efficiency of configuration adaptation has been greatly improved with the MOTrack method.

Figure 4.10: Configuration adaptation time of MOTrack compared with periodic profiling
4.8 Related work

4.8.1 Traditional Configuration Optimization

Researches on optimization [6, 61, 160, 111, 193, 118, 178] for selecting the optimal configuration have been seen in different problem domain. CherryPick [6] use Bayesian optimizer to find the right cloud configuration for recurring big data analytics jobs.

Hill et al. [61] explore the layout space in interactive web pages, and use a multi-armed bandit methodology and hill-climbing algorithm to select the optimal content. Ernest et al. [160] uses an optimal experiment design based on the behavior of the job on small samples of data to find the right hardware configuration in large-scale advanced analytic applications. Another similar types of research [127, 90, 139, 163] considers job scheduling for the best resource allocation in large machine learning system. AlloX [90] considers the dynamic fair allocation over time in deep learning for a variety of interchangeable resources of CPU, GPU, etc., by transferring the scheduling problem to a min-cost bipartite matching problem. Autopilot [139] addresses the problem of user manually requesting resources limits and automatically configures resources through adjusting both number of tasks and CPU/memory limit for each task. It uses machine learning algorithms to learn historical data about prior executions of a job and a set of finely-tuned heuristics.

4.8.2 Video Analytics Pipeline Optimization

Existing research for video pipeline resource management and optimization mainly develops algorithms through specialized DNN models or through adjusting configurations with costly profiling [181, 179, 99, 162, 72, 39, 24]. MCDNN [59], NoScope [80] and Focus [64] optimize video analytics with resource-light specialized DNN models to detect objects. The core of specialized DNN models is to train compressed DNN models with fewer layers or parameters on a few objects that typically appear on
video streams. Some other researchers consider the temporal and spatio-temporal characteristics of videos. Samvit et al. [71] utilize the cross camera spatio-temporal characteristics to remain or improve inference for scalable camera deployment. Our work utilizes the spatio-temporal characteristics to capture object movement features for configuration adaptation dynamically.

VideoStorm [181] processes live video streams over large clusters by profiling each video query in the cluster and change configurations to maximize performance on accuracy and latency. AWStream [179] and JetStream [130], however, consider the wide-area network changing by profiling the video query to achieve the trade-off of accuracy and bandwidth. Our work avoids the fixed period scheduling and overcomes the expensive profiling for configuration adaptation.

The configuration involved in the video analytics could be frame rate, resolution, bandwidth, models, etc. [75] proposes dynamically picks the best configurations of frame rate, resolution and models based on existing NN-based video analytics pipelines. It uses periodical profiling in each interval to decide the best configuration and apply to the rest of the interval, and considers the cross similarities and configuration knob independence for multiple video stream analytics as well. [162] considers the frame rate, resolution and bandwidth allocation as the configuration to select the optimization based on Lyapunov optimization and Markov approximation to achieves a provable performance bound. In addition, there is a novel research area about the configuration selection of decoding options for video analytics. Tarek et al. [41] optimize the selection of encoded videos by proposing a technique for semantic video encoding, in which video encoders are aware of the semantics of the downstream task (e.g., object detection) for the analytics system to reduce the latency and increase the throughput of analytics over video streams.

There are also some work exploring the configuration scheduling of different applications of video analytics. Xukan et al. [131] consider the augmented reality app
with the expensive deep learning framework. To achieve the framework, they explore the complex interaction between model accuracy, video quality, battery constraints, network data usage, and network conditions for an optimal offloading strategy, and design an augmented reality framework that ties together front-end devices with more powerful backend “helpers” (e.g., home servers). Ning et al. [24] explore configuration selection on the augmented reality application. They propose a system that generates video configuration decisions for augmented reality application using reinforcement learning (RL) by training a neural network model that picks a configuration for the next encoding slot based on historical observations collected by augmented reality devices. Our work serves on the core object detection and tracking algorithms which could also be applicable on the configuration adaptation for augmented reality application.
CHAPTER 5
PRIVACY-PRESERVING VIDEO ANALYTICS

5.1 Introduction

Video analytics is increasingly utilized for traffic control, business intelligence, action/event analysis, human-machine interaction, VR/AR, etc. Currently, video analytics as a service on cloud servers are widely used with deep learning methods. It involves lots of private data directly transmitted to the service providers’ cloud servers. Many of the video surveillance data contains clients’ private information such as medical records, personalized profiles, financial information, and so on. These private and sensitive information are usually offered by data owner parties (other than cloud service providers), making it difficult to build trust toward other parties. To make deep learning or video analytics as a service widely applicable, the demand for privacy-preserving deep learning is increasing.

Recent privacy-preserving deep learning methods are mainly divided into two types. One type method is the secure multi-party computation [40, 86], which is to create certain methods for multiple parties to jointly compute a function over their input data and also keep those input data private. The second type is through cryptography. With cryptography, homomorphic encryption (HE), garbled circuits, secret sharing and secure processors are the most widely used cryptographic techniques to achieve privacy-preserving deep learning [4, 128]. Our work explores the feasibility of fully homomorphic encryption (FHE) for video analytics. FHE could allow computations on the encrypted data without disclosing the raw data, making it suitable for privacy-preserving in the clouds. FHE for deep learning is an on-going hot research...
topic in recent years. There exists some work on FHE exploring the feasibility of computer visions tasks on deep neural networks. Due to the commonly used deep neural networks for lots of applications and the highly expensive computations involved in the FHE, research on how to apply FHE to deep neural network-based computer vision tasks focuses on the image classification. CryptoNets [52] is one of initiators that explores deep neural network with MNIST optical character recognition tasks with 5–9 layers of neural network, which is implemented based on the Simple Encrypted Arithmetic Library (SEAL) [22] for homomorphic encryption. Juvekar et al. [79] combine homomorphic encryption with garbled circuit and experiment on the image classification with 5 layers of neural networks for MNIST and CIFAR-10 datasets. Lee et al. [93] explore FHE with more deep layers of ResNet-20 image classification using approximation and bootstrapping, which improve the classification efficiency on encrypted images based on RNS-CKKS homomorphic encryption scheme. It achieves the average classification accuracy on encrypted image about 90.67%, and the inference time is about $10^6$ magnitude more time compared without the classification on raw image.

These current work on FHE for image classification focuses on deep neural networks with a small number of neural network layers. To the best of our knowledge, we are the first to explore FHE on private data for video analytics pose estimation application. We explore the feasibility of video analytics and implement a privacy-preserving pose estimation system to protect private data on the cloud servers. We have shown the accuracy and inference time of pose estimation with SEAL-based library.

We have achieved the average accuracy of 76.5% on encrypted frames with FHE compared to the accuracy 88.2% on raw frames. About the inference efficiency, it achieves the 252,201 second per frame on encrypted data with FHE compared to 1.232 second per frame on the raw frames. Moreover, if we apply a preprocessing
technique of identifying the human object first at the client-side, the efficiency could be further reduced 20%–34% compared to the time without preprocessing and only about 2% accuracy loss.

5.2 Preliminary

In this section, we briefly introduce the foundation of fully homomorphic encryption and the pose estimation.

5.2.1 Fully Homomorphic Encryption

Homomorphic encryption is an encryption scheme that allows the operation on the encrypted data without accessing to the raw data. It achieves similar results like doing the operations in the raw data. This encrypted scheme belongs to public-key encryption [146] (also known as asymmetric encryption). The basic supported operation for homomorphic encryption is addition, which is also called partial homomorphic encryption. Fully homomorphic encryption allows for more complex computation over encryption. It supports not only addition but also multiplication operation.

Figure 5.1 shows the diagram of FHE. In this diagram, a message (raw data) \( m \) is encrypted to \( \text{enc}(m) \). FHE scheme allows the operation \( f \) directly on the \( \text{enc}(m) \), which is approximate to the operation after the encryption of \( f(m) \). That is, \( f(\text{enc}(m)) \approx \text{enc}(f(m)) \). For example, for a basic operation \( f \), we define \( f \) standing for \( + \), \(-\) and \( * \). Assume there exist two messages \( a \) and \( b \), we have, \( \text{Enc}(a) f \text{Enc}(b) = \text{Enc}(a \ f \ b) \).

The encryption scheme for public key or private key scheme is as follows:

- **KeyGen**(\( 1^k \)) \( \rightarrow (k, pp) \): The input is the security parameter, and the outputs are the key \( k \) and some public parameters \( pp \).

- **Encrypt**(\( k, m \)) \( \rightarrow c \): The inputs are the key \( k \) and a message \( m \), and the output is a ciphertext \( c \).
• Decrypt \( (k, c) \rightarrow m \): the inputs are the key \( k \) and a ciphertext \( c \), and the output is a message \( m \).

### 5.2.2 Pose Estimation

Pose estimation localizes anatomical keypoints or “parts” from the human body [8]. Each human object in a frame has 17–19 keypoints as the video analytics result. In our work, we use the CMU OpenPose model [17, 112] as our pose estimation model. OpenPose is one of the classical pose estimation to detect human pose with multi-stages of neural network. The OpenPose model architecture is shown in Figure 5.2. The architecture includes multi-stages, and each stage includes two branches with different blocks of layers. The model performance depends on the number of stage used. From the experiments on the OpenPose, stage 2–6 are good choices to achieve similar performance.

### 5.3 Fully Homomorphic Encryption for Pose Estimation

We explore the fully homomorphic encryption on pose estimation. We use 2 stages to reduce the complexity and inference time of the neural network model. Moreover, it is enough to get a 98% accuracy compared to using 3–6 stages. With 2 stages, the
neural network model contains 45 layers. It has the most number of layers considered for FHE compared to the existing work. We show in Table 5.1 about part of the layers in the model for the input size, output size and parameter size when the input frame size is 128 x 128.

In this model, there are convolutional layer, max pooling layer, ReLU and batch normalization layer involved in this pose estimation model only.

Table 5.1: The neural network layers used for FHE on pose estimation

<table>
<thead>
<tr>
<th>No.</th>
<th>Layer</th>
<th>Input size</th>
<th>Filter size</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Conv</td>
<td>128 x 128 x 3</td>
<td>3 x 3</td>
<td>128 x 128 x 64</td>
</tr>
<tr>
<td>2</td>
<td>ReLU</td>
<td>128 x 128 x 64</td>
<td>-</td>
<td>128 x 128 x 64</td>
</tr>
<tr>
<td>3</td>
<td>Conv</td>
<td>128 x 128 x 64</td>
<td>3 x 3</td>
<td>128 x 128 x 64</td>
</tr>
<tr>
<td>4</td>
<td>MaxPool</td>
<td>128 x 128 x 64</td>
<td>2 x 2</td>
<td>64 x 64 x 64</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Conv</td>
<td>32 x 32 x 128</td>
<td>3 x 3</td>
<td>32 x 32 x 128</td>
</tr>
<tr>
<td>42</td>
<td>ReLU</td>
<td>32 x 32 x 128</td>
<td>-</td>
<td>32 x 32 x 128</td>
</tr>
<tr>
<td>43</td>
<td>Conv</td>
<td>32 x 32 x 128</td>
<td>1 x 1</td>
<td>32 x 32 x 512</td>
</tr>
<tr>
<td>44</td>
<td>ReLU</td>
<td>32 x 32 x 512</td>
<td>-</td>
<td>32 x 32 x 512</td>
</tr>
<tr>
<td>45</td>
<td>Conv</td>
<td>32 x 32 x 512</td>
<td>1 x 1</td>
<td>32 x 32 x 19</td>
</tr>
</tbody>
</table>
5.3.1 Encryption Scheme

We implement our FHE for video analytics with a FHE library–Pyfhel CKKS[67]. It is based on SEAL/PALISADE backends, implemented with Cython, and uses Cheon-Kim-Kim-Song (CKKS) scheme [29] with a Python interface. Pyfhel supports sum, subtraction, multiplication, scalar product or matrix multiplication.

The CKKS high-level scheme is shown in Figure 5.3. CKKS uses the ring of polynomials to ensure the trade-off between efficiency and security. The input message \( m \) is encoded into a plaintext polynomial \( p(X) \) and then encrypted into a ciphertext with a ring of polynomials.

![Figure 5.3: CKKS scheme overview [66]](image_url)

5.3.2 Convolutional Layer

For convolutional layer, there are two common ways to implement it on encrypted data. The first way is to get the convolution operation around each kernel block on the input matrix. The second way is to flatten the input matrix to the column vector (doing im2col) and then do the kernel convolution on the vector without many rotations. Our pose estimation model involves multiple convolutions with layers by layers. If using im2col, it consumes more memory spaces and also trivial to get the indices back and then transfer to vectors again in the FHE space. We have
implemented the first method with multiplication and addition on the encrypted data with multiple convolutional layer operations.

5.3.3 Batch Normalization Layer

Batch normalization [69] is usually to accelerate training time convergence in the training stage with multiple samples as a mini-batch size and improve the accuracy on test data. It uses the average and variance of the layer output of each element in a mini-batch to reduce internal covariate shift in the distribution of the network from changing significantly. In the testing stage, there is no mini-batch of sampling size. One way is to take the original training stage’s mean and variance as the statistics here. The other way is to set the batch size as 1. Then it is a simple linear function involving constant coefficients, homomorphic addition, and the homomorphic scalar multiplication.

5.3.4 ReLU Activation

ReLU activation involves max operation, which is not supported directly by FHE. There are some research work on how to approximate ReLU with some arithmetic function or non-linear polynomials. The simplest arithmetic function is to use $X^2$ to approximate it [52]. Other more accurate but more time-consuming approximation methods are based on different degrees of polynomial mapping functions [16, 94, 70]. In our work, we use polynomial degree of 4 and its coefficients from [70] to approximate the ReLU layer.

$$ReLU(x) \approx 0.119782 + 0.5x + 0.0147298x^2 - 0.002015x^4$$  \hspace{1cm} (5.1)

5.3.5 Max Pooling Layer

For the max pooling layer, it involves the max operation. We could use the approximate operation of the ReLU idea to get the max operation. However, it is
very time-consuming with each pooling kernel operation. Therefore, one commonly
used method [16] is to use the convolutional sum or average operation to replace
max pooling operation to approximate max pooling layer. As this could reduce the
inference computation and is also more stable than max pool to perturbations. Here
we use the average operation of kernel to approximate max pooling layer. With a
weight kernel $F$ in a size $k \times k$ on an input frame $I$, we obtain the average pooling
as the output. As show in Equation 5.2, for each position ($i$ and $j$ are the index in
the input frame matrix $I$, the average pooling $G[i, j]$ is obtained through the average
values of kernel option.

\[
G[i, j] = \frac{1}{k \times k} \sum_{u=-k/2}^{k/2} \sum_{v=-k/2}^{k/2} I[u, v] F[i + u, j + v]
\] (5.2)

5.4 Improving Inference Performance through Preprocessing

In our video analytics, there are many scenarios that include redundant infor-
mation, such as the static backgrounds. In the pose estimation, the human objects
in the video are the effective point of interest. To save time of transmission and
inference operation on the server, it is also meaningful to do some preprocessing on
the client. Therefore, we propose a method to improve the inference performance by
doing object detection first to extract the human objects on the client first, encrypt
only the detected human objects and then send to the server. The object detection
is highly efficient and accurate with recent years of technique advances, which makes
it reliable and efficient.

In our work, out of many object detection algorithms, we use Faster R-CNN [133,
168], which is generally more accurate and also efficient on many low-end configuration
dge servers.
5.5 Experimental Evaluation

We have implemented the pose estimation with FHE on encrypted data for pose estimation. The FHE is based on the Pyfhel library with CKKS scheme [67]. The plaintext pose model up to 2 stages with 45 neural network layers are retrained on the raw data–Microsoft COCO [102]. The FHE are tested on the encrypted data in the test stage. We have tested a sequence of video frames and obtain the output.

5.5.1 Environment and Parameter Setting

We experiment on a server with 16 CPUs @4500 MHz, 32GB RAM, 192GB virtual RAM and 1TB disk.

There are some parameters involving with the encryption and security levels shown in Table 5.2. Out of them, polynomial degree modulus is set as a power of $2^{16} + 1$ as 65537, which impacts the encryption noise depth. With our deep neural network layers, this is the smallest number that we have tried to achieve the accurate encryption and decryption with large amounts of FHE operations. The polynomial coefficient modulus is set as $m = 2048$, which is directly related to the encrypted ciphertext depth. This impacts the level of encrypted operation of multiplication. The security level is a measure of the strength that a cryptographic primitive achieves, which is equivalent to that in the Advanced Encryption Standard (AES). AES comes in 128-bit, 192-bit, and 256-bit, while AES 256-bit being the most secure. We select the minimum security level 128-bit for FHE operation. Relinearization key is used to reduce size of the ciphertexts when multiplying orponenting them. If there are multis/exponentiations between encrypted data, ciphertexts would grow in size with higher noises and may lead the failure of decryption.

The dataset for pose estimation is from our previous work of configuration adaptation video analytics. We use a video clip of dancing here. Due to time limit, we only
show the average performance of continuous 10 frames of a video. We have obtained
the performance of the encryption and the final inference of FHE on pose estimation.

5.5.2 Encryption Efficiency

Here we experiment the efficiency of different frame sizes on our server in Table 5.3. It shows the encryption time monotonically increases on the client with different frame sizes. This makes necessary the limitation of the frame size with preprocessing and sending the most valuable information to the server.

Table 5.3: Encryption efficiency with FHE

<table>
<thead>
<tr>
<th>Frame size</th>
<th>Processing time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 x 32</td>
<td>25.04</td>
</tr>
<tr>
<td>64 x 64</td>
<td>100.31</td>
</tr>
<tr>
<td>128 x 128</td>
<td>400.03</td>
</tr>
<tr>
<td>256 x 256</td>
<td>1601.12</td>
</tr>
</tbody>
</table>

5.5.3 Inference Efficiency of Neural Network Layers and Activation Functions

Here we experiment the inference efficiency on the necessary neural network layers and activation functions with different frame sizes on our server in Table 5.4. It shows ReLU takes the highest time among them, then convolutional layer and max pooling layer take similarly less time.

5.5.4 Performance of FHE on Pose Estimation

We experiment on a sequence of frames for FHE on pose estimation and obtain the performance output. We have observed that the input frame in the plaintext

<table>
<thead>
<tr>
<th>Plaintext modulus</th>
<th>Polynomial coefficient modulus</th>
<th>Polynomial base</th>
<th>Security level</th>
<th>Relinearization key</th>
</tr>
</thead>
<tbody>
<tr>
<td>65537</td>
<td>2048</td>
<td>2</td>
<td>128</td>
<td>20</td>
</tr>
</tbody>
</table>
Table 5.4: Inference time with different layers and activation function for a frame

<table>
<thead>
<tr>
<th>Frame size</th>
<th>Convolutional layer (second)</th>
<th>Max pooling layer (second)</th>
<th>ReLU (second)</th>
<th>Batch normalization (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 x 32</td>
<td>2.135</td>
<td>2.0426</td>
<td>4.4622</td>
<td>1.4948</td>
</tr>
<tr>
<td>64 x 64</td>
<td>8.858</td>
<td>8.61575</td>
<td>17.6832</td>
<td>4.7536</td>
</tr>
<tr>
<td>128 x 128</td>
<td>31.53</td>
<td>32.1614</td>
<td>70.5934</td>
<td>16.5160</td>
</tr>
<tr>
<td>256 x 256</td>
<td>123.23</td>
<td>124.52</td>
<td>286.392</td>
<td>66.313</td>
</tr>
</tbody>
</table>

pose model should be at least 128 x 128 to get an effective output of a certain pose estimation result but with a very low accuracy. In our model, it is better to have 256 x 256 to get a certain accuracy close to 90% of the ground truth. This is because the pre-defined channel number is up to 512 in the OpenPose neural network design. Therefore, we show the result for the minimum input frame-256 x 256 here in Table 5.5. With our server capacity, we have achieved an average inference time about 252,201 second/frame. Meanwhile, we have achieved the accuracy of 76.5% on the average compared to the ground truth, while the plaintext pose estimation on raw data has the average accuracy of 88.2%. That is, we have achieved 86.7% agreement compared with the plaintext pose model on raw data. Even though the long inference time, we have shown the potential of privacy-preserving video analytics and possible improvement with more advanced encryption scheme and powerful servers for future work.

Table 5.5: The performance of FHE pose estimation and plaintext pose estimation

<table>
<thead>
<tr>
<th>Application</th>
<th>Accuracy</th>
<th>Inference time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FHE OpenPose</td>
<td>76.5%</td>
<td>252,201</td>
</tr>
<tr>
<td>Plaintext OpenPose</td>
<td>88.2%</td>
<td>1.232</td>
</tr>
</tbody>
</table>

5.5.5 Performance of FHE on Pose Estimation with Preprocessing

Here we compare the inference efficiency of different input frame sizes with the preprocessing of extracting human objects or without preprocessing on the same
configuration in Table 5.6. We can see that the inference time has decreased about 20% with preprocessing for these frame resolutions.

Table 5.6: Comparison of efficiency of FHE w/ and w/o processing

<table>
<thead>
<tr>
<th>Application</th>
<th>128 x 128</th>
<th>256 x 256</th>
<th>512 x 512</th>
</tr>
</thead>
<tbody>
<tr>
<td>FHE OpenPose without preprocessing</td>
<td>169,862</td>
<td>252,201</td>
<td>578,453</td>
</tr>
<tr>
<td>FHE OpenPose with preprocessing</td>
<td>131,506</td>
<td>202,421</td>
<td>429,063</td>
</tr>
</tbody>
</table>

We further demonstrate the performance with the preprocessing for 256 x 256 input frames on the same machine configuration in Figure 5.7. With this processing, the obtained input frame size takes about 40%–80% of the plaintext frame input’s width or height in the pose estimation video dataset. This operation is similar to cropping a region out of an image into a small resolution as the new input size for encryption. This could dramatically reduce the frame size input for encryption. We show the performance change of pose estimation with FHE on encrypted data and the plaintext pose estimation on raw data compared to that without preprocessing. It exhibits that the accuracy reduces only a little about 0.8%–2.2%, while the inference time has reduced about 20%–34%. It indicates the usefulness of the preprocessing method.

Table 5.7: Pose estimation performance with preprocessing

<table>
<thead>
<tr>
<th>Application</th>
<th>Accuracy reduction</th>
<th>Inference time reduction (second/percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FHE OpenPose with preprocessing</td>
<td>-2.2%</td>
<td>-49,780/-20%</td>
</tr>
<tr>
<td>Plaintext OpenPose with preprocessing</td>
<td>-0.8%</td>
<td>-0.423/-34%</td>
</tr>
</tbody>
</table>

5.6 Related Work

For privacy-preserving deep learning, there are two common types of methods. The secure multi-party computation, and cryptographic method. For cryptographic
methods, fully homomorphic encryption is currently one of the most popular methods to achieve privacy-preserving.

5.6.1 Secure Multi-party Computation

Secure multi-party computation (MPC) [135, 188, 25, 110, 104] allows multiple party to compute their combine data without exposing each private data. There are a number of studies implementing machine-learning models via secure MPC. CRYPTEN [86] develops a software framework that exposes secure MPC primitives via abstractions such as tensor computations, automatic differentiation, and modular neural networks, which are commonly used in deep learning frameworks.

5.6.2 Fully Homomorphic Encryption for Deep Learning

Cryptographic methods could be utilized to perform ML training/testing on encrypted data. Out of these methods, fully homomorphic encryption (FHE) is one popular method that draws lots of attentions in the academic community. As deep neural networks gain popular in many domains, lots of tasks are achieved with this. To achieve homomorphic encryption on private data for deep neural networks, lots of research work has also been explored extensively [52, 30, 159, 93, 153]. Deep neural network involves a lot of different layers with different operations. To achieve FHE for deep neural network, it needs to achieve the complex operation of arithmetic and non-arithmetic. Due to the limited support of addition and multiplication operations by FHE, lots of research explores how to approximate the complex operation involved in different layers. Common deep neural network models contains fully connected layer, convolutional layer, pooling layer, and other activation functions. For the layer operation that involves addition and multiplication, they are directly supported by FHE. CryptoNets [52] explore 5–9 layers with convolutional and fully connected layers, which is one of the earliest explorer for FHE on deep neural network with MNIST optical character recognition tasks. The activation functions (e.g. ReLU, LeakyReLU,
Sigmod and Softmax) usually involve non-arithmetic operations, which could not directly supported by FHE. The approximation methods are researched extensively to approximate with non-linear polynomial operations.

5.7 Conclusion

To make the video analytics as a service more applicable with cloud service providers, we explore the plausibility of privacy-preserving on encrypted frames for video analytics pose estimation application on deep neural network layers with fully homomorphic encryption. We develop a privacy-preserving pose estimation system based on the FHE SEAL library and demonstrate the feasibility on a CPU server.
CHAPTER 6
CONCLUSIONS

In this dissertation, we explore the big data analytics in the relational database integrated to a graph database, then we further develop a graph query in the graph database with hierarchical inheritance relations. Thirdly, we propose a real-time configuration adaption for video analytics pipeline through movement prediction. Finally, we investigate the feasibility of privacy-preserving video analytics with fully homomorphic encryption. The summary of our work is as follows:

Firstly, we primarily discuss the big database integration and lays the foundation for network management and analytics on a graph database. It has direct implications for network management to help network operators/administrators with network query, network diagnosis, fault detection, network performance monitoring, etc. One specific example for network query is that it is more efficient to use graph traversal algorithms to find out which network routers communicated with CiscoASR9010 have the most frequent incidents in the last year. Another example is that if a network ticket/incident occurs, with the help of graph clustering or propagation models, the administrators could easily locate the network failure and other affected networks, and analyze its root cause. The system can also automatically suggest a potential solution to the network failure based on previous histories of tickets/incidents.

Secondly, we consider an additional dimension of hierarchical inheritance relations on real-world heterogeneous information networks for graph query in the knowledge graph database. The problem is reformulated with hierarchical inheritance relations, and we propose a graph query algorithm based on that for star query and general
graph query. With the bounding-based techniques, our algorithm can effectively capture hierarchical inheritance relations on information networks for better query answers, and competitive performances are also achieved.

Thirdly, we propose a configuration adaption algorithm for video analytics through movement tracking. Considering estimating object movement information from past object tracking results, we devise a machine learning-based method to predict effectively and efficiently the configuration over time dynamically. This reduces the cost of expensive profiling and overcomes the fixed period of configuration adaptation. Our results suggest that our method can make smart decisions under different video analytics applications, which achieves better accuracy and less resource cost compared to state-of-the-art methods.

Finally, we explore the privacy preserving video analytics and successfully implement a privacy preserving pose estimation with fully homomorphic encryption with the SEAL library. We demonstrate the feasibility and potential of privacy preserving scheme for video analytics on a low-end configuration CPU server.

There are some limitations of the privacy-preserving video analytics. The inference time for the current model is scores of hours. It is somewhat large for the practical use. This work firstly shows the feasibility of applying FHE to a deep neural network model with higher number of neural network layers, but it can be optimized and improved in various ways to reduce the inference time and keep the similar accuracy for future work.

(1) Investigate different FHE schemes and optimize the implementations for efficient processing.

(2) Explore the GPU implementation. Currently the model is implemented on a CPU server, which could be further studied on GPU servers with FHE and expected to have a significant efficiency improvement.


[57] Gupta, Manish, Gao, Jing, Yan, Xifeng, Cam, Hasan, and Han, Jiawei. Top-k interesting subgraph discovery in information networks. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on* (2014), IEEE, pp. 820–831.


[97] Ley, Michael. Dblp computer science bibliography.


[151] Sun, Yizhou, and Han, Jiawei. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery* 3, 2 (2012), 1–159.


[180] Zhang, Hanqi, Guo, Jianfeng, Deng, Chao, Fan, Ying, and Gu, Fu. Can video surveillance systems promote the perception of safety? evidence from surveys on residents in Beijing, China. Sustainability 11, 6 (2019), 1595.


[192] Zhu, Yuanyuan, Qin, Lu, Yu, Jeffrey Xu, and Cheng, Hong. Answering top-k graph similarity queries in graph databases. *IEEE Transactions on Knowledge and Data Engineering* (2019).