中图分类号：TP392
论 文 编 号：10006BY1206122

北京航空航天大學

博 士 学 位 论 文

图匹配查询的高效性和易用性理论与算法研究

作 者 姓 名　李佳
学 科 专 业　计算机软件与理论
指 导 教 师　刘旭东　教授
　　马帅　教授
培 养 院 系　计算机学院

Improving the Efficiency and Usability of Graph Pattern Matching: Theory, Algorithms and Frameworks

A Dissertation Submitted for the Degree of Doctor of Philosophy

Candidate: LI Jia

Supervisor: Prof. LIU Xudong

Prof. MA Shuai

School of Computer Science and Engineering

Beihang University, Beijing, China

中图分类号：TP392

论 文 编 号：10006BY1206122

博 士 学 位 论 文

图匹配查询的高效性和易用性理论与算法研究

作者姓名	李佳	申请学位级别	工学博士
指导教师姓名	刘旭东　马帅	职　　称	教授
学科专业	计算机软件与理论	研究方向	数据库和图数据查询
学习时间自	2012 年 09 月 01 日	起至	2018 年 10 月 01 日止
论文提交日期	2018 年 09 月 XX 日	论文答辩日期	2018 年 11 月 XX 日
学位授予单位	北京航空航天大学	学位授予日期	2018 年 11 月 XX 日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：_____ 日期：　　年　　月　　日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名：_____ 日期：　　年　　月　　日

指导教师签名：_____ 日期：　　年　　月　　日

摘　要

与传统的关系数据模型和 XML 数据模型相比，图在描述和刻画复杂的现实世界方面具有更强的表达能力。图可以将现实世界中的实体抽象为节点，将实体间的关系抽象为节点之间的边，从而能直观地表达实体间的复杂关系。图数据在各个领域日益增长的应用使得图查询（从图中查询信息）成为了学术界和工业界共同的研究热点。本文主要关注图查询中最复杂的一类查询：图匹配查询。

图匹配查询指从数据图中找出与查询模式图结构匹配的子图集合。图匹配查询在数据清洗中的复杂对象识别、社交网络和 Web 网络查询、生物数据分析和软件代码剽窃检测等方面都有重要的应用。然而，相对于研究较为成熟的关系数据库查询和 Web 网络搜索技术，图匹配查询技术尚处于研究的初始阶段，存在很多问题。

（1）准确性低。查询准确性指查询引擎输出的结果与用户输入的查询之间的匹配程度。图匹配查询根据不同的匹配语义形成了基于子图同构、图模拟以及其扩展强模拟等不同的图匹配查询语义。这些图匹配查询语义在查询准确度和计算复杂度之间做出了不同的平衡，使得用户可以根据具体应用需求选择合适的查询语义。然而，由于图数据表达的实体关系的复杂性和图查询应用的多样性，这些图匹配查询语义仍然存在实用性较低、难以表达实际应用的查询需求、以及查询结果过少等问题。

（2）效率低。查询效率指查询引擎计算得到查询结果的快慢程度。传统的基于子图同构的图匹配查询是 NP-完全问题，基于图模拟的图匹配查询通过一定程度地降低查询准确度达到了二次方时间复杂度。由于社交网络、生物数据等应用产生的图数据规模巨大，即使传统意义上认为效率较高的计算模式（如线性时间）在这些大规模复杂数据上也会具有极高计算开销。因此，现有的图匹配查询在大规模图数据上的执行效率仍然很低，并且我们已不能继续仅依赖于通过提出新的查询语义的方法以降低查询准确度为代价提升图匹配查询效率。此外，社交网络等应用产生的图数据及其相关的图查询具有高度的动态性。每当数据图或模式图有更新变化，重新执行图匹配查询整个计算过程的更新处理法计算开销太大，不具备可行性。因此，我们需要从新的维度提出方法来提高已有各种图匹配查询语义的执行效率，并且能有效处理静态和动态环境下的查询计算过程。

（3）易用性差。易用性衡量查询引擎能够为用户提供友好的查询技术并能根据用户的查询意图返回令用户满意的结果的能力。相较于图查询中的点查询和 Web 网络上的

关键词搜索，当前图匹配查询的用户友好性较低。图匹配查询严格的匹配约束条件经常导致模式图在数据图中的图匹配结果较少甚至为空集。此外，对于规模较大并且结构复杂的数据图，由于无法全面掌握数据图基本信息，用户很容易构造出不存在任何查询结果的模式图。提高图匹配查询的易用性是降低图匹配查询使用难度亟待解决的问题。

本文以提高图匹配查询的准确性、高效性和易用性为目标，设计新的“通用图匹配查询计算方法”以提供准确的、高效的和用户友好的图匹配查询过程。更重要的是该方法是与语义无关的通用方法，能够支持现有的所有图匹配查询语义。论文主要贡献如下：

（1）基于视图的近似图匹配查询方法。针对图匹配查询所有图匹配语义在大规模数据上计算复杂度过高的问题，本文提出一种通用的基于视图的近似图匹配查询方法。该方法使用视图技术来加速图匹配查询计算过程，并引入上界和下界近似查询的概念来提高视图查询中存储视图的命中率和利用率，从而提高图匹配查询引擎的效率。

（2）图匹配查询增量计算方法。针对图匹配查询中模式图和数据图的频繁更新导致动态环境下图匹配查询计算复杂度过高的问题，本文提出一种统一的图匹配查询增量计算方法。该方法可以处理连续的单独或混合的模式图和数据图更新，有效避免因为查询或数据图更新而导致的重复计算，从而提高动态环境下图匹配查询引擎的效率。

（3）图匹配查询松弛重构方法。针对图匹配查询语义约束条件较严格经常导致图匹配查询结果较少甚至为空集的问题，本文提出一种通用的图匹配查询松弛重构方法。该方法包含一种基于语义分类信息的查询重构技术，使得图匹配查询引擎能够自动对模式图进行松弛重构以确保能返回有意义的查询结果，从而提高图匹配查询引擎的易用性。该方法同时能对查询松弛后计算得到的结果进行解释，保证查询结果的准确性。

此外，在介绍以上图匹配查询计算方法的过程中，本文还提出并使用了组合模拟图匹配语义以及基于语义分类信息的泛化图匹配查询语义。这些新的图匹配语义能为社交推荐和团队搜索等应用领域提供更高的查询准确性。

关键词：图匹配查询，高效性，易用性，通用图匹配计算方法，计算复杂性，算法

Abstract

Compared with traditional relational and XML models, graphs have more expressive power and play an important role in many applications. This is because the core data involved in real-life applications can be conveniently represented as graphs, by treating entities as nodes and the relationships between entities as edges. The wide use of graphs has brought about the emergence of graph search, *i.e.,*, retrieving information from big graphs in a timely, easy and accurate manner, which has drawn increasing attention from both the industry and academia. This paper focuses on a core class of graph search queries: graph pattern matching queries.

Given a pattern graph and a data graph, graph pattern matching is to identify subgraphs of the data graph that match the given pattern graph. Graph pattern matching plays an important role in many applications, such as complex object identification in data cleaning, social network analyses, Web search, biological data analyses and software plagiarism detection. However, compared to the well-studied relational query processing and Web search technologies, graph pattern matching is still in its initial stage and there are many problems to be studied.

(1) Lack of accuracy. Accuracy measures the ability of graph pattern matching queries in capturing reasonable matching results. There are several matching semantics proposed for graph pattern matching queries, namely, subgraph isomorphism, graph simulation, and its extensions such as strong simulation. These matching semantics vary in their way of striking a balance between the accuracy and computational complexity, allowing users to pick appropriate query semantics for their applications. However, due to the complexity of graphs and the diversity of applications, these matching semantics still have difficulty in accurately expressing the query requirements of real-life applications and easily returning too few or even empty query results.

(2) Lack of efficiency. Efficiency measures the speed of graph pattern matching engines in computing query results. Subgraph isomorphism is NP-complete, while graph simulation achieves a quadratic time complexity by reducing the query accuracy to a certain extent. However, over the unprecedentedly large-scale data graphs generated by, *e.g.,* social networks and biological applications, even computations that are regarded as super efficient traditionally, such as linear time algorithms, can incur extremely high computational cost and are becoming infeasible.

Therefore, existing graph pattern queries are inefficient on large-scale graph data, and we can no longer improve query efficiency by revising matching semantics at the cost of a lower accuracy. In addition, for *e.g.,* social network analyses, the data graphs and pattern queries are highly dynamic. Whenever the data graph or pattern graph is updated, recomputing the matching results from scratch is often too expensive and is not feasible. Therefore, we need to approaches from new perspectives to improve the efficiency of all existing graph pattern matching semantics, and to handle graph pattern matching queries in both static and dynamic environments.

(3) Lack of usability. Usability measures the ability of graph pattern matching engines in providing users with friendly query technologies and returning results that well match the users' query intents. Compared with node queries of graph search and keyword search on the Web network, graph pattern matching is less user-friendly. This is because existing graph pattern matching semantics are often too restrictive to identify meaningful matches. Moreover, for large-scale and complex data graphs, it is typically impossible for users to have a thorough knowledge of the data graphs so that it is quite common for users to construct a pattern graph without any query results in the data graphs. Improving the usability of graph pattern matching is an urgent need for expanding the use of graph pattern matching in real-life applications.

This paper aims to improve the accuracy, efficiency and usability of graph pattern matching. This paper proposes new "generic graph pattern matching frameworks" to provide accurate, efficient and user-friendly graph pattern matching methods. More importantly, the frameworks are generic in the sense that they are independent of query semantics. Hence, they can support all existing graph pattern matching semantics. The main contribution of this paper is as follows.

(1) Approximating graph pattern queries using views. As we know, the computational complexity of existing graph pattern matching semantics are often too high on large-scale graph data. To tackle the problem, this paper proposes a generic framework for approximating graph pattern queries using views. This method utilizes views to accelerate the process of graph pattern matching computation, and introduces the concept of upper and lower approximations to improve the utilization of materialized views in the view answering process. Combining these two approaches, the framework can effectively improve the efficiency of graph pattern matching.

(2) Incremental computation for dynamic graph pattern matching. Frequent updates in both pattern and data graphs lead to high computational overhead of graph pattern matching in

the dynamic environment. To remedy this, this paper proposes a generic incremental framework to uniformly handle both pattern and data updates together. This method improves the efficiency of graph pattern matching in dynamic environment by avoiding re-computing graph pattern matching results from scratch when the pattern graphs or/and data graphs are updated.

(3) Relaxing graph pattern matching with explanations. For the problem that the matching semantics are often too restrictive to identify meaningful matches, this paper proposes a generic relaxation framework to enrich the results of graph pattern matching. It utilizes taxonomy information to generate pattern relaxations, which allow graph pattern matching engines to automatically relax pattern graphs to ensure that meaningful match results can be captured using the relaxed patterns, thereby improving the usability of graph pattern matching. It also provides a notion of explanations for answers to the relaxations, ensuring the accuracy of query results.

In addition, when designing the above graph pattern matching frameworks, this paper has also proposed new graph pattern matching semantics including group simulation and taxonomy simulation. These query semantics improve the accuracy of graph pattern matching for real-life applications such as social recommendation and team formation.

Key words: Graph pattern matching, Efficiency, Usability, Generic frameworks, Complexity, Algorithms

目　录

插 图 目 录

表 格 目 录

第一章　绪论

1.1　研究背景

互联网时代，社交网络与电子商务等网络和应用的升级推动了社会计算的发展，产生了以 PB 为数量级的非结构化数据信息。据 Internet World 统计，2017 年互联网用户达到 41.57 亿 [1], 在社交网络（如 Facebook）等新应用中，2017 年月平均活跃用户达到了 21.2 亿，相比 2016 年增长了 13.9% [2]。另外，数据处于高度变化状态，据统计 Facebook 每天新增用户超过 50 万，每分钟产生 51 万新评论，29.3 万用户状态更新, 每天新增数据量达到 4PB [3, 4]。这些统计数据说明数据的规模越来越大。数据已成为互联网时代的先进生产力，成为支撑基础科学研究和上层各类应用服务至关重要的战略资源，社会计算进入“大数据”时代。

自 1970 年，关系数据模型 (Relational model) 理论被提出后，由于其具有简单明确的模型特征和坚实的理论基础，建立于关系数据模型上的关系数据库 (Relational database) 很快被广泛应用，至今仍为数据库应用的主流。其本质是将数据按预先设计好的关系模式组织为若干个按行与列存储的数据表格。然而，随着互联网的飞速发展，关系数据库逐渐暴露出其固有的缺陷和问题。现如今，网络及商业应用生成了大量信息互联的数据，每个实体都与周围实体存在广泛的联系关系，这些联系关系里存在大量的潜在信息。然而关系数据模型更加注重刻画实体内部的属性，实体间的关系主要通过外键来实现。其存储模式是将不同类型实体存储为独立的表格，将一个实体在另一个实体表格中存储为一个键值，从而记录实体之间的关系。这种存储模式使得关系数据库已经越来越难以承载查询海量数据深层次关系的需求。

具体而言，（1）关系数据库在处理涉及到实体间相关关系的查询时效率低下。当数据库查询涉及到两个实体之间相关关系时，关系数据库需要借助其连接操作（Join）将两个表格中的相关实体链接到一起，进而执行后续查询操作。然而由于连接操作复杂度高，通常非常耗时。当数据库查询涉及到多个实体间的相关关系时，需要将这些实体所在表格依次执行连接操作，多次连接操作将导致关系数据库在大型数据集上执行查询操作的效率极其低下。（2）关系数据库提供的查询模式（如 SQL 查询语句）表达能力有限，无法表达实体间深层次关系的查询，如实体间可达性查询。因为 SQL 查询语句只能将已知个数有限个实体之间关系用连接操作表达，而不能将未知个数多个实体间的关

系用连接操作表达出来。事实上，SQL 是基于一阶逻辑设计，而理论上可证明一阶逻辑无法表达可达性查询 [5]。

1.1.1　图数据、图查询和图匹配查询

与关系数据模型相比，图（Graph）在描述和刻画复杂的现实世界方面具有更强的表达能力。图可以将现实世界中的实体抽象为图中的节点，将实体间的关系抽象为节点之间的边，节点和边上通常有表示实体或关系相关信息的标签。图数据能直观地表达实体间的复杂关系，从而将现实世界直观建模。

图数据的优势主要有以下几个方面：（1）图数据的一个重要特征是为每个节点存储与其相连的所有邻接节点的索引。当处理涉及到实体间关系的查询时，在图数据上可以利用邻接节点索引在常数时间内获取出一个实体的某个相关实体。所以，实体关系的查询时间与图数据的整体规模无关，只与邻接节点的数量成正比。图数据的这种数据存储方式避免了传统关系数据库中表格间的连接操作，减少了大量计算成本。（2）由于图数据以图论的存储模式为基础，所以图数据可以支持图论中被广泛研究的丰富的查询模式，如可达性查询 [6, 7]、邻接查询 [8]、最短路径查询 [9, 10]、子图同构查询 [11, 12]、图模拟 [13] 及其扩展强模拟查询 [14, 15] 等。这些基于图论的查询模式可以直接表达出现实世界的查询需求，通常代表人们对世界认知的一种高效、直观、自然的抽象描述。然而，传统关系数据模型中的查询模式需要将查询需求转化成按照严格语法规定的统一的结构化查询语言，这种实际查询需求与严格规定的语法之间的落差必然导致有价值信息的丢失。比如关系数据查询无法表达可达性和最短路径等查询 [5, 7, 9, 10]。（3）相较于结构化关系数据模型，非结构化图数据具有更强的动态性和可扩展性，可以承载海量数据存储、数据量迅猛增长、数据来源种类不断增多等带来的挑战。正是由于图数据可以直接描述各种复杂的现实世界系统，所以被广泛地应用于各个领域。在社交网络 [16, 17]、生物数据分析 [18, 19]、路径规划 [9, 10]、推荐系统 [20, 21]、复杂对象识别 [22, 23] 和软件代码剽窃检测 [24] 等领域都起着重要作用。

在当今“大数据”时代，如何从海量数据中获取、归纳信息，从而进行快速、准确的判断和决策已经成为各类应用中最迫切的需求。图在各个领域中日益增长的应用使得图查询（从图中查询信息）自然而然的成为学术界和工业界的共同研究热点。

图查询指图数据上的查询语言。下面首先给出图查询的一个抽象的形式化定义 [25, 26]。给定一个模式图 Q 和一个数据图 G：判断 Q 是否“匹配”G，并且从 G 中

找出所有与 Q“匹配”的子图。这里的图由节点集和边集构成，节点上通常有表示实体有关信息的标签。另外，模式图的规模较小，仅由几个至几十个节点和边组成；但是数据图的规模非常大，如上所述，现如今已包含以“亿”为数量级的节点和边。根据（1）模式图的拓补结构的不同和（2）“匹配”语义的不同形成不同类型的图查询语言。首先，根据模式图的不同拓补结构可以分为：点查询，路径查询和图匹配查询。点查询指当模式图 Q 为一个节点，在数据图 G 中查找满足 Q 中所述匹配条件的节点集合；路径查询指当模式图 Q 为两个节点，在数据图 G 中查找满足 Q 中两个节点间所述匹配条件的路径；图匹配查询指当模式图 Q 为一个图结构，在数据图 G 中查找满足 Q 中所述匹配条件的子图结构集合。其次，根据“匹配”语义的不同，以上三类查询语言又包括不同类型的图查询：点查询包括邻居节点查询 [8] 等；路径查询包括最短路径查询 [9, 10] 和可达性查询 [6, 7] 等；图匹配查询包括子图同构查询 [11, 12]、图模拟查询 [13] 及其扩展强模拟查询 [14, 15] 等。图查询语言中，点查询和路经查询已经有了较为成熟的研究和高效的算法，本文主要关注于图查询中最复杂的一类查询：图匹配查询（Graph pattern matching）。后文中提到的图查询在未指明情况下均指图匹配查询。

图匹配查询理论和技术为用户从海量的非结构化数据中获取信息提供了一种方便快捷的搜索方式，在数据清洗中的复杂对象识别 [22, 23]、社交网络和 Web 网络查询 [16, 17]、生物数据分析 [18, 19] 和软件代码剽窃检测 [24] 等方面都有重要的应用（参见综述 [27]）。比如，在数据清洗中，复杂对象识别即如何识别出表示同一实体的复杂对象是一个非常困难的问题。图匹配查询为复杂对象识别提供了解决方案。通过将复杂对象表达为子图结构模式图 Q，即可采用图匹配查询技术在数据图 G 中有效的发现和识别出与 Q 匹配的子图集合，这些子图表示相同的实体，从而达到复杂对象识别 [11, 13, 23]。

1.1.2 图匹配查询面临的问题

图匹配查询在各个领域中已有日益增长的应用。然而，相对于研究较为成熟的关系数据查询和 Web 网络搜索技术，图匹配查询技术尚处于研究的初始阶段，存在很多问题与挑战。

对于一个查询引擎而言，最重要的三个原则为：准确性、高效性和易用性 [25]。（1）准确性是数据库和信息检索等领域所关注的最基础问题。其要求查询引擎输出的结果与用户的查询输入匹配，并且能够提供多种匹配语义满足用户在不同应用场景和不同应用模型下的需求。（2）高效性要求查询引擎能够在较短时间内查询得到正确的查询结果。

高效稳定的数据库系统是上层应用的基础支撑，特别是在“大数据”时代，高效是所有查询引擎技术必须保证的特性。（3）易用性要求查询引擎能够提供用户友好的查询技术，使得用户在查询引擎中表达出他们的查询意图后，查询引擎能够根据用户的查询意图返回令用户满意的结果。本文将从这三个方面分析图匹配查询存在的问题。

（1）在图匹配查询准确性方面。图匹配查询的准确性方面已经有大量相关研究 [11–15]。如前文中提到，尽管查询模式图 Q 结构都一样，由于“匹配”语义的不同，图匹配查询形成了基于子图同构 [11, 12]、图模拟 [13] 以及其扩展强模拟 [14, 15] 不同的图匹配查询语义，这些查询语义具有不同的查询准确度。其中，子图同构是图匹配查询的最基础语义。该语义要求在数据图中查找出与模式图拓补结构完全相同的子图。然而，因为子图同构语义的匹配约束条件最严格，经常导致模式图在数据图中查询结果较少甚至为空集 [14, 27, 28]。另外，基于子图同构的图匹配查询问题是NP-完全问题 [29]，由于时间复杂度较高而不具实用性；为了降低时间复杂度和增强查询结果实用性，后续工作提出了图模拟匹配语义 [13]，放松了子图同构中极为严格的结构匹配要求，将时间复杂度降为二次方多项式（PTIME）时间。然而该语义对结构匹配要求过于松散，导致查询结果与模式图差异较大，查询准确度较低；而后，基于图模拟的语义被相继提出，如强模拟语义 [14, 15] 等。相较于图模拟语义，强模拟大幅提高了查询准确度，并且保证时间复杂度为三次方多项式时间。这些图匹配查询语义在查询准确度和计算复杂度之间做出了不同的平衡，使得用户可以根据具体应用需求选择合适的查询语义。然而，现有图匹配查询语义仍然存在实用性较低、难以表达实际应用的查询需求、以及查询结果过少等问题。

（2）在图匹配查询高效性方面。尽管图匹配查询相继提出了时间复杂度较低的图匹配语义，其计算效率依然较低而不具备实用性。大数据时代颠覆了传统的计算模式，即使传统意义上认为效率较高的计算模式（如线性时间），在大数据背景下也会需要极高的计算开销。例如，假设使用现有存储与读取性能最优的 SSD 固态硬盘 [30]，当读取速度为 12GB/s 时，线性扫描一个 30TB 的数据集需要花费 41 分钟。如上文所述，即使是时间复杂度最低的图匹配查询语义（图模拟语义），其时间复杂度仍然达到二次方多项式时间，而且图模拟语义的查询结果准确度已经很低。这些事实提醒我们不能仅依赖于通过提出新的查询语义的方法以降低查询准确度为代价提升图匹配查询效率。我们需要提出新的图匹配查询“模式”来提高查询效率，而该“模式”是与语义无关或弱耦合的通用方法，能使现有的所有图匹配查询语义均受益。

而且，图匹配查询处于高度动态环境。数据图的更新非常频繁，并且每时每刻都

在发生更新。在社交网络等应用中每天更新的数据量已经达到以 10 万为数量级的规模 [3, 4, 31]。另外，对于所有类型查询引擎和所有类型数据集，迭代设计查询是一个常见的过程，即用户根据上一轮查询结果的满意程度多次反复调整查询输入，以期得到预期的查询结果 [32, 33]。然而，即使是时间复杂度最低的图匹配查询语义（图模拟语义），其时间复杂度仍然达到二次方多项式时间。每当数据图或模式图发生更新，重新执行图匹配查询计算过程这种更新处理方法计算开销太大，不具备可行性。在大数据时代，高度动态环境对高效查询引擎的要求日益增长。关于提高图匹配查询引擎在静态和动态环境的高效性是本课题的主要研究内容。

（3）在图匹配查询易用性方面。相较于图查询中的点查询和 Web 网络上的关键词搜索，图匹配查询的用户友好性较低，通常只能在数据图中查找到较少甚至无法得到任何结果。因为点查询和 Web 查询的查询输入为节点集合和关键词集合，其查询目标为与输入查询匹配的节点集合和网络链接列表。这种查询的匹配约束条件较少，在数据集中很容易查找到匹配结果。但是，图匹配查询的查询输入为模式图，包括点集和边集，其查询目标为与模式图拓补结构匹配的子图，这种要求拓补结构匹配的查询约束条件较强，更容易导致查询结果较少或为空集。另外，对于规模较大并且结构复杂的数据图，由于用户无法全面掌握数据图基本信息，导致用户很容易构造出不存在任何查询结果的模式图。这些原因必然导致用户无法轻易得到预期的图匹配查询结果。用户需要根据查询结果的状态反复调整查询模式图，甚至可能无法构造出能够得到预期查询结果的模式图。然而，图匹配查询所有语义的时间复杂度都较高，如此迭代反复调整模式图，必然触发多轮次高时间花费的图匹配查询过程。这些因素导致图匹配查询不具备易用性。关于提高图匹配查询引擎易用性的研究还处于空白状态，是本课题的重要研究内容。

综上所述，我们看到了大数据时代图的查询理论及相关技术具有重要的科学意义和应用价值。另外，我们看到了图匹配查询在当前大数据时代是一个亟待研究的科学问题。同时我们也看到了图匹配查询所面临的问题和挑战。

1.2 主要研究内容

本文以提高图匹配查询的准确性、高效性和易用性为目标。本文将以现有图匹配查询语义和算法为基础设计新的“通用图匹配查询计算方法”提供准确的、高效的和用户友好的图匹配查询方法。而该计算方法是与语义无关的通用方法，能够支持现有的所有图匹配查询语义。本文将以图论、数据库理论、计算复杂性理论和可近似性理论为理论工

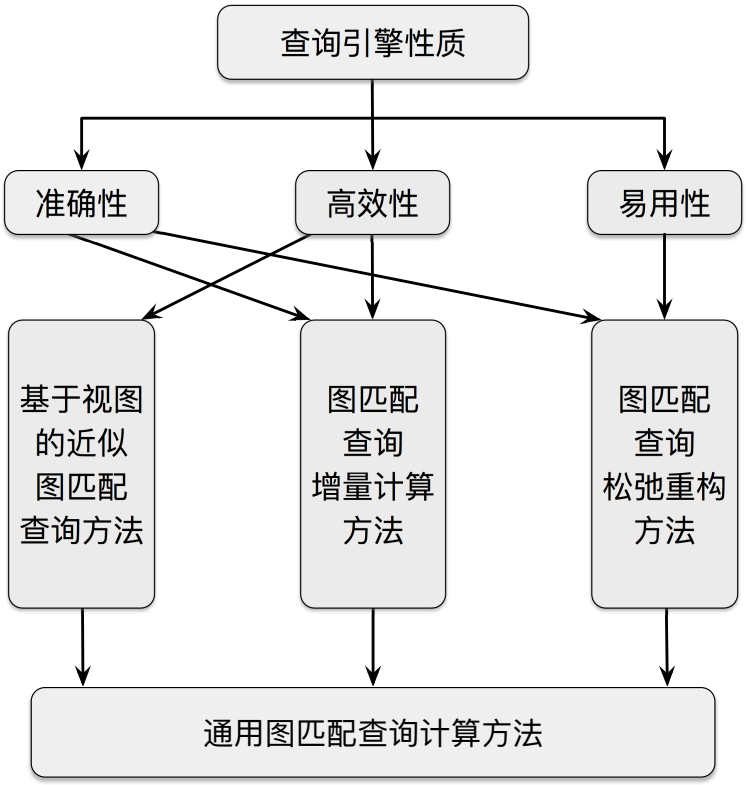


图 1　论文总体架构图

具，结合数据库和信息检索领域相关技术来解决现有图匹配查询方法中存在的问题，从理论分析到算法设计再到应用分析和实验验证多层次系统深入的研究图匹配查询方法。

论文主要从提高图匹配查询的高效性和易用性两方面开展，并同时提高图匹配查询的准确性。具体而言，论文分为以下三个研究点：研究点（1）和（2）是关于提高图匹配查询高效性的研究，研究点（3）是关于提高图匹配查询易用性的研究。其中，研究点（2）和（3）提出的方法能够同时提高图匹配查询的准确性。此外，本文提出方法都是通用的图匹配查询计算方法，是新的图匹配查询计算“模式”，而该“模式”能使所有图匹配查询语义均受益。论文总体架构如图 1 所示。

（1）基于视图的近似图匹配查询方法。针对图匹配查询所有图匹配语义在大数据上计算复杂度过高的问题，本文提出一种通用的基于视图的近似图匹配查询方法。现有工作提出了基于视图的图匹配查询方法，将数据库领域常用的视图查询技术引入了图匹配查询来提高查询效率。然而由于存储视图的物理空间有限，并且由于图匹配查询的模式图表达的查询条件比较复杂，匹配约束条件非常严格，所以导致大部分模式图无法满足视图回答的命中条件，造成视图的命中率和利用率都很低。本文提出基于视图的近似图匹配查询方法，该方法通过使用视图技术来加速图匹配查询计算过程，并引入上界和下界近

似查询的概念来提高存储视图的命中率和利用率，从而提高图匹配查询引擎的效率。

（2）图匹配查询增量计算方法。针对图匹配查询中模式图和数据图频繁更新导致动态环境下图匹配查询计算复杂度过高的问题，本文提出一种统一的图匹配查询增量计算方法，可以处理连续的单独或混合的模式图和数据图更新。该方法在统一处理模式图和数据图更新时可以复用已有结果，仅计算依赖于更新数据上的查询结果，并与原有查询结果合并即可得到模式图和数据图动态更新后的图匹配查询结果。该方法基于模式图划分和标识更新影响区域的增量计算策略，从而将模式图和数据图的更新影响范围最小化和局域化。该方法避免每次动态更新时需要重新计算查询结果的昂贵计算开销，从而提高动态环境下图匹配查询引擎的效率。另外，本文提出了组合模拟图匹配语义和基于组合模拟语义的 top-k 图匹配查询方法。相较于传统图匹配查询语义，该匹配语义更具实用价值，能够表达社交推荐和团队搜索等实际应用的查询需求，提高了图匹配查询引擎的准确性。

（3）图匹配查询松弛重构方法。针对图匹配查询语义约束条件较严格经常导致图匹配查询结果较少甚至为空集的问题，本文提出一种通用的图匹配查询松弛重构方法。该方法通过利用信息检索领域常用的查询重构技术对图匹配查询过程进行优化，使得图匹配查询引擎能够根据用户输入的模式图自动对模式图进行松弛重构，即生成一系列松弛重构模式图并同时返回所有图匹配查询结果，保证用户能够得到满意的查询结果。查询引擎如何重构模式图以满足用户的期望，查询引擎如何在数据图中快速查找所有重构模式图的图匹配结果，以及查询引擎如何为用户解释新的查询结果产生的原因都是提高图匹配查询引擎易用性所需解决的关键问题。其中，该方法对新结果提供溯源解释的功能保证了图匹配查询的准确性。另外，本文提出了基于语义分类信息的泛化图匹配查询方法。对于任意一种图匹配查询语义，该方法通过将表达标签间类别从属关系的标签语义分类信息引入该图匹配查询语义来减弱图匹配查询中要求标签完全相同的匹配约束条件，从而得到更多合理有效的图匹配查询结果，提高了图匹配查询引擎的准确性。

1.3 课题来源

本课题来源于国家重点基础研究发展计划（973 计划）网络信息空间大数据计算理论（2014CB340300）子课题“大数据的计算复杂性与算法理论”以及国家自然科学基金委优秀青年科学基金资助“数据库理论与系统”（61322207）。

1.4　论文结构

本文以提高图匹配查询的准确性、高效性和易用性为目标，按照章节 1.2 中的三个研究点展开，具体如下：

第一章，主要介绍本文的研究背景和图匹配查询面临的问题，并提出本文的研究目标和主要工作内容。

第二章，主要介绍和分析与本文相关的国内外研究工作的现阶段研究成果，包括各类图匹配查询语义介绍、图匹配查询的典型算法分析以及图匹配查询优化技术分析。

第三章，主要针对图匹配查询在大数据上计算复杂度过高的问题，本文提出一种基于视图的近似图匹配查询方法，通过利用视图查询方法来提高查询效率，并利用近似查询技术提高存储视图的命中率和利用率，从而提高图匹配查询引擎的效率。

第四章，主要针对图匹配查询中模式图和数据图的频繁更新导致动态环境下图匹配查询计算复杂度过高的问题，本文提出一种统一的图匹配查询增量计算方法，可以处理连续的单独或混合的模式图和数据图更新,从而提高动态环境下图匹配查询引擎的效率。

第五章，主要针对图匹配查询语义约束条件较严格经常导致图匹配查询结果较少甚至为空集的问题，本文提出一种通用的图匹配查询松弛重构方法。通过利用信息检索领域的查询重构技术使得图匹配查询引擎能够自动对模式图进行松弛重构并返回所有查询结果，保证用户能够得到满意的查询结果，从而提高查询引擎的易用性。

第六章，作为结束语章节，对本文的主要内容和贡献进行了总结，同时讨论了未来可能的研究问题和方向。

为了论述简洁，本文只将部分定理证明列于正文中，章节三、章节四和章节五的其他定理证明分别详见附录 A、附录 B 和附录 C。

第二章　图匹配查询相关技术综述

本章首先介绍图匹配查询的基本概念：图匹配查询的三个主要匹配语义（章节 2.1）和相应图匹配查询算法（章节 2.2）。而后，介绍用于优化图匹配查询计算过程的相关技术（章节 2.3）。本章在介绍相关技术的同时对现有技术中存在的问题进行分析。

2.1　图匹配查询语义介绍

首先，模式图和数据图的详细定义如下所示。

定义 2.1 (模式图和数据图): 模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$ 是分别由节点集和边集构成的有向图，节点上有表示实体有关信息的标签。例如，数据图 G 的节点集合为 V_G；边集为 E_G，并且 $E_G \subseteq V_G \times V_G$；$l_G$ 为标签函数将 V_G 中任意一个节点 v 映射至节点标签集合 Σ 中的一个标签。另外，在上下文已知的情况下通常将数据图 $G(V_G, E_G, l_G)$ 简写为 $G(V_G, E_G)$ 或 G。数据图 G 的大小 $|G|$ 为其节点数目和边的数目之和 $|V_G|+|E_G|$。模式图的定义和表示同理可知。 □

图匹配查询即给定一个模式图 $Q(V_Q, E_Q)$ 和一个数据图 $G(V_G, E_G)$，从 G 中找出所有与 Q 相“匹配”的子图。根据模式图 Q 与数据图 G “匹配”语义的不同形成不同的图匹配查询语言（参见综述 [27]），主要包括基于子图同构匹配 [11, 12]、基于图模拟匹配 [13, 28] 和基于强模拟匹配 [14, 15] 等。本课题提出的用于提高图匹配查询高效性和易用性的方法都是通用方法，可以应用于任意一种图匹配查询语言以提高其查询效率和查询友好性。为了便于介绍和论证，本文在第三、四和五章详细论述这些方法时是基于以上三种典型的图匹配查询语言。本章将详细介绍这三种图匹配查询语言的定义。

2.1.1　基于子图同构的图匹配查询

基于子图同构的图匹配查询要求从数据图 G 中找出所有与 Q 拓补结构完全相同的子图，子图同构是图匹配查询的最基础语义。下面给出子图同构的定义 [11, 12]。

定义 2.2 (子图同构): 给定模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$，则 Q 与 G 通过子图同构匹配，记为 $Q \lhd G$，当且仅当 G 中存在一个子图 G_s 和一个双射函数 f，使得

（1）对于 Q 中每个节点 u，在 G_s 中都存在一个节点 $f(u)$ 与 u 具有相同节点标签，即 $l_Q(u) = l_G(f(u))$；并且

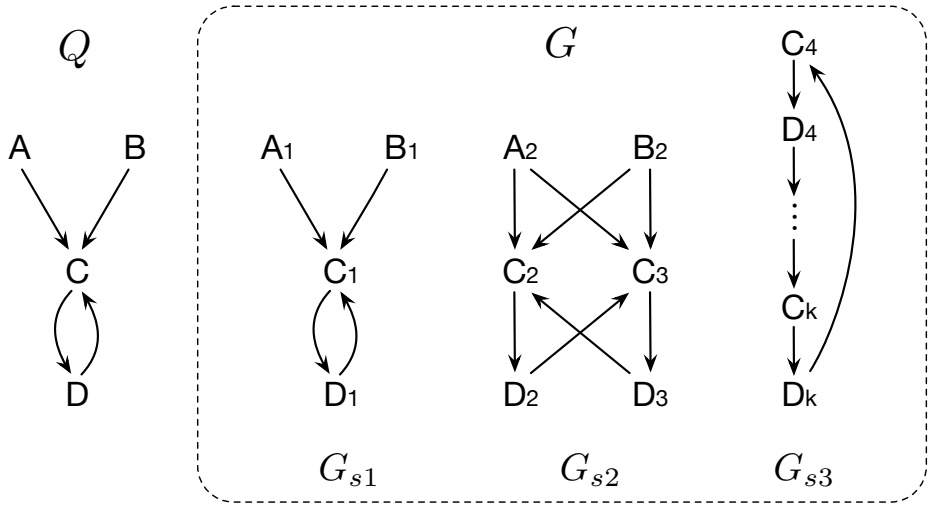


图 2　图匹配查询示例

（2）Q 中存在一条边 (u, u')，当且仅当 G_s 中存在一条对应的边 $(f(u), f(u'))$。

其中，G_s 被称为模式图 Q 在 G 中的匹配子图。 □

直观理解，模式图 Q 与数据图 G 通过子图同构匹配当且仅当 G 中存在一个与 Q 一模一样的子图 G_s，如例 2.1 所示。

例 2.1: 给定模式图 Q 和数据图 G，如图 2 所示。Q 与 G 基于子图同构语义匹配，因为 G 中存在子图 G_{s1} 与 Q 具有完全相同的拓补结构和相同的对应节点标签，并且 Q 在 G 中只存在 G_{s1} 这一个基于子图同构的匹配子图。 □

然而，子图同构语义的计算复杂度较高。基于子图同构的图匹配查询问题是NP-完全问题 [29]，因此查询效率较低。通过理论分析可知，数据图 G 中可能存在指数多个与模式图 Q 匹配的结果。然而，由于子图同构要求数据图中存在子图与模式图具有完全相同的拓扑结构，严格的查询条件经常导致模式图在数据图中的查询结果较少甚至为空集。这些因素限制了子图同构的应用范围。这些缺点促使相关研究寻找更具实用性的图匹配语义，本文将在后续章节中陆续介绍。

2.1.2　基于图模拟的图匹配查询

通过章节 2.1.1 对子图同构语义的介绍，我们知道基于子图同构的图匹配查询问题是 NP-完全问题，严格的匹配约束条件导致该语义查询效率较低并且很难查询得到匹配结果从而不具备实用性。随着应用中图数据规模的不断扩大，需要找到高效率及高性能的图匹配查询语义。相关研究通过减弱子图同构语义的匹配约束条件来提高图匹配查询

的实用性。其中，图模拟被提出作为图匹配查询语义 [34]。下面给出图模拟的定义 [13]。

定义 2.3 (图模拟): 给定一个模式图 $Q(V_Q, E_Q, l_Q)$ 和一个数据图 $G(V_G, E_G, l_G)$，则 Q 与 G 通过图模拟语义匹配，记为 $Q \prec G$，当且仅当存在一个节点集合之间的二元关系 $R \subseteq V_Q \times V_G$，使得

（1）对于每对节点 $(u, v) \in R$, 节点 u 和节点 v 有相同的标签，即 $l_Q(u) = l_G(v)$；并且

（2）对于 Q 中每个节点 u，在 G 中都存在一个节点 v，使得（a）$(u, v) \in R$，并且（b）对于 Q 中每条边 (u, u')，在 G 中都存在一条边 (v, v')，使得 $(u', v') \in R$。

其中，G 中会存在多个与 Q 匹配的二元关系，但只存在一个最大二元匹配关系 R_M，使得对于 G 中所有与 Q 匹配的二元关系 P，P 都是 R_M 的子集，即 $P \subseteq R_M$。最大二元匹配关系 R_M 中所有数据图 G 中的节点和邻接边构成的子图即为 Q 在 G 中的匹配图。 □

直观理解，图模拟语义将子图同构中要求拓补结构一模一样的匹配约束条件，转变为只保持父子关系匹配，即模式图与匹配图中的两个节点匹配当且仅当两个节点的所有子节点匹配，如例 2.2 所示。

例 2.2: 给定模式图 Q 和数据图 G，如图 2 所示。Q 与 G 基于图模拟语义匹配，因为存在基于图模拟的最大二元匹配关系 $R_M \subseteq V_Q \times V_G$, 将 Q 中的节点 A、B、C 和 D 分别匹配至 G 中的节点 $\{A_1, A_2\}$、$\{B_1, B_2\}$、$\{C_1, C_2, C_3, C_4, \ldots, C_k\}$ 和 $\{D_1, D_2, D_3, D_4, \ldots, D_k\}$。也就是说，$G$ 中所有节点都属于 R_M，Q 在 G 中基于图模拟的匹配图即为 G 自身。

由例 2.1 可知，当使用子图同构匹配语义时，Q 在 G 中的匹配结果只包括子图 G_{s1}；当使用图模拟语义时，Q 在 G 中的匹配结果包括子图 G_{s1}、G_{s2} 和 G_{s3}。 □

由定义可知，图模拟语义将子图同构语义定义中要求模式图与数据图结构对应的双射函数减弱为二元关系对应，相应由子图同构问题的 NP-完全计算复杂度转变为图模拟问题的多项式时间可解（PTIME ）计算复杂度。基于图模拟的图匹配查询问题存在二次方多项式时间的算法 [13]。然而，低时间复杂度是以牺牲查询结果准确度为代价的。图模拟语义松散的结构匹配约束条件导致匹配图与模式图存在较大的结构差异。另外，根据图模拟定义可知模式图 Q 在数据图 G 中只存在一个匹配图，而图模拟语义松散的结构匹配约束条件导致匹配结果规模通常较大，所以该匹配语义通常会查询得到一个规模较大、不便于分析理解的查询结果。例如，如例 2.2 所示，图模拟语义松散的结构匹配约束条件导致模式图 Q 在数据图 G 中的匹配结果为整个数据图 G；此外，匹配结果中由节点 $C_4, D_4, \ldots, C_k, D_k, C_4$ 构成的环状子图 G_{s3} 与模式图 Q 存在较大结构差异。为了

提高图匹配查询结果的准确度，各种图模拟的扩展语义被相继提出 [6, 14, 28]，本文将在下一章节中重点介绍一种图模拟的扩展语义。

2.1.3　基于强模拟的图匹配查询

通过章节 2.1.1 对子图同构语义的介绍，我们知道基于子图同构的图匹配查询问题是NP-完全问题，严格的匹配约束条件导致该语义查询效率较低并且很难查询得到匹配结果从而不具备实用性；通过章节 2.1.2 对图模拟语义的介绍，我们知道基于图模拟的图匹配查询问题是二次方多项式时间可解问题，虽然该语义查询效率较高但是查询结果准确度较低。随后，相关研究在图模拟语义的基础上提出了强模拟语义 [14, 15]，该语义增强了图模拟的结构匹配约束条件，在子图同构和图模拟语义之间做出了平衡，也就是在查询效率和查询结果准确度之间做出了平衡。

为了介绍强模拟的定义，首先需要介绍对偶模拟的定义 [14]。

定义 2.4 (对偶模拟): 给定模式图 $Q(V_Q, E_Q)$ 和数据图 $G(V_G, E_G)$，则 Q 与 G 通过对偶模拟匹配，记为 $Q \prec_D G$，当且仅当

（1）$Q \prec G$，即 Q 与 G 通过图模拟匹配，匹配二元关系为 $R \subseteq V_Q \times V_G$；并且

（2）对于任意一对匹配节点 $(u, v) \in R$ 和模式图中的一条边 $(u_2, u) \in E_Q$，存在数据图中的一条边 $(v_2, v) \in E_G$ 使得 $(u_2, v_2) \in R$。

其中，G 中会存在多个与 Q 匹配的二元关系，但只存在一个最大二元匹配关系 R_M，使得对于 G 中所有与 Q 匹配的二元关系 P，P 都是 R_M 的子集，即 $R \subseteq R_M$。最大二元匹配关系 R_M 中所有数据图 G 中的节点和邻接边构成的子图即为 Q 在 G 中的匹配图。 □

直观理解，对偶模拟将图模拟中要求保持父子关系匹配的结构匹配约束条件，增强为既保持父子关系匹配，又保持子父关系匹配的结构匹配条件，即模式图与匹配图中的两个节点匹配当且仅当两个节点的所有子节点和所有父节点都匹配。然而，基于对偶模拟语义的匹配结果存在和图模拟语义相同的问题，即匹配图规模通常较大导致查询结果不便于分析与使用。强模拟在对偶模拟的基础上增加了结果的局域性要求，其要求匹配子图节点之间的距离在一定范围内。为了实现这个目的，强模拟引入了球结构概念。

定义 2.5 (球结构): 对于数据图 G 中一个节点 v，和一个非负整数 r，以 v 为球心以 r 为半径的球 $\hat{G}[v, r]$ 是 G 的一个子图，使得

（1）对于球 $\hat{G}[v, r]$ 中的所有节点 v'，v' 和 v 之间的最短距离小于等于 r，即 $\mathsf{dist}(v, v') \leq r$。

（2）球 $\hat{G}[v, r]$ 含有 G 中所有相同节点的所有邻接边。 □

下面给出强模拟的定义[14]。

定义 2.6 (强模拟): 给定模式图 $Q(V_Q, E_Q)$ 和数据图 $G(V_G, E_G)$，则 Q 与 G 通过强模拟匹配，记为 $Q \prec_D^L G$，当且仅当 G 中存在一个节点 v 和一个连通子图 G_s，使得

（1）$Q \prec_D G_s$，即 Q 与 G_s 通过对偶模拟匹配，最大二元匹配关系为 $R_M \subseteq V_Q \times V_{G_s}$；

（2）G_s 是对应于最大二元匹配关系 R_M 的 Q 在 G 中的匹配图；

（3）G_s 被包含于球结构 $\hat{G}[v, d_Q]$ 中。其中，d_Q 是 Q 的直径。

其中，G_s 被称为模式图 Q 在 G 中的完美匹配子图。 □

直观理解，强模拟语义增强了图模拟语义的结构匹配约束条件，其要求在（i）保持图模拟要求的节点父子关系匹配的同时还要保持节点的子父关系匹配，并且要求（ii）匹配图的所有节点和边都必须在以 Q 的直径为半径的球结构内，这一局域性要求去除了大量冗余的匹配结果。如例 2.3 所示。

例 2.3: 给定模式图 Q 和数据图 G，如图 2 所示。Q 与 G 基于强模拟语义匹配，Q 在 G 中存在两个匹配子图，G_{s1} 和 G_{s2}。

（1）对于匹配子图 G_{s1}，验证可知：（a）$Q \prec_D G_{s1}$，最大二元匹配关系 $R_{M1} \subseteq V_Q \times V_{G_{s1}}$ 将 Q 中的节点 A、B、C 和 D 分别匹配至 G_{s1} 中的节点 $\{A_1\}$、$\{B_1\}$、$\{C_1\}$ 和 $\{D_1\}$；并且（b）以 C_1 为球心，以 2（Q 的直径）为半径的球恰好为 G_{s1}。

（2）对于匹配子图 G_{s2}，验证可知：（a）$Q \prec_D G_{s2}$，最大二元匹配关系 $R_{M2} \subseteq V_Q \times V_{G_{s2}}$ 将 Q 中的节点 A、B、C 和 D 分别匹配至 G_{s2} 中的节点 $\{A_2\}$、$\{B_2\}$、$\{C_2, C_3\}$ 和 $\{D_2, D_3\}$；并且（b）以 C_2 为球心，以 2（Q 的直径）为半径的球恰好为 G_{s2}。

另外，与图模拟语义不同，G 中由节点 $C_4, D_4, \ldots, C_k, D_k, C_4$ 构成的环状子图 G_{s3} 不能与 Q 基于强模拟语义匹配。 □

相较于图模拟语义，强模拟语义增强了结构匹配约束条件，极大程度提高了图匹配查询结果的准确度。相较于图模拟语义的二次方多项式时间复杂度，强模拟语义的时间复杂度为三次方多项式时间，然而相对于子图同构语义的 NP-完全时间复杂度，强模拟依然具有高效的查询效率。所以相关研究认为，强模拟语义在图模拟和子图同构语义之间做出了准确性和高效性的良好平衡[35, 36]。

通过分析图匹配查询语义的研究现状，我们发现图匹配语义具有多样性。相关研究在定义查询语言时在其表达应用需求的准确性和计算的高效性之间采取了不同的平衡策略，实际应用中需要根据具体应用需求采用合适的图匹配语义。

现有技术存在的问题。本节通过对图匹配查询各种匹配语义的分析得知以下两个问题。

（1）图匹配查询语义的计算复杂度普遍较高：基于子图同构的图匹配查询，该查询语义计算复杂度最高，因为子图同构问题为 NP-完全问题；基于图模拟的图匹配查询，该查询语义计算复杂度最低，为二次方多项式时间；基于强模拟的图匹配查询，该查询语义为三次方多项式时间。以上事实说明图匹配查询的计算开销较大，即使复杂度最低的图匹配查询语义（图模拟）仍然达到了二次方多项式时间。图匹配查询的效率问题是其在实际应用中的瓶颈问题。针对这一问题，本文将提出新的图匹配查询“模式”来提高图匹配查询效率，而该“模式”是与语义无关的通用方法，能使现有的所有图匹配查询语义均受益。

（2）图匹配查询语义的查询条件普遍较严格：图匹配查询的查询输入为由节点集和边集构成的模式图，其查询目标为数据图中与模式图拓补结构匹配的子图。这种要求拓补结构匹配的查询条件比较复杂，极易导致查询结果较少甚至为空集。另外，对于规模较大并且结构复杂的数据图，非专业用户由于无法全面掌握数据图基本信息，很容易构造出不存在任何查询结果的模式图。针对这一问题，本文将提出新的图匹配查询“模式”来提高图匹配查询易用性，保证用户能够查询得到期望的图匹配查询结果。同前者一样，本文提出的图匹配查询“模式”是与语义无关的通用方法，能使现有的所有图匹配查询语义均受益。

（3）图匹配查询语义的实用性仍然较低，难以表达实际应用的实际查询需求。本文将针对具体应用设计更具实用性的图匹配查询语义。

2.2　图匹配查询典型算法

本章将详细介绍子图同构、图模拟和强模拟这三种典型图匹配查询语义的图匹配查询算法。本文在第 三、四 和 五 章论述本课题提出的用于提高图匹配查询高效性和易用性的理论和方法时是以这些算法为基础。

2.2.1　基于子图同构的图匹配查询算法

子图同构问题是NP-完全问题中的经典问题 [29]，其最坏时间复杂度可以达到输入数据图的指数次方。相关研究提出多种启发算法以减少计算搜索空间（参见综述 [37]）。Ullmann 提出一个基于递归回溯的搜索算法 [11]，该算法通过一个有效的前瞻函数来减少搜索空间，是计算子图同构问题的经典算法。而后，[12] 在 Ullmann 算法的基础上提出了另一经典 VF2 算法，该算法提出了有效的启发策略用于优化节点匹配验证过程和

表 1　基于子图同构的图匹配查询算法 VF2

Algorithm：基于子图同构的图匹配查询算法 VF2

Data: 模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$。
Input: 一个中间状态 s；初始状态为 s_0，并且 $M(s_0) = \emptyset$。
Output: 模式图 Q 在数据图 G 中的匹配子图集合 Θ。

1. **if** $M(s)$ covers all the nodes of Q **then**　/* 查询得到一个 Q 在 G 中的匹配子图 */
2. 　Construct the match subgraph G_s *w.r.t.* $M(s)$; /* 根据节点匹配集合 $M(s)$ 构建匹配子图 */
3. 　$\Theta := \Theta \bigcup \{G_s\}$;
4. **else then**
5. 　Compute the set $P(s)$ of the pairs candidate for inclusion in $M(s)$;
6. 　**for each** pair $(u, v) \in P(s)$ **do**　/* 为每个 Q 和 G 的候选节点对，判定加入后能否匹配 */
7. 　　**if** the feasibility rules succeed for the inclusion of (u, v) in $M(s)$ **then**
8. 　　Compute the state s' obtained by adding (u, v) to $M(s)$;
9. 　　VF2(s');　/* 递归调用算法VF2 验证状态 s' 的后续状态能否满足匹配条件 */
10. 　Restore data structures;
11. **return** Θ;

顺序，该策略显著减少了搜索空间和占用内存。VF2 算法是迄今为止使用最广泛的子图同构算法，是图/网络数据分析软件包 igraph [38] 及程序语言开发源代码库 Boost 库 [39] 等采用的算法。本文采用 VF2 算法作为基于子图同构的图匹配查询算法，本章将对该算法主要过程进行详细介绍。

基于子图同构的图匹配查询算法 VF2。算法伪代码如表 1所示，该算法是递归回溯查找算法。算法的使用数据为模式图 Q 和数据图 G；算法的输入为一个中间状态，算法利用辅助数据结构 $M(s)$ 记录当前状态 s 下 Q 和 G 中当前匹配的节点对集合，初始状态 s_0 下 $M(s)$ 为空集；算法最终输出为模式图 Q 在数据图 G 中的匹配子图集合 Θ。对于任意一个中间状态 s，算法首先判定 $M(s)$ 中是否包含 Q 中所有节点及其在 G 中的匹配节点（第 1 行）。如果满足条件，则算法根据当前 $M(s)$ 中的节点对集合构建匹配子图 G_s 并将其加入匹配子图结果集合 Θ（第 2-3 行）; 如果 Q 中仍然存在节点在 G 中未找到匹配节点，则算法计算得到一个候补匹配节点对集合 $P(s)$，其包含 Q 中与已存在匹配的节

表 2　基于图模拟的图匹配查询算法 gSim

Algorithm：基于图模拟的图匹配查询算法 gSim

Input: 模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$。
Output: 模式图 Q 在数据图 G 中的匹配图 G_s。

```
1.  for each u ∈ V_Q do
2.      sim(u) := {v | v is in G and l_Q(u) = l_G(v)};
3.  while there are changes do
4.      for each edge (u, u') in E_Q and each node v ∈ sim(u) do
5.          if there is no edge (v, v') in G with v' ∈ sim(u') then
6.              sim(u) := sim(u) \ {v};
7.      if sim(u) = ∅ then return ∅;
8.  R_M := {(u, v) | u ∈ V_Q, v ∈ sim(u)};   /* 生成最大二元匹配关系 R_M */
9.  Construct the match graph G_s w.r.t. R_M;   /* 根据 R_M 构建匹配图 G_s */
10. return G_s;
```

点邻接但不存在匹配的节点以及其在 G 中的候补匹配节点组成的节点对（第 5 行）。对于 $P(s)$ 中任意一个节点对 (u, v)，算法判定在 $M(s)$ 中加入 (u, v) 后 $M(s)$ 是否还能够满足匹配可行规则。匹配可行规则包括语法判定和语义判定两类规则。如果满足所有可行规则，则生成新的状态 s'，并递归调用算法 VF2 验证状态 s' 的后续状态能否满足匹配条件（第 6-9 行）。算法在递归过程中不断维护当前状态的数据结构（第 10 行）并在回溯完毕后输出最终查询结果（第 11 行）。其中，候补节点对的验证顺序对搜索空间大小有重要影响，对于此步骤 (第 6 行) 的优化可以极大程度减少调用递归算法的次数 [12, 37]。

2.2.2　基于图模拟的图匹配查询算法

基于图模拟的图匹配查询问题是多项式可解问题，[13] 提出实现图模拟的二次方多项式时间算法。该算法核心思想为：首先初始化一个由数据图节点构成的候补匹配节点集合；而后设计迭代过程不断去除候补匹配节点集合中经过验证不满足图模拟匹配关系的节点；直至迭代过程收敛算法生成最终匹配节点集合，并根据最大二元匹配关系构建模式图在数据图中的匹配图。本章将对该算法主要过程进行详细介绍。

基于图模拟的图匹配查询算法 gSim。算法伪代码如表 2 所示，该算法输入为模式图 Q

和数据图 G，输出为模式图 Q 在数据图 G 中的匹配图 G_s。对于 Q 中任意一个节点 $u \in V_Q$，算法 gSim 首先计算得到 u 在 G 中具有相同标签的候补匹配节点集合 sim(u)，即 $l_Q(u) = l_G(v)$（第 1-2 行）。而后，算法不断循环为每个模式图节点 u，从 sim(u) 中去除经验证不匹配的数据图节点（第 3-7 行）。对于一个数据图节点 v，除非满足以下条件，否则算法将从 sim(u) 中将 v 去除：如果在 Q 中 u 有一个子节点 u'，那么在 G 中存在一个 v 的子节点 v' 使得 $v' \in$ sim(u')。最终，算法 gSim 根据最大二元匹配关系 R_M 构建出匹配图 G_s 并将其作为查询结果返回 (第 8-10 行)。

算法 gSim 的时间复杂度为 $O((|V_Q|+|E_Q|)(|V_G|+|E_G|))$。其中 $|V_Q|$ 和 $|E_Q|$ 分别为模式图 Q 的节点个数和边的个数；$|V_G|$ 和 $|E_G|$ 分别为数据图 G 的节点个数和边的个数。

2.2.3 基于强模拟的图匹配查询算法

基于强模拟的图匹配查询问题是多项式可解问题，[14, 15] 提出实现强模拟的三次方多项式时间算法。该算法核心思想为：在数据图中，为以每个节点为球心的球结构计算其与模式图基于对偶模拟的二元匹配关系；并根据二元匹配关系构建完美匹配子图，由所有完美匹配子图构成的集合则为最终查询结果。本章将对该算法主要过程进行详细介绍。

基于强模拟的图匹配查询算法 sSim。算法伪代码如表 3 所示，算法的输入为模式图 Q 和数据图 G。算法 sSim 为 G 中以每个节点 w 为球心以 Q 的直径 d_Q 为半径的球结构计算完美匹配子图。算法 sSim 最终输出所有完美匹配子图的集合 Θ。具体如下：对于 G 中任意一个节点 w，算法 sSim （1）首先调用对偶模拟查询过程 dSim 计算 Q 在球 $\hat{G}[w, d_Q]$ 中的最大二元匹配关系 R_M（第 3 行）；（2）而后构建球 $\hat{G}[w, d_Q]$ 中 Q 的完美匹配子图 G_s（第 4 行）；（3）如果 G_s 不为空，则将 G_s 并入结果集合 Θ 中（第 5 行）。最终，检查完毕所有球后返回查询结果 Θ（第 6 行）。

对偶模拟查询过程 dSim。过程伪代码如表 3 所示，输入为模式图 Q 和球结构 $\hat{G}[w, d_Q]$，输出为球 $\hat{G}[w, d_Q]$ 中 Q 的最大二元匹配关系 R_w。该过程扩展了基于图模拟的图匹配查询算法 gSim （表 2）。两者计算过程基本相同，详细步骤介绍参见算法 gSim 。不同之处在于，对偶模拟在图模拟结构匹配要求的保持节点的父子关系匹配外（表 2，表 3 第 4-6 行），增加了要求保持节点的子父关系匹配的约束条件（表 3 第 7-9 行）。

算法 sSim 的时间复杂度为 $O(|V_G|(|V_G|+(|V_Q|+|E_Q|)(|V_G|+|E_G|)))$。其中，过程 dSim 的时间复杂度为 $O((|V_Q|+|E_Q|)(|V_G|+|E_G|))$。

表 3　基于强模拟的图匹配查询算法 sSim

Algorithm：基于强模拟的图匹配查询算法 sSim

Input: 模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$。
Output: 模式图 Q 在数据图 G 中的完美匹配子图集合 Θ。

1. $\Theta := \emptyset$;
2. **for each** ball $\hat{G}[w, d_Q]$ in G **do**
3. $R_M := \mathsf{dSim}(Q, \hat{G}[w, d_Q])$;　/* 计算基于对偶模拟的最大二元匹配关系 R_M */
4. Construct the match graph G_m *w.r.t.* R_M, and find the connected component G_s containing w in G_m;　/* 构建完美匹配子图 G_s */
5. **if** $G_s \neq nil$ **then** $\Theta := \Theta \bigcup \{G_s\}$;
6. **return** Θ;

Procedure：对偶模拟查询过程 $\mathsf{dSim}(Q, \hat{G}[w, d_Q])$

Input: 模式图 $Q(V_Q, E_Q, l_Q)$ 和球 $\hat{G}[w, d_Q]$。
Output: 球 $\hat{G}[w, d_Q]$ 中模式图 Q 的最大二元匹配关系 R_M。

1. **for each** $u \in V_Q$ **do**
2. $\mathsf{sim}(u) := \{v \mid v$ is in $\hat{G}[w, d_Q]$ and $l_Q(u) = l_G(v)\}$;
3. **while** there are changes **do**
4. **for each** edge (u, u') in E_Q and **each** node $v \in \mathsf{sim}(u)$ **do**
5. **if** there is no edge (v, v') in $\hat{G}[w, d_Q]$ with $v' \in \mathsf{sim}(u')$ **then**
6. $\mathsf{sim}(u) := \mathsf{sim}(u) \setminus \{v\}$;
7. **for each** edge (u', u) in E_Q and **each** node $v \in \mathsf{sim}(u)$ **do**
8. **if** there is no edge (v', v) in $\hat{G}[w, d_Q]$ with $v' \in \mathsf{sim}(u')$ **then**
9. $\mathsf{sim}(u) := \mathsf{sim}(u) \setminus \{v\}$;
10. **if** $\mathsf{sim}(u) = \emptyset$ **then return** $\emptyset$;
11. $R_M := \{(u, v) \mid u \in V_Q, v \in \mathsf{sim}(u)\}$;　/* 生成最大二元匹配关系 R_M */
12. **return** R_M;

2.3 图匹配查询优化技术

本章主要介绍用于优化图匹配查询过程，提高图匹配查询引擎查询效率和易用性相关的查询优化技术，主要包括 (1) 视图查询技术、(2) 数据图增量查询技术、(3) 数据索引技术、(4) 数据压缩技术、(5) 数据采样技术和 (6) 查询重构技术。

本章首先对这些技术进行概述，而后再针对每项技术进行详细介绍。本文用 Q 表示模式图，G 表示数据图，$Q(G)$ 表示 Q 在 G 中的图匹配查询结果。

（1）视图查询技术（章节 2.3.1）：该技术的核心思想是将查询时间复杂度较高的一个模式图 Q 转换成若干在数据图 G 中查询结果已知的模式图 $Q'_1, Q'_2, \ldots, Q'_n$ 的组合，如图 3 所示。这些模式图在数据集中的查询结果 $Q'_i(G)$ 以视图形式存储在内存中，利用这些已知结果进行合并等相关处理，可以得到 Q 的最终查询结果。视图查询技术的目标是采用预处理提前构建存储视图的方式来提高输入查询的计算效率。本文将在章节 2.3.1 详细介绍该技术。

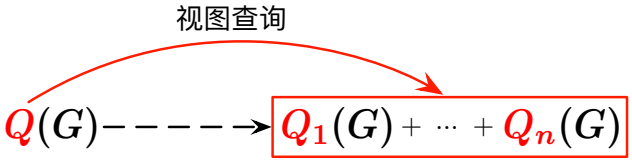


图 3　视图查询技术

（2）动态数据图增量查询技术（章节 2.3.2）：当数据图有更新时，通常需要重新计算查询结果以反映数据图的变化，然而在实践中数据图是经常被更新修改的，如图 4 所示。其中，Δ 表示数据图更新，当数据图更新时如果每次都要重新计算模式图 Q 在更新后的数据图 $G+\Delta$ 中的查询结果，这种方式的计算开销十分昂贵。增量查询的核心思想是通过复用已有查询结果以减少运算时间，仅计算这些“依赖于”改变数据上的结果即可。如图所示，增量查询只需重新计算 $Q(\Delta)$，并将其与已有查询结果 $Q(G)$ 进行合并，即可得到 Q 在 $G+\Delta$ 中的最终查询结果。本文将在章节 2.3.2 详细介绍该技术。

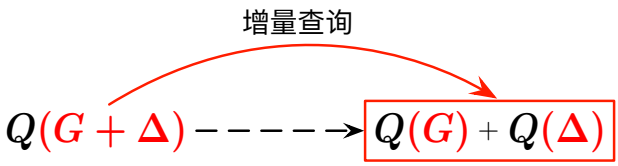


图 4　增量查询技术

（3,4,5）数据索引/压缩/采样技术（章节 2.3.3）： 这三类技术的核心思想是将模式图 Q 在较大规模数据图 G 上的查询转换成 Q 在较小规模数据图 G' 上的查询，如图 5 所示。其中，数据索引所采用的方法是通过预处理为数据图额外建立索引数据结构，当输入模式图 Q，访问索引结构后可以快速过滤掉 G 中大部分与 Q 显然不匹配的数据，使得只需访问小部分与 Q 相关的数据图 G' 得以计算得出准确查询结果；数据压缩所采用的方法是通过预处理去除掉数据图 G 上与输入查询无关的冗余数据，只存储与查询相关的小规模数据图 G'，使得 Q 在 G' 上可以高效率返回准确查询结果；数据采样所采用的方法是通过预处理从大规模数据图 G 进行采样并转换成一个保留 G 的关键特征信息的小规模数据图 G'，使得 Q 在 G' 上可以高效率返回精度可接受的近似查询结果。本文将在章节 2.3.3 详细介绍这三类优化技术。

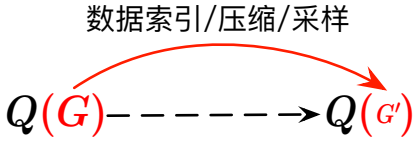


图 5　数据索引/压缩/采样技术

（6）查询重构技术（章节 2.3.4）： 该技术的核心思想是根据用户输入模式图 Q，在返回查询结果 $Q(G)$ 的同时为用户提供一组可以更好的捕捉用户搜索意图的替代模式图 $Q'_1, Q'_2, \ldots, Q'_n$ 和他们在数据图 G 中的查询结果供用户选择，如图 6 所示。查询重构技术的目标是通过捕捉用户搜索意图重构用户输入的查询，从而最小化用户输入工作量并使用户简单、快捷的得到期望的查询结果，以提高图匹配查询引擎的易用性。本文将在章节 2.3.4 详细介绍该技术。

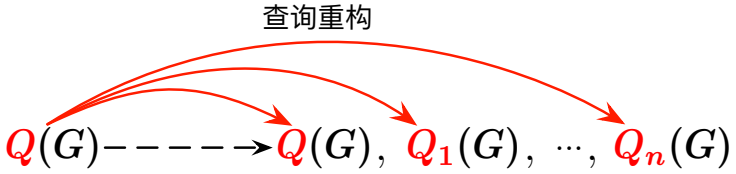


图 6　查询重构技术

2.3.1　图匹配查询的视图查询技术

视图技术被广泛应用于关系数据查询 [40–42] 和 XML 数据查询 [43–45] 等不同数据类型的应用中，使得查询引擎可以高效的从海量数据中快速获取相关信息，其有效性已经

得到了学术界和工业界的一致认可。[46, 47] 提出将视图技术扩展应用于优化图匹配查询过程，基于图模拟匹配语义。本章将详细介绍这项研究，其问题描述如下所示：

定义 2.7 (基于视图的图匹配查询): 给定视图集合 $\mathcal{V} = \{\mathsf{V}_1, \ldots, \mathsf{V}_n\}$，其中 V_i 为模式图结构。对于模式图 Q，如果存在另外一个模式图 A，使得对于任意数据图 G，（1）A 与 Q 等价，即 A 在 G 中的图匹配结果 $A(G)$ 与 $Q(G)$ 相同；并且（2）计算 $A(G)$ 只需要访问 $\mathcal{V}$ 中的视图 $\mathsf{V}_i \in \mathcal{V}$，以及他们相应的视图实例集合 $\mathcal{V}(G) = \{\mathsf{V}_1(G), \ldots, \mathsf{V}_n(G)\}$，即视图在 G 中已经计算并存储的图匹配查询结果，那么称模式图 Q 可以被视图集合 $\mathcal{V}$ 回答。□

由以上定义可知，如果模式图 Q 能够被视图 $\mathcal{V}$ 回答，那么只需访问视图实例 $\mathcal{V}(G)$ 即可回答出 Q 在数据图 G 中的查询结果 $Q(G)$，从而可以避免直接访问数据图 G 为 Q 计算查询结果这一时间花费较大的过程。这种视图回答方式相对于直接计算结果方式可以将查询效率提升几个数量级。然而，由于视图的物理存储空间有限，并非所有输入模式图 Q 都可以被视图回答。给定视图集合 $\mathcal{V}$，当输入一个模式图 Q 时，如何判定 Q 是否能够被 $\mathcal{V}$ 回答是基于视图的图匹配查询的核心问题。

[46, 47] 针对该问题给出了一个模式图能够被给定视图集合回答的特征描述。

图匹配查询的视图可回答性。给定一个模式图 $Q(V_Q, E_Q, l_Q)$ 和一个视图集合 $\mathcal{V} = \{\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_n\}$，其中 $\mathsf{V}_i = (V_{\mathsf{V}_i}, E_{\mathsf{V}_i}, l_{\mathsf{V}_i})$。$Q$ 能够被视图集合 $\mathcal{V}$ 回答当且仅当 $Q_s \sqsubseteq \mathcal{V}$（$Q_s$ 包含于 $\mathcal{V}$）。

其中，如果 $Q_s \sqsubseteq \mathcal{V}$ 成立，则存在一个从 Q 的边 E_Q 到所有视图的边的集合 $\bigcup_{i\in[1,n]} E_{\mathsf{V}_i}$ 的一个单射 λ 使得对于任意一个数据图 G，对于 Q 中任意一条边 $e \in E_Q$，都存在 $S_e \subseteq \bigcup_{e'\in\lambda(e)} S_{e'}$。回顾图模拟匹配语义定义可知，$R \subseteq V_Q \times V_{\mathsf{V}_i}$ 是 Q（或 V_i）和 G 节点之间基于图模拟匹配的二元关系，此处延伸定义对于 Q（或 V_i）中任意一条边 $e = (u, u')$，S_e 为 G 中与 e 匹配的边的集合，即若存在 G 中的一条边 $(v, v') \in S_e$，则 $(u, v) \in R$ 并且 $(u', v') \in R$。而后，[46, 47] 进一步给出了如何判定一个模式图能否被给定视图集合回答的充分必要条件。

视图可回答性充要条件。给定一个个视图集合 $\mathcal{V}$ 和一个模式图 $Q(V_Q, E_Q)$，$Q_s \sqsubseteq \mathcal{V}$ 成立当且仅当 $E_Q = \bigcup_{\mathsf{V}\in\mathcal{V}} M_{\mathsf{V}}^Q$ 成立。其中，对于任意一个视图 $\mathsf{V} \in \mathcal{V}$，将 V 视作“模式图”，将 Q 视作“数据图”，则如果 $\mathsf{V} \prec Q$ 成立，即 V 与 Q 通过图模拟语义匹配，那么对于 V 中任意一条边 e_{V}，$S_{e_{\mathsf{V}}}$ 为 e_{V} 在 Q 中的匹配边集合，M_{V}^Q 为 V 中所有边在 Q 中匹配边 $S_{e_{\mathsf{V}}}$ 的集合，即 $M_{\mathsf{V}}^Q = \bigcup_{e_{\mathsf{V}}\in E_{\mathsf{V}}} S_{e_{\mathsf{V}}}$。

至此，判断一个 Q 能否被 $\mathcal{V}$ 回答问题转化成判断 $E_Q = \bigcup_{V \in \mathcal{V}} M_V^Q$ 是否成立的问题。[46, 47] 提出算法 contain 用于判定可回答性充要条件 $E_Q = \bigcup_{V \in \mathcal{V}} M_V^Q$ 是否成立。该算法通过调用图模拟算法 gSim 为每个 $\mathcal{V}$ 中视图 V 计算其在 Q 中的匹配结果，并将匹配结果合并，最终判断以上充要条件是否成立。该算法时间复杂度为 $O(\|\mathcal{V}\| \cdot |Q|^2 + |\mathcal{V}|^2 + |Q||\mathcal{V}|)$。其中，$|Q|$ 表示 Q 的大小，即节点和边的个数之和；$|\mathcal{V}(G)|$ 表示 $\mathcal{V}$ 中所有视图在 G 中图匹配结果的大小之和；$\|\mathcal{V}\|$ 表示视图集合 $\mathcal{V}$ 中视图的个数。

当算法 contain 判定得知模式图 Q 可以被视图集合 $\mathcal{V}$ 回答，为了减少计算开销，需要从 $\mathcal{V}$ 中选取出一个视图子集 $\mathcal{V}_s$ 使得其既可以回答 Q，又具有较小的规模，从而通过访问视图实例 $\mathcal{V}_s(G)$ 可以计算得到 Q 在 G 中的图匹配查询结果 $Q(G)$。[46, 47] 提出图匹配查询视图回答算法 QAV 用于计算得出 Q 在 G 中的最终查询结果。

图匹配查询视图回答算法 QAV。该算法由两部分构成：（1）视图选择算法 minimalC 或 minimumC，用于从 $\mathcal{V}$ 中选取出一个可以回答 Q 的视图子集 $\mathcal{V}_s$；和（2）视图合并算法 viewJoin 通过合并 $\mathcal{V}_s(G)$ 中的视图实例计算得出 Q 在 G 中的图匹配查询结果。

算法 minimalC 的输入为视图集合 $\mathcal{V}$ 和模式图 Q，其目标是找出 $\mathcal{V}$ 中的一个视图子集 $\mathcal{V}_s$ 使得 $\mathcal{V}_s$ 可以回答 Q，并且不存在 $\mathcal{V}_s$ 的子集可以回答 Q。算法 minimalC 与算法 contain 类似，其将 $\mathcal{V}$ 中 V 在 Q 上的匹配结果合并后，去除掉冗余视图 V，并返回剩余的视图集合为视图子集 $\mathcal{V}_s$。该算法时间复杂度为 $O(\|\mathcal{V}\| \cdot |Q|^2 + |\mathcal{V}|^2 + |Q||\mathcal{V}|)$。

算法 minimumC 的输入为视图集合 $\mathcal{V}$ 和模式图 Q，其目标是找出 $\mathcal{V}$ 中的一个视图子集 $\mathcal{V}_s$ 使得 $\mathcal{V}_s$ 可以回答 Q，并且在 $\mathcal{V}$ 的所有可以回答 Q 的视图子集中，$\mathcal{V}_s$ 中视图个数最小。算法 minimumC 首先初始化 $\mathcal{V}_s$ 为空集，而后基于贪心算法策略每次选取可以回答 Q 中更多边的视图为 $\mathcal{V}_s$ 中视图，直至 $\mathcal{V}_s$ 可以回答出 Q 算法停止。该算法时间复杂度为 $O(\|\mathcal{V}\| \cdot |Q|^2 + |\mathcal{V}|^2 + |Q||\mathcal{V}|)$。

算法 viewJoin 通过合并视图实例 $\mathcal{V}_s(G)$ 中所有 $V_i(G)$ 来得到最终查询结果 $Q(G)$。该算法首先将 $\mathcal{V}_s$ 中所有视图实例 $V_i(G)$ 中与 Q 中每条边匹配的边取并集，得到初始化的匹配结果 M；再不断循环为每条模式图边，从 M 中去除经验证不匹配的数据图边，直至 M 不再发生变化并输出最终查询结果 M 即为 Q 在 G 中基于图模拟匹配的查询结果。该算法时间复杂度为 $O(|Q||\mathcal{V}(G)| + |\mathcal{V}(G)|^2)$。其中，$|Q|$ 表示 Q 的大小，即节点和边的个数之和；$|\mathcal{V}(G)|$ 表示 $\mathcal{V}$ 中所有视图在 G 中图匹配查询结果的大小之和。

现有技术存在的问题。基于视图的图匹配查询方法，由于其存储视图的物理空间有限，并且由于图匹配查询语义的模式图表达的查询条件较复杂，导致很多模式图无法满足视

图回答的命中条件，造成视图的命中率和利用率都很低。本文提出基于视图的近似图匹配查询方法，该方法根据视图的信息为模式图找到一对上界和下界近似查询。该方法利用视图查询方法来提高查询效率，并利用近似查询技术提高存储视图利用率来进一步达到提高查询效率目的。本文将提出通用的基于视图的近似图匹配查询方法的计算框架，适用于任意一种图匹配查询语义。

2.3.2 图匹配查询的动态数据图增量查询技术

增量计算技术即在数据动态更新情况下通过复用已有查询结果计算数据更新后结果从而提高查询效率技术，增量计算技术已经被广泛应用于各种应用领域中（参见综述 [48]）。[48, 49] 在增量计算理论研究方面做出了开创性工作。他们提出了增量算法计算复杂性理论，作为设计增量算法和衡量增量算法效率的指导性原则。

传统的用于评估算法计算复杂性理论是通过将算法的计算成本表示为算法当前输入大小的函数。该函数的渐近最坏情况所花费的时间即为该算法的时间复杂度。然而，如果用传统方法来评估增量算法的时间复杂度，对于很多现有增量算法，其时间复杂度并不低于直接计算算法的时间复杂度。也就是说，增量算法在其渐近最坏情况下的计算量至少等于甚至多于重新执行计算过程的计算量。所以，传统的计算复杂性理论无法衡量对于同一问题不同增量算法的好坏，从而不能作为算法的衡量标准指导增量算法的设计。针对这一问题，[48, 49] 提出利用增量算法输入和输出变化大小的总和来衡量算法的计算成本。其将增量算法的时间复杂度表示为算法输入和输出变化大小的总和 |CHANGED| 的函数。其中 |CHANGED| 表示处理数据更新需要花费的固有计算成本。[48, 49] 进一步定义有界增量算法如下。

定义 2.8 (增量有界性): 如果增量算法的时间复杂度可以表示为 |CHANGED| 的多项式函数，则称该算法为有界增量算法。如果一个增量问题存在有界增量算法，则称该问题为有界增量问题；如果增量问题不存在有界增量算法，则称该问题为无界增量问题。 □

[48, 49] 证明了边为正权重的单源最短路径和全源最短路径等问题都存在有界增量算法。而后，[28, 50, 51] 提出将该增量计算复杂性理论扩展应用于优化图匹配查询过程。首先，图匹配查询的动态数据图增量查询问题的描述如下所示:

定义 2.9 (图匹配查询的动态数据图增量查询): 给定模式图 Q，数据图 G，Q 在 G 中的图匹配查询结果 $Q(G)$。当数据图 G 上发生更新 ΔG 时，图匹配查询的动态数据图增量查询指通过计算 Q 在 ΔG 上的图匹配查询结果 $Q(\Delta G)$，并将其与 $Q(G)$ 合并，即可得到

Q 在更新后的数据图上的图匹配查询结果 $Q(G \oplus \Delta G)$，即 $Q(G \oplus \Delta G)=Q(G) \oplus Q(\Delta G)$。□

直观理解，相对于直接为模式图 Q 计算其在更新后数据图 $G \oplus \Delta G$ 上的图匹配查询结果，增量查询方法将与 $G \oplus \Delta G$ 相关的较大规模的计算量减小到仅与 ΔG 和 $Q(G)$ 相关的较小规模的计算量。[50, 51] 利用增量有界性理论来衡量图匹配查询的数据图增量查询问题的增量有界性。然而，他们发现增量有界性的定义过于严格，导致很难在实际应用中刻画出问题的增量计算复杂性。根据增量有界性定义，|CHANGED| 在图匹配增量查询问题中定义为 $|\Delta G| + |Q(\Delta G)|$，即输入的改变量 $|\Delta G|$ 和输出的改变量 $|Q(\Delta G)|$ 之和。[50, 51] 证明对于图匹配查询的典型匹配语义（图模拟和子图同构语义），其动态数据图查询问题均不存在有界增量算法。因为 |CHANGED| 并不能表示此问题中增量算法为了计算数据更新后的查询结果。

针对这一问题，[50, 51] 扩展了增量计算复杂性理论在图匹配查询中的定义。其提出影响域 AFF 概念，用来表示图匹配查询的动态数据图查询问题中增量算法处理数据图更新需要花费的固有计算成本。具体而言，影响域 AFF 不仅包括增量算法的输出改变量 $|Q(\Delta G)|$，还包括增量算法为了计算数据更新后的查询结果所必须维护的辅助数据结构的改变量。[50, 51] 定义相对有界增量算法如下。

定义 2.10 (增量相对有界性): 如果（1）增量算法的时间复杂度可以表示为 |AFF|、$|Q|$ 和 $|\Delta G|$ 的多项式函数，并且（2）增量算法需要维护的辅助数据结构大小可以表示为 $|G|$ 的多项式函数，则称该算法为相对有界增量算法。如果一个增量问题存在相对有界增量算法，则称该问题为相对有界增量问题。 □

[50, 51] 证明了对于基于图模拟的图匹配查询，其动态数据图查询问题（a）在处理单元（边或点）减更新和通用模式图时，存在时间复杂度为 $O(|AFF|)$ 的相对有界增量算法；（b）在处理单元（边或点）加更新和有向无环图模式图时，存在时间复杂度为 $O(|AFF|)$ 的相对有界增量算法；（c）在处理（边和点）批更新和通用模式图时，存在时间复杂度为 $O(|\Delta G|(|Q||AFF| + |AFF|^2))$ 的相对有界增量算法。然而对于基于子图同构的图匹配查询，即使是特殊情况当处理单元更新，并且模式图为树结构、数据图为森林结构时，其动态数据图查询问题依然不存在相对有界增量算法。

而后，基于增量相对有界性理论，[52] 证明了强连通分量查询问题和正则路径查询问题的动态数据图查询问题都存在有界增量算法。

现有技术存在的问题。现有图匹配查询的增量查询技术只考虑数据图动态变化环境下的

增量计算问题。然而，对于所有类型查询引擎和所有类型数据集，迭代设计查询是一个常见的过程，即用户根据上一轮查询结果满意程度多次反复调整查询输入，以期得到预期的查询结果。然而，本文通过对图匹配查询的各种匹配语义分析得知（章节 2.1），图匹配查询语义的计算复杂度普遍较高。即使采用复杂度最低的图匹配查询语义（图模拟语义，二次方多项式时间），每当模式图发生更新，重新执行图匹配查询计算过程这种更新处理方法计算开销太大，是绝对不可行的。另外，数据图和模式图很可能同时发生更新。针对这些问题，本文将提出统一的图匹配查询增量计算框架，该框架可以处理连续的单独或混合的模式图和数据图更新。这是对模式图更新的第一次研究，也是迄今为止所考虑的最广泛和最实用的动态设置。

2.3.3 图匹配查询的数据索引/压缩/采样技术

数据索引、数据压缩和数据采样技术是数据库和数据挖掘领域中用于提高大规模数据集上的数据查询和数据分析效率的主要技术 [40, 53]。相关研究提出分别将这三项技术应用于优化图匹配查询过程，提高图匹配查询效率。

数据索引。[54–56] 将数据索引技术扩展应用于优化图匹配查询过程，基于子图同构匹配语义。其假定数据图 G 包括数量庞大的中等规模子图，图匹配查询即为模式图 Q 在 G 中找出所有与 Q 子图同构的数据子图。图匹配查询计算过程需要为 Q 和 G 中每个数据子图计算是否基于子图同构匹配，这一过程需要极高的计算花费。图匹配查询的数据索引技术通过执行预处理过程，为数据图中的每个数据子图按路径结构 [56]、树结构 [54]、图结构 [55] 抽取出所有特征结构，再将子图中出现的所有特征结构编码为索引数据结构。当输入模式图 Q，首先为 Q 按同样方法抽取出所有特征结构。而后，通过访问数据图索引，将 Q 与 G 中每个子图对应的索引数据结构逐一比对，判断数据子图是否包括 Q 的所有特征结构。如果不包括，则该数据子图不可能与 Q 基于子图同构匹配，将其快速过滤即可；如果包括，则进一步为该数据子图执行子图同构图匹配计算判定是否与 Q 匹配。也就是说，索引结构后可以快速过滤掉 G 中大部分与 Q 显然不匹配的数据，使得只需访问小部分与 Q 相关的数据图 G' 得以计算得出准确查询结果。

数据压缩。[57] 提出将数据压缩技术扩展应用于优化图匹配查询中。对于图模拟及其扩展匹配语义 [13, 28]，[57] 定义了基于双向模拟等价类 [58] 的压缩方法，该压缩方法可以保证模式图在原始数据图上的图匹配查询结果和压缩后数据图上的图匹配查询结果完全相同。图匹配查询的数据压缩技术通过执行预处理过程，可以在 $O(|E|log|V|)$ 时间内将大

规模数据图 G 压缩为较小规模数据图 G'，并且保证（a）对于每个基于图模拟语义（或其扩展语义）的模式图 Q，$Q(G) = Q(G')$ 成立；并且（b）任意一个用于计算 $Q(G)$ 的算法可以被直接用于计算 Q 在压缩后数据图 G' 上的图匹配查询结果 $Q(G')$。也就是说，虽然数据压缩技术不能降低图匹配查询的计算复杂度，但是其通过减少数据图的数据量，并且保证不影响图匹配查询结果的准确性，从而降低图匹配查询过程的计算开销。

数据采样。[59–62] 提出用于优化大规模数据图上查询和分析计算的通用数据图采样技术。数据图采样技术的目标是通过采用基于遍历采样 [59, 60] 或随机游走 [61, 62] 等采样方法从大规模数据图 G 中抽取出具有代表性的小部分节点和边，并且这部分节点和边构成的采样数据图 G' 能够代表原始数据图 G 的关键特征，例如节点出入度分布、强弱连通分量大小分布、节点可达性及最短路径距离分布等，使得输入查询或数据分析任务在 G' 上可以高效率计算得出精度可接受的近似查询或数据分析结果。数据图采样技术是通用技术，其采样后的数据图 G' 可以用于高效率近似回答模式图 Q 的图匹配查询结果。

2.3.4　图匹配查询的查询重构技术

查询重构是数据库、数据挖掘和信息检索领域中的经典问题，其主要目的是为用户提供一组可以更好的捕捉其搜索意图的替代查询。根据重构目标的不同，查询重构主要分为查询细化 [63–65] 和查询松弛 [66–72]。

查询细化。查询细化重构技术 [63–65] 是用于解决查询过程中查询结果信息过量问题的技术。对于规模较大的数据集，用户的输入查询可能会返回大量查询结果，使得用户无法从海量查询信息中获取出理想查询结果。此时，查询引擎通过分析用户的查询意图在用户输入查询的基础上为用户生成一组查询条件细化后的查询，供用户选择。

查询松弛。查询松弛重构技术 [66–72] 是用于解决查询过程中查询结果过少问题的技术。对于规模较大并且结构复杂的数据集，由于用户无法全面掌握数据集基本信息，使得用户很可能构造出不存在任何查询结果的输入查询。此时，查询引擎通过分析用户的查询意图并结合数据集相关数据预测查询结果数量，为用户生成一组查询条件松弛后且很可能能够得到用户理想查询结果的查询，供用户选择。

[63, 65] 提出将查询细化重构技术扩展应用于优化图匹配查询过程。其根据用户输入的模式图 Q，通过在模式图的不同位置上添加节点和边生成一组模式图的超图 $\{Q_1^{\text{sup}}, Q_2^{\text{sup}}, \cdots, Q_n^{\text{sup}}\}$ 作为替代模式图供用户选择。其中，Q_i^{sup}（$i \in [1, n]$）是 Q 的超图；并且，Q_i^{sup} 在数据图 G 中的图匹配查询结果 $Q_i^{\text{sup}}(G)$ 是 Q 在数据图 G 中的图匹配查询结

果 $Q(G)$ 的子集，即 $Q_i^{\text{sup}}(G) \subseteq Q(G)$。用户利用替代模式图 Q_i^{sup} 可以在数据图中快速过滤掉不相关的信息并得到用户期望的图匹配查询结果。另外，[72] 提出将查询松弛重构技术扩展应用于优化图匹配查询过程。其根据用户输入的模式图 Q，通过从模式图中删除不同节点和边生成一组模式图的子图 $\{Q_1^{\text{sub}}, Q_2^{\text{sub}}, \cdots, Q_n^{\text{sub}}\}$ 作为替代模式图供用户选择。其中，Q_i^{sub}（$i \in [1,n]$）是 Q 的子图；并且，Q_i^{sub} 在数据图 G 中的图匹配查询结果 $Q_i^{\text{sub}}(G)$ 是 Q 在数据图 G 中的图匹配查询结果 $Q(G)$ 的超集，即 $Q(G) \subseteq Q_i^{\text{sub}}(G)$。用户利用替代模式图 Q_i^{sub} 可以在数据图中查询得到更多满足用户需求的图匹配查询结果。

现有技术存在的问题。对于 Web 网络上的关键词搜索，由于其查询约束条件较少，Web 搜索引擎通常能够从互联网上查找到千万个甚至数亿个与关键词匹配的 Web 网页。所以通常需要查询细化重构技术帮助用户快速过滤掉不相关网页并找到用户期望的查询结果。然而，对于图数据上的图匹配查询，其查询输入为模式图，查询目标为与模式图拓补结构匹配的子图。由于其查询条件较复杂，通常导致模式图在图数据中的查询结果较少甚至为空集（参见章节 5.1 实验）。所以，本文关注于解决图匹配查询的结果稀少问题。现有工作 [72] 提出利用查询松弛技术解决这一问题。其通过改变模式图结构，即减少模式图中的节点和边来查找得到更多查询结果。然而，图匹配查询的目标是查询与模式图结构匹配的子图，修改模式图结构意味着从根本改变了用户查询意图。针对这一问题，本文将提出保持原有模式图结构的图匹配查询松弛重构技术，能够有效的查找出更多满足用户需求的结果，同时能够解释通过松弛技术产生的新结果。

第三章　基于视图的近似图匹配查询

基于视图的图匹配查询方法，由于其存储视图的物理空间有限，并且由于图匹配查询的模式图表达的查询条件比较复杂，匹配约束条件非常严格，所以导致大部分模式图无法满足视图回答的命中条件，造成视图的命中率和利用率都很低。本文提出基于视图的近似图匹配查询方法。该方法通过利用视图查询方法来提高查询效率，并利用近似查询技术提高存储视图利用率来进一步达到提高查询效率目的。具体而言，给定模式图 Q 和视图集合 $\mathcal{V}$，在 $\mathcal{V}$ 无法回答 Q 的情况下，本文提出基于视图的近似图匹配查询方法，该方法根据 $\mathcal{V}$ 中信息为模式图 Q 找到一个上界近似查询 Q_u 和一个下界近似查询 Q_l，满足（a）对于任意数据图 G，Q 在 G 中的图匹配查询结果（或部分图匹配查询结果）被包含于 Q_u 在 G 中的图匹配查询结果 $Q_u(G)$ 中，并且包含 Q_l 在 G 中的图匹配查询结果 $Q_l(G)$；并且（b）Q_u 和 Q_l 在 G 中的图匹配查询结果可以通过访问视图集合 $\mathcal{V}$ 计算得出。本文提出基于视图的近似图匹配查询方法的通用计算框架，适用于任意一种图匹配查询语义。本文在论述该方法工作原理时以图模拟为图匹配查询语义，并将该方法扩展至子图同构图匹配查询语义以验证方法的通用性。本文对基于视图的近似图匹配查询方法中存在的理论问题开展了深入研究。给定模式图 Q 和视图集合 $\mathcal{V}$，本文研究的问题包括：根据视图集合 $\mathcal{V}$ 的信息，（1）如何合理定义 Q 的上界和下界近似查询；（2）如何判断是否存在 Q 的上界和下界近似查询；（3）如果存在，如何找到最优的 Q 的上界和下界近似查询。本文从计算复杂性、可近似性和算法设计三方面对上述问题进行了深入研究，并通过实验分析验证了基于视图的近似图匹配查询方法在真实数据集知识图谱 DBpedia 和视频网络图 YouTube 上能够显著提高图匹配查询的效率。

3.1　引言

视图技术在关系数据库中的研究与应用已经非常深入 [42]。[46, 47] 通过将视图技术应用于优化图匹配查询过程，提出基于视图的图匹配查询方法（章节 2.3.1）。其问题描述如下：如果模式图 Q 可以被视图集合 $\mathcal{V} = \{\mathsf{V}_1, \ldots, \mathsf{V}_n\}$ 回答，那么对于任意数据图 G，模式图 Q 在数据图 G 中的图匹配查询结果 $Q(G)$ 可以通过访问视图集合在 G 中的图匹配查询结果 $\mathsf{V}_1(G), \ldots, \mathsf{V}_n(G)$ 计算得出。其中，视图 V_i 可以看做一个已知的模式图，$\mathsf{V}_i(G)$ 为已经计算并存储的 V_i 在 G 中的图匹配查询结果。从原理上分析，基于视图的图匹配查询方法应该能够大幅度提高图匹配查询的效率。然而，由于存储视图的物理空间

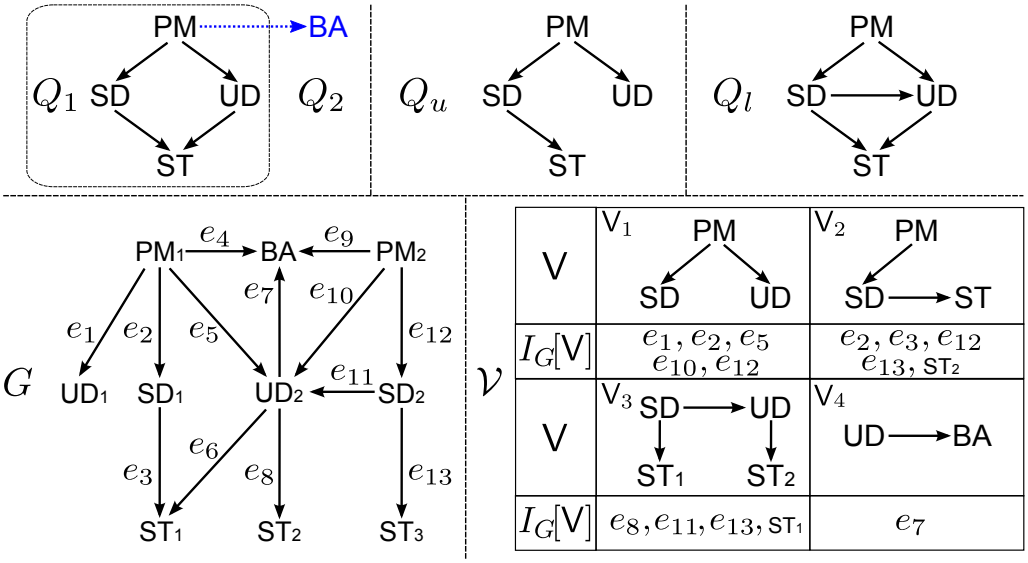


图 7　查询推荐网络图

有限，并且由于图匹配查询的模式图表达的查询条件比较复杂，匹配约束条件非常严格，所以导致大部分情况下，输入的模式图无法满足视图回答的命中条件，即模式图在数据图中的图匹配查询结果无法由存储视图计算得出，造成视图的命中率和利用率都很低。

针对上述问题，本文提出基于视图的近似图匹配查询方法，该方法可以最大化利用存储视图信息。在模式图 Q 不能被视图集合 $\mathcal{V}$ 精确回答的情况下，该方法通过利用 $\mathcal{V}$ 中信息，可以为 Q 计算得出 Q 在 G 中精确图匹配查询结果的上界近似结果和下界近似结果。具体而言，对于任意模式图 Q, 可能存在两个模式图 Q_u 和 Q_l，满足

（1）Q_u 和 Q_l 在 G 中的图匹配查询结果可以由视图集合 $\mathcal{V}$ 计算得出；

（2）Q 被包含于 Q_u 并且 Q 包含 Q_l，即对于任意数据图 G，$Q_l(G) \subseteq Q(G)$ 并且 $Q(G) \subseteq Q_u(G)$ 成立。其中，$\subseteq$ 表示图匹配查询结果之间的包含关系。

也就是说，对于任意数据图 G，模式图 Q、Q_u 和 Q_l 在 G 中的图匹配查询结果分别为 $Q(G)$、$Q_u(G)$ 和 $Q_l(G)$。其中，$Q_u(G)$ 是 $Q(G)$ 的上限，$Q_l(G)$ 是 $Q(G)$ 的下限。本文称 Q_u 为 Q 的上界近似查询，Q_l 为 Q 的下界近似查询。Q_u 和 Q_l 一起为 Q 提供了一对基于视图信息的上界和下界近似。

例 3.1: 以一个推荐网络图 G 为例 [20]，如图 7 所示。G 中每个节点表示一个具有专业知识的技术人员，节点上的标签表示不同的专业，比如项目经理（PM）、软件开发工程师（SD）、软件测试工程师（ST）、界面设计工程师（UD）以及业务分析师（BA）；G 中每条边表示两个技术人员之间存在过良好的合作关系，比如边 (PM_1, SD_1) 表示 SD_1 曾

经在 PM_1 的领导下高效的合作并完成过项目开发。一个人力资源部经理想要为他所在的软件公司组建一个开发团队致力于即将开展的企业软件开发项目。为了组建一个能够胜任软件开发业务需求并且可以良好合作的开发团队，他将团队组建需求用模式图 Q_1 来表示，并借助于推荐网络图 G 来查询出满足需求的团队，Q_1 如图 7 所示（虚线方格内）。他期望找到的目标团队需要包括 PM、SD、UD 和 ST，并且（1）PM 能够监督 SD 和 UD 工作，即 PM 与 SD 和 UD 都有高效的合作关系，（2）SD 和 UD 分别能够与 ST 高效合作。而后，查询引擎可以为 Q_1 在 G 中执行图匹配查询过程，从而在 G 中找到与 Q_1 中节点标签及结构均匹配的子图作为目标团队。以图模拟匹配语义为例，验证可知，G 中存在 Q_1 的基于图模拟的图匹配查询结果。

假设查询引擎中已经预先存储了视图集合 $\mathcal{V} = \{V_1, V_2, V_3, V_4\}$ 和他们在 G 中基于图模拟的图匹配查询结果，如图 7 所示。分析可知以下结论：（1）仅访问视图集合 $\mathcal{V}$ 和他们在 G 中的图匹配查询结果无法计算得出 Q_1 在 G 中的图匹配查询结果，因为 $\mathcal{V}$ 中不存在一个视图子集可以通过合并计算得出 Q_1 的查询结果。（2）然而，存在两个模式图 Q_u 和 Q_l，其在 G 中的图匹配查询结果可以由视图在 G 中的图匹配查询结果合并计算得出。其中，Q_u 在 G 中的图匹配查询结果可以由 $I_G[V_1]$ 和 $I_G[V_2]$ 合并计算得出，得到查询结果为 $Q_u(G)=\{PM_i, UD_j, SD_k, ST_h\}$ $(i, j, k \in [1,2], h \in [1,3])$；$Q_l$ 在 G 中的图匹配查询结果可以由 $I_G[V_1]$ 和 $I_G[V_3]$ 计算得出，得到查询结果为 $Q_l(G) = \{PM_2, UD_2, SD_2, ST_h\}$ $(h \in [1,3])$。（3）Q_1 在 G 中基于图模拟的图匹配查询结果为 $Q_1(G)=\{PM_i, UD_2, SD_k, ST_h\}$ $(i,k \in [1,2], h \in [1,3])$. 因此，$Q_l(G) \subseteq Q_1(G) \subseteq Q_u(G)$ 成立。以上说明，尽管 $\mathcal{V}$ 不能精确的回答 Q_1，然而 $\mathcal{V}$ 能够回答出 Q_1 的上界近似结果和下界近似结果。

再考虑另一个模式图 Q_2，如图 7 所示。人力资源部经理想要再为该团队找到一个业务分析师 BA，于是他在 Q_1 的基础上增加一个标签为 BA 的节点和一条由 PM 指向 BA 的边，在图 7 所示 Q_2 中以蓝色标识。验证可知，Q_2 不能被 $\mathcal{V}$ 精确回答。甚至，根据 $\mathcal{V}$ 中信息，无法找到 Q_2 的上界近似查询和下界近似查询。也就是说，无法为 Q_2 找到类似 Q_1 的上界近似查询 Q_u 和下界近似查询 Q_l，可以为 Q_2 在 G 中的图匹配查询结果 $Q_2(G)$ 界定上限和下限。然而，因为 Q_1 是 Q_2 的子图，Q_u 和 Q_l 作为 Q_2 的子图的上界和下界近似查询，显然可以为 Q_2 提供上界和下界近似。 □

为了充分利用上述想法解决图匹配查询的视图技术中存在的瓶颈问题，本文将对以下三个核心问题展开深入研究：（1）如何合理定义 Q 的基于视图的上界和下界近似查询，使其既能够提供“完全”上界和下界近似（类似例 3.1 中 Q_1、Q_u 和 Q_l），也能够提

供“部分”上界和下界近似（类似例 3.1 中 Q_2、Q_u 和 Q_l）；（2）根据视图 $\mathcal{V}$ 的信息，如何判断是否存在 Q 的上界近似查询 Q_u 和下界近似查询 Q_l；（3）如果存在，如何找到 Q 的最优的上界近似查询 Q_u 和下界近似查询 Q_l，使得 Q_u 和 Q_l 的拓补结构与 Q 的拓补结构最相近。

本章主要内容提纲。基于对以上三个核心问题的研究，本章节研究内容提纲如下所示。

（1） 本文提出了基于视图的上界和下界近似图匹配查询方法，定义了模式图 Q 的（完全）上界近似查询和（完全）下界近似查询（章节 3.2）。

（2） 本文对基于视图的近似图匹配查询的基本问题进行了理论分析，分析了图匹配查询的上界近似查询和下界近似查询问题的计算复杂性和可近似性（章节 3.3）。

（3） 本文提出了精确算法以及具有近似度保证的近似算法，可以根据视图信息计算得出与 Q 拓补结构最相近的上界近似查询（章节 3.4）和下界近似查询（章节 3.5）。

（4） 本文提出基于视图的近似图匹配查询方法的通用计算框架，适用于任意一种图匹配查询语义。本文将该方法扩展至子图同构图匹配查询语义以验证方法的通用性（章节 3.6）。

（5） 本文采用知识图谱 DBpedia 和视频网络图 YouTube 两个真实数据集来评估方法性能，数据集包含百万级节点数据和千万级边数据。经实验验证得知，基于视图的近似图匹配查询方法对图匹配查询过程的效率提升可以达到 2-3 个数量级，验证了方法的有效性（章节 3.7）。

为了论述简洁，本文只将部分定理证明列于正文中，其他定理证明详见附录 A。

3.2　基于视图的近似查询

本文在章节 2.1 和章节 2.3.1 分别对模式图、数据图、图匹配查询和基于视图的图匹配查询方法进行了详细介绍。本章首先对这些基本概念进行回顾，而后给出基于视图的上界和下界近似查询方法和其中基本概念的定义。

本文在介绍基于视图的近似图匹配查询方法时首先以图模拟为图匹配查询语义。

定义 3.1 (图模拟 [13]): 给定模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$，Q 与 G 通过图模拟匹配，记为 $Q \prec G$，当且仅当存在一个节点集合之间的二元关系 $R \subseteq V_Q \times V_G$，满足（1）对于每对节点 $(u, v) \in R$, 节点 u 和节点 v 有相同的标签，即 $l_Q(u) = l_G(v)$；并且（2）对于 Q 中每个节点 u，在 G 中都存在一个节点 v，满足（a）$(u, v) \in R$，并且（b）对于 Q 中每条边 (u, u')，在 G 中都存在一条边 (v, v')，满足 $(u', v') \in R$。

其中，G 中会存在多个与 Q 匹配的二元关系，但只存在一个最大二元匹配关系 R_M，使得对于 G 中所有与 Q 匹配的二元关系 P，P 都是 R_M 的子集，即 $P \subseteq R_M$。 □

定义 3.2 (图匹配结果和镜像): 给定模式图 Q 和数据图 G，假设 Q 与 G 基于图模拟匹配，即 $Q \prec G$，本文定义 Q 在 G 中的图匹配结果 $Q(G)$ 为 G 的一个节点子集，即 $Q(G) \subseteq V_G$。其中，节点 $v \in Q(G)$ 当且仅当节点 v 属于最大二元匹配关系 R_M。本文定义 Q 在 G 中的镜像 $I_G[Q]$ 为 G 的一个子图 $G_s[V_s, E_s]$。其中，（1）$V_s = Q(G)$，（2）边 $(v, v') \in E_s$ 当且仅当 Q 中存在一条边 (u, u')，满足 $(u, v) \in R_M$，$(u', v') \in R_M$。直观理解，镜像 $I_G[Q]$ 是 G 的子图，其节点集为 $Q(G)$，边集包括 G 中两个端点均属于 $Q(G)$ 的所有边。 □

本文定义模式图 Q_1 与 Q_2 等价，当且仅当对于任意数据图 G，$Q_1(G) = Q_2(G)$。

定义 3.3 (视图): 视图查询（或简称视图）V 是一个模式图，并且其在数据图中的镜像已经被存储。视图集合 $\mathcal{V}$ 是由多个视图组成的集合。对于视图集合 $\mathcal{V}$，本文用 $\|\mathcal{V}\|$ 表示 $\mathcal{V}$ 中视图的个数，用 $|\mathcal{V}|$ 表示 $\mathcal{V}$ 中所有视图的大小之和。 □

定义 3.4 (基于视图的图匹配查询 [46, 47]): 给定视图集合 $\mathcal{V} = \{\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_n\}$，对于模式图 Q, 如果存在另外一个模式图 A，使得对于任意数据图 G，（1）A 与 Q 等价，即 A 在 G 中的图匹配结果 $A(G)$ 与 Q 在 G 中的图匹配结果 $Q(G)$ 相同；并且，（2）计算 $A(G)$ 只需要访问 $\mathcal{V}$ 中视图以及他们在 G 中的镜像 $I_G[\mathcal{V}] = \{I_G[\mathsf{V}_1], \ldots, \mathsf{V}_n(G)\}$，则称模式图 Q 可以被视图集合 $\mathcal{V}$ 回答。 □

例 3.2: 以例 3.1 中视图集合 $\mathcal{V}$ 为例，如图 7 所示。视图集合 $\mathcal{V}$ 中每个视图 V_i 在数据图 G 中的镜像 $I_G[\mathsf{V}_i]$ $(i \in [1, 4])$ 如图 7 所示，图 7 中镜像结构以边和节点表示。分析可知，Q_u 和 Q_l 均可以被视图集合 $\mathcal{V}$ 回答。具体而言，Q_u 在 G 中的图匹配结果可以通过将镜像 $I_G[\mathsf{V}_1]$ 和 $I_G[\mathsf{V}_2]$ 合并计算得出；Q_l 在 G 中的图匹配结果可以通过将镜像 $I_G[\mathsf{V}_1]$ 和 $I_G[\mathsf{V}_3]$ 合并计算得出。 □

定义 3.5 (（完全）上界/下界包含): 对于模式图 Q 和 Q_u，如果存在 Q 的一个诱导子图 Q_s，满足对于任意数据图 G，$Q_s(G) \subseteq Q_u(G)$ 成立，则称 Q 被上界包含于 Q_u，记为 $Q \sqsubseteq_{\mathcal{U}} Q_u$；类似的，对于模式图 Q 和 Q_l，如果存在 Q 的一个诱导子图 Q_s，满足对于任意数据图 G，$Q_l(G) \subseteq Q_s(G)$ 成立，则称 Q 下界包含 Q_l，记为 $Q_l \sqsubseteq_{\mathcal{L}} Q$。存在一个特殊情况，当上述 Q_s 为 Q 时，则称 Q 被完全上界包含于 Q_u，记为 $Q \sqsubseteq^c_{\mathcal{U}} Q_u$；类似的，称 Q 完全下界包含 Q_l，记为 $Q_l \sqsubseteq^c_{\mathcal{L}} Q$。

其中，如果图 $Q_s[V_s, E_s]$ 是图 $Q(V_Q, E_Q)$ 的一个子图，则 $V_s \subseteq V_Q$，$E_s \subseteq E_Q$ 成立。如果 Q_s 的边集 E_s 包括 Q 中两个端点都属于 V_s 的所有边，则称 Q_s 为 Q 的诱导子图。□

例 3.3: 以例 3.1 中模式图 Q_1、Q_2、Q_u、Q_l 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。分析可知以下结论，$Q_1 \sqsubseteq^c_{\mathcal{U}} Q_u$，$Q_2 \sqsubseteq_{\mathcal{U}} Q_u$，$Q_l \sqsubseteq^c_{\mathcal{L}} Q_1$ 和 $Q_l \sqsubseteq_{\mathcal{L}} Q_2$ 成立。

另外，验证可知 Q_1 是 Q_2 的诱导子图。进一步结合图 7 中数据图 G 的信息，分析可知，Q_1 在 G 中基于图模拟的图匹配结果为 $Q_1(G) = \{\mathsf{PM_i}, \mathsf{UD_2}, \mathsf{SD_k}, \mathsf{ST_h}\}$ $(i, k \in [1,2], h \in [1,3])$。$Q_u$ 在 G 中基于图模拟的图匹配结果为 $Q_u(G) = \{\mathsf{PM_i}, \mathsf{UD_j}, \mathsf{SD_k}, \mathsf{ST_h}\}$ $(i, j, k \in [1,2], h \in [1,3])$；$Q_l$ 在 G 中基于图模拟的图匹配结果为 $Q_l(G) = \{\mathsf{PM_2}, \mathsf{UD_2}, \mathsf{SD_2}, \mathsf{ST_h}\}$ $(h \in [1,3])$。所以, $Q_l(G) \subseteq Q_1(G) \subseteq Q_u(G)$ 成立。□

定义 3.6 (基于视图的（完全）上界/下界近似查询): 如果（1）$Q \sqsubseteq_{\mathcal{U}} Q_u$ 成立，即 Q 被上界包含于 Q_u，并且（2）Q_u 可以被视图集合 $\mathcal{V}$ 回答，则称 Q_u 为模式图 Q 的一个基于 $\mathcal{V}$ 的上界近似查询；类似的，如果（1）$Q_l \sqsubseteq_{\mathcal{U}} Q$ 成立，即 Q 下界包含 Q_l，并且（2）Q_l 可以被视图集合 $\mathcal{V}$ 回答，则称 Q_l 为模式图 Q 的一个基于 $\mathcal{V}$ 的下界近似查询。

存在一个特殊情况，当上述条件（1）中 $Q \sqsubseteq^c_{\mathcal{U}} Q_u$ 成立时，即 Q 被完全上界包含于 Q_u，则称 Q_u 为模式图 Q 的一个基于 $\mathcal{V}$ 的完全上界近似查询；类似的，当上述条件（1）中 $Q_l \sqsubseteq^c_{\mathcal{L}} Q$ 成立时，即 Q 完全下界包含 Q_l，则称 Q_l 为模式图 Q 的一个基于 $\mathcal{V}$ 的完全下界近似查询。□

例 3.4: 由例 3.3 可知 $Q_2 \sqsubseteq_{\mathcal{U}} Q_u$ 和 $Q_l \sqsubseteq_{\mathcal{L}} Q_2$ 成立。此外，由例 3.2 可知 Q_u 和 Q_l 均可以被视图集合 $\mathcal{V}$ 回答。所以，Q_u 是模式图 Q_2 的一个基于 $\mathcal{V}$ 的上界近似查询，Q_l 是模式图 Q_2 的一个基于 $\mathcal{V}$ 的下界近似查询。

另外，由于 $Q_1 \sqsubseteq^c_{\mathcal{U}} Q_u$ 和 $Q_l \sqsubseteq^c_{\mathcal{L}} Q_1$ 成立，并且 Q_u 和 Q_l 均可以被视图集合 $\mathcal{V}$ 回答。所以，Q_u 是模式图 Q_1 的一个基于 $\mathcal{V}$ 的完全上界近似查询，Q_l 是模式图 Q_1 的一个基于 $\mathcal{V}$ 的完全下界近似查询。□

3.3　理论分析

本章对基于视图的近似图匹配查询方法中的核心问题进行理论分析。首先，章节 3.3.1 对方法的基本性质进行刻画描述。而后，章节 3.3.2 对方法核心问题的计算复杂性和可近似性展开分析。

3.3.1 基本性质

为了研究基于视图的上界和下界近似图匹配查询，需要首先对（1）上界和下界包含关系和（2）基于视图的图匹配查询问题的基本性质进行刻画描述。

（完全）上界/下界包含。上界包含，下界包含，完全上界包含和完全下界包含四种包含关系的性质刻画如下。

定理 3.1: 对于模式图 Q，Q_u 和 Q_l，

(1) $Q \sqsubseteq_{\mathcal{U}} Q_u$ 当且仅当 $Q_u \prec Q$；

(2) $Q \sqsubseteq^c_{\mathcal{U}} Q_u$ 当且仅当 $Q_u \prec Q$ 并且 $V_Q = V_{I_Q[Q_u]}$；

(3) $Q_l \sqsubseteq_{\mathcal{L}} Q$ 当且仅当存在 Q 的诱导子图 Q_s 满足 $Q_s \prec Q_l$ 并且 $V_{Q_l} = V_{I_{Q_l}[Q_s]}$；

(4) $Q_l \sqsubseteq^c_{\mathcal{L}} Q$ 当且仅当 $Q \prec Q_l$ 并且 $V_{Q_l} = V_{I_{Q_l}[Q]}$；

其中，$I_{Q'}[Q'']$ 是 Q'' 在 Q' 中的镜像。 □

直观理解，四种包含关系的性质将测试 Q 与 Q_u 或 Q_l 之间是否满足上界或下界包含关系转换为测试 Q 与 Q_u 或 Q_l 之间是否满足图匹配关系（图模拟）。

基于视图的图匹配查询。本文在章节 2.3.1 给出了如何判定一个模式图是否可以被给定视图集合回答的充分必要条件。为了便于阅读，再次引用该充要条件如下。

引理 3.1 ([46]): 对于视图集合 $\mathcal{V}$ 和模式图 Q，模式图 Q 可以被视图集合 $\mathcal{V}$ 回答当且仅当 $E_Q = \bigcup_{V \in \mathcal{V}} E_{I_Q[V]}$ 成立。 □

例 3.5: 以 Q_l 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。分析可知，Q_l 可以被 $\mathcal{V}$ 回答，因为充要条件 $\bigcup_{V_i \in \mathcal{V}\,(i \in [1,4])} E_{I_{Q_l}[V_i]} = \{(\mathrm{PM}, \mathrm{SD}), (\mathrm{PM}, \mathrm{UD})\} \cup \{(\mathrm{PM}, \mathrm{SD}), (\mathrm{SD}, \mathrm{ST})\} \cup \{(\mathrm{SD}, \mathrm{UD}), (\mathrm{SD}, \mathrm{ST}), (\mathrm{UD}, \mathrm{ST})\} \cup \emptyset = E_{Q_l}$ 成立。 □

基于引理 3.1，[46] 设计了图匹配查询视图回答算法 QAV。该算法可以判断模式图 Q 是否可以被视图集合 $\mathcal{V}$ 回答，并根据 $\mathcal{V}$ 计算得出 Q 在 G 中的图匹配查询结果。本文在章节 2.3.1已经对算法 QAV 进行了详细介绍。

3.3.2 计算复杂性和可近似性分析

本章将对基于视图的上界近似查询和下界近似查询，以及其特殊情况完全上界近似查询和完全下界近似查询相关核心问题的计算复杂性和可近似性展开分析。

近似查询的存在性。基于视图的（完全）上界和下界近似查询并不总是存在，如例 3.1 所示。本章首先对（完全）近似查询的存在性进行分析。

上界近似查询的存在性。上界近似查询的存在性问题，记为 EUA，指给定视图集合 $\mathcal{V}$ 和模式图 Q，判断是否存在一个 Q 的基于 $\mathcal{V}$ 的上界近似查询 Q_u；完全上界近似查询的存在性问题，记为 EUA^c，指给定视图集合 $\mathcal{V}$ 和模式图 Q，判断是否存在一个 Q 的基于 $\mathcal{V}$ 的完全上界近似查询 Q_u^c。

定理 3.2: 给定视图集合 $\mathcal{V}$ 和模式图 Q，

(a) 存在一个 Q 的基于 $\mathcal{V}$ 的上界近似查询当且仅当存在视图 $\mathsf{V} \in \mathcal{V}$ 使得图匹配结果 $\mathsf{V}(Q) \neq \emptyset$；

(b) 存在一个 Q 的基于 $\mathcal{V}$ 的完全上界近似查询当且仅当 $V_Q = \bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]}$；

(c) EUA 和 EUA^c 的时间复杂度都是 $|Q|$ 和 $|\mathcal{V}|$ 的二次方多项式时间。 □

本文将在章节 3.4 通过分别设计 EUA 问题和 EUA^c 问题的二次方多项式时间算法证明定理 3.2（c）。定理 3.2（a,b）的证明见附录 A。

例 3.6: 以模式图 Q_1 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。根据定理 3.2（a）可知，因为存在一个视图 $\mathsf{V}_1 \in \mathcal{V}$ 满足 $\mathsf{V}_1(Q_1) \neq \emptyset$，所以存在 Q_1 的基于 $\mathcal{V}$ 的上界近似查询。此外，根据定理 3.2（b）可知，因为 $\bigcup_{\mathsf{V}_i\in\mathcal{V}\,(i\in[1,4])} V_{I_{Q_1}[\mathsf{V}_i]} = \{\mathsf{PM}, \mathsf{SD}, \mathsf{UD}, \mathsf{ST}\} = V_{Q_1}$，所以存在 Q_1 的基于 $\mathcal{V}$ 的完全上界近似查询。 □

下界近似查询的存在性。与上界近似查询的存在性问题类似，下界近似查询的存在性问题，记为 ELA，指给定视图集合 $\mathcal{V}$ 和模式图 Q，判断是否存在一个 Q 的基于 $\mathcal{V}$ 的下界近似查询 Q_l；完全下界近似查询的存在性问题，记为 ELA^c，指给定视图集合 $\mathcal{V}$ 和模式图 Q，判断是否存在一个 Q 的基于 $\mathcal{V}$ 的完全下界近似查询 Q_l^c。

定理 3.3: 给定视图集合 $\mathcal{V}$ 和模式图 Q，

(a) 存在一个 Q 的基于 $\mathcal{V}$ 的完全下界近似查询 Q_l^c 当且仅当 $E_Q \subseteq \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{\widehat{Q}}[\mathsf{V}]}$；

(b) ELA^c 的时间复杂度为 $O(|\mathcal{V}||Q|^2)$；

(c) ELA 是 NP-完全问题。

其中，$\widehat{Q}$ 是具有与 Q 相同节点的完全图。 □

本文将在章节 3.5 通过设计 ELA^c 问题的多项式时间算法证明定理 3.3（b）。定理 3.3

（a,c）的证明见附录 A。

与上界近似查询的存在性问题不同，下界近似查询的存在性问题的计算复杂度比完全下界近似查询的存在性问题的计算复杂度高很多。

例 3.7: 以模式图 Q_1 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。根据定理 3.3（a）可知，存在 Q_1 的基于 $\mathcal{V}$ 的完全下界近似查询，因为通过为视图集合中每个视图 $\mathsf{V}_i \in \mathcal{V}$ $(i \in [1,4])$ 计算其在 Q_1 的完全图 $\widehat{Q_1}$ 中基于图模拟的图匹配结果，可以得到 $\bigcup_{\mathsf{V}_i \in \mathcal{V}\,(i \in [1,4])} E_{I_{\widehat{Q_1}}[\mathsf{V}_i]} = \{(\mathsf{PM},\mathsf{SD}), (\mathsf{PM},\mathsf{UD}), (\mathsf{SD},\mathsf{UD}), (\mathsf{SD},\mathsf{ST}), (\mathsf{UD},\mathsf{ST})\} \supseteq E_{Q_1}$。 □

最优近似查询。给定视图集合 $\mathcal{V}$ 和模式图 Q，可能会存在多个 Q 的基于 $\mathcal{V}$ 的近似查询。显然，我们需要找到与 Q 拓补结构最相近的近似查询，即最优近似查询。下面首先定义用于衡量近似查询质量的结构距离函数。而后，基于结构距离函数定义最优上界近似查询和最优下界近似查询。

定义 3.7 (结构距离): 对于模式图 Q_1 和模式图 Q_2，本文定义 Q_1 和 Q_2 的结构距离 $\mathsf{clo}(Q_1, Q_2)$，为 Q_1 和 Q_2 中不属于 Q_1 和 Q_2 的最大（边诱导）公共子图的边的个数之和。直观理解，如果 $\mathsf{clo}(Q_1, Q_2)$ 越小，则 Q_1 和 Q_2 的拓补结构越相近。 □

最优上界近似查询。最优上界近似查询问题，记为 CUA，指给定视图集合 $\mathcal{V}$ 和模式图 Q，找到与 Q 拓补结构最相近的上界近似查询 Q_u，使得对于任意一个上界近似查询 Q'_u，$\mathsf{clo}(Q_u, Q) \leq \mathsf{clo}(Q'_u, Q)$。本文称 Q_u 为 Q 的基于 $\mathcal{V}$ 的最优上界近似查询。

类似的，最优完全上界近似查询问题，记为 CUA^c，指给定视图集合 $\mathcal{V}$ 和模式图 Q，找到与 Q 拓补结构最相近的完全上界近似查询 Q_u^c，使得对于任意一个完全上界近似查询 $Q_u'^c$，$\mathsf{clo}(Q_u^c, Q) \leq \mathsf{clo}(Q_u'^c, Q)$。本文称 Q_u^c 为 Q 的基于 $\mathcal{V}$ 的最优完全上界近似查询。

虽然计算两个图的结构距离 clo 的问题是 NP-难问题，因为该问题包含了计算两个图的最大公共子图问题，而该问题是 NP-难问题。但是，最优上界近似查询问题存在高效的多项式时间精确算法。

定理 3.4: CUA 和 CUA^c 的判定问题时间复杂度都是 $|Q|$ 和 $|\mathcal{V}|$ 的二次方多项式时间。 □

本文将在章节 3.4 通过分别设计 CUA 问题和 CUA^c 的问题二次方多项式时间算法证明定理 3.4。

最优下界近似查询。与最优上界近似查询问题类似，最优下界近似查询问题，记为 CLA，指给定视图集合 $\mathcal{V}$ 和模式图 Q，找到与 Q 拓补结构最相近的下界近似查询 Q_l。

类似的，最优完全下界近似查询问题，记为 CLA^c，指给定视图集合 $\mathcal{V}$ 和模式图 Q，找到与 Q 拓补结构最相近的完全下界近似查询 Q_l^c。

与最优上界近似查询问题存在高效的多项式时间算法不同，最优下界近似查询问题的计算复杂度比前者高很多。本文用 DCLA 表示 CLA 问题的判定问题，用 DCLA^c 表示 CLA^c 问题的判定问题。给定视图集合 $\mathcal{V}$，模式图 Q 和自然数 k，DCLA 即判定是否存在 Q 的最优下界近似查询 Q_l 满足 $\text{clo}(Q_l, Q) \le k$；DCLA^c 即判定是否存在 Q 的最优完全下界近似查询 Q_l^c 满足 $\text{clo}(Q_l^c, Q) \le k$。本文用 OCLA 表示 CLA 问题的优化问题，用 OCLA^c 表示 CLA^c 问题的优化问题。给定视图集合 $\mathcal{V}$ 和模式图 Q，OCLA 即在所有 Q 的基于 $\mathcal{V}$ 的下界查近似询 Q_l 中查找结构距离 $\text{clo}(Q_l, Q)$ 最小的下界近似查询；OCLA^c 即在所有 Q 的基于 $\mathcal{V}$ 的完全下界近似查询 Q_l^c 中查找结构距离 $\text{clo}(Q_l^c, Q)$ 最小的完全下界近似查询。基于以上判定问题和优化问题的定义，可以得到以下结论。

定理 3.5: （1）DCLA 和 DCLA^c 都是 NP-完全问题。

（2）OCLA 和 OCLA^c 均不属于 APX 类可近似问题（APX 类包含具有常数近似度保证的多项式时间近似算法的问题 [73]）。 □

定理 3.5（1）说明在当前计算复杂性理论体系下，CLA 问题和 CLA^c 问题均不存在多项式时间精确算法。定理 3.5（2）说明 CLA 问题和 CLA^c 问题均不存在常数近似度保证的近似算法。尽管问题的计算复杂度较高难以设计高效的精确算法，但是本文在仔细分析 CLA 和 CLA^c 问题特征的基础上设计了实用的近似算法和启发算法。

3.4　基于视图的上界近似查询算法

本章将分别设计用于计算最优上界近似查询和最优完全上界近似查询的多项式时间精确算法。这些算法是基于上界近似查询的小型模型性质设计的。给定视图集合 $\mathcal{V}$ 和模式图 Q，基于视图的上界近似查询的小型模型性质如下。

引理 3.2 (小型模型性质): 给定视图集合 $\mathcal{V}$ 和模式图 Q，对于任意 Q 的基于 $\mathcal{V}$ 的上界近似查询 Q_u（或完全上界近似查询 Q_u^c），存在 Q 的一个与 Q_u（或 Q_u^c）等价的子图 Q_s (或 Q_s^c 并且 $V_{Q_s^c} = V_Q$)，满足：

- Q_s（或 Q_s^c）也是一个 Q 的基于 $\mathcal{V}$ 的上界近似查询（或完全上界近似查询）；并且
- Q_s（或 Q_s^c）与 Q 比 Q_u（或 Q_u^c）与 Q 的结构更相近，即 $\text{clo}(Q_s, Q) \le \text{clo}(Q_u, Q)$（或 $\text{clo}(Q_s^c, Q) \le \text{clo}(Q_u^c, Q)$）。 □

表 4　最优完全上界近似查询算法 CUAsim^c

Algorithm：**最优完全上界近似查询算法 CUAsim^c**

Input: 视图集合 $\mathcal{V}$ 和模式图 Q。
Output: Q 的基于 $\mathcal{V}$ 的最优完全上界近似查询（如果存在）。

1. **for each** V in $\mathcal{V}$ **do** compute the image I_Q[V] of V in Q;
2. $V_u := \bigcup_{V \in \mathcal{V}} V_{I_Q[V]}$; $E_u := \bigcup_{V \in \mathcal{V}} E_{I_Q[V]}$; Let Q_u be (V_u, E_u);
3. **if** $V_u = V_Q$ **then return** Q_u;
4. **return** "no";　/* 不存在 Q 的基于 $\mathcal{V}$ 的完全上界近似查询 */

引理 3.2 保证了以下性质。对于 Q 的任意基于 $\mathcal{V}$ 的上界近似查询 Q_u，如果 Q_u 不是 Q 的子图，那么一定存在另外一个 Q 的基于 $\mathcal{V}$ 的上界近似查询 Q_s，满足：（i）Q_u 与 Q_s 等价，即对于任意数据图 G，$Q_u(G) = Q_s(G)$；并且（ii）Q_s 与 Q 的结构距离不大于 Q_u 与 Q 的结构距离。所以，只需要从 Q 的子图中即可查找到 Q 的上界近似查询，从而避免了从无穷多种可能中计算 Q 的上界近似查询。

基于引理 3.2，本文分别提出了最优上界近似查询问题 CUA 和最优完全上界近似查询问题 CUA^c 的多项式时间精确算法，从而证明定理 3.4。下面首先介绍 CUA^c 问题算法。

3.4.1　最优完全上界近似查询算法

最优完全上界近似查询问题 CUA^c 的最优完全上界近似查询算法如表 4 所示，记为 CUAsim^c。算法 CUAsim^c 的输入为视图集合 $\mathcal{V}$ 和模式图 Q。如果存在 Q 的基于 $\mathcal{V}$ 的完全上界近似查询，则 CUAsim^c 输出最优的完全上界近似查询。

具体而言，算法 CUAsim^c 首先为 $\mathcal{V}$ 中每个视图 V 计算其在 Q 中的镜像 I_Q[V]（第 1 行）。而后，CUAsim^c 根据镜像中的节点和边构建 Q 的子图 Q_u（第 2 行）。最后，CUAsim^c 验证以下条件是否成立：如果 Q_u 包含了 Q 中所有节点，则算法返回 Q_u 作为 Q 的基于 $\mathcal{V}$ 的最优完全上界近似查询（第 3 行）；否则，算法返回 "No"，因为不存在任何 Q 的基于 $\mathcal{V}$ 的完全上界近似查询（第 4 行）。

例 3.8: 以模式图 Q_1 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。算法 CUAsim^c 首先为 $\mathcal{V}$ 中每个视图 V_i ($i \in [1,4]$) 计算其在 Q_1 中的镜像 $I_{Q_1}[V_i]$（第 1 行）。而后，CUAsim^c 构建 Q_1 的子图 Q_u。Q_u 的节点集为 $V_u = \{PM, SD, UD, ST\}$, Q_u 的边集为 $E_u = \{(PM, SD), (PM, UD),$

(SD, ST) }（第 2 行）。算法最终经过验证得知 $V_u = V_{Q_1}$。所以，CUAsim^c 返回 Q_u 作为 Q_1 的最优完全上界近似查询（第 3 行），如图 7 所示。 □

算法正确性和复杂性分析。算法 CUAsim^c 的时间复杂度为 $O((|V_Q|+|E_Q|)|\mathcal{V}|+|V_Q|^2)$。其中，$|\mathcal{V}| = \sum_{V\in\mathcal{V}}(|V_V| + |E_V|)$。算法 CUAsim^c 的第一行镜像计算步骤时间花费为 $O((|V_Q| + |E_Q|)|\mathcal{V}|)$，第三行验证步骤的时间花费最多为 $O(|V_Q|^2)$。

算法 CUAsim^c 的正确性由以下性质保证：（1）根据引理 3.2，如果存在 Q 的基于 $\mathcal{V}$ 的完全上界近似查询，则其最优完全上界近似查询必然是 Q 的子图。（2）对于 Q 中任意可以被 $\mathcal{V}$ 回答的子图 Q_s，Q_s 必然只包含由算法 CUAsim^c 计算得出的 Q_u 中的节点和边。（3）因此，Q_u 是 Q 的基于 $\mathcal{V}$ 的最优上界近似查询。（4）算法 CUAsim^c 返回 Q_u 当且仅当 Q_u 是 Q 的基于 $\mathcal{V}$ 的完全上界近似查询。

3.4.2　最优上界近似查询算法

最优上界近似查询问题 CUA 的最优上界近似查询算法如表 5 所示，记为CUAsim 。算法CUAsim 以算法 CUAsim^c 为基础，并从以下几个方面扩展了 CUAsim^c。首先，CUAsim 按与 CUAsim^c 同样的方法计算视图集合 $\mathcal{V}$ 中所有视图在模式图中的镜像（第 1 行）。而后，CUAsim 不再执行验证 Q_u 是否包含 Q 中所有节点的步骤，而是判断 Q_u 是否为空集（第 2 行）。如果 Q_u 非空，则CUAsim 将 Q_u 作为最优上界近似查询返回（第 3 行）；否则，算法返回“No”，因为不存在任何 Q 的基于 $\mathcal{V}$ 的上界近似查询（第 4 行）。

例 3.9: 以模式图 Q_2 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。算法CUAsim 按与例 3.8 中同样方法构建 Q_2 的子图 Q_u，并将 Q_u 返回作为 Q_2 的最优上界近似查询，如图 7 所示。 □

算法正确性和复杂性分析。算法CUAsim 的正确性分析与算法 CUAsim^c 的正确性分析类似。算法CUAsim 判定是否存在 Q 的基于 $\mathcal{V}$ 的上界近似查询。如果存在，则返回 Q 的基于 $\mathcal{V}$ 的最优上界近似查询。算法CUAsim 的时间复杂度为 $O((|V_Q| + |E_Q|)|\mathcal{V}|)$。

3.5　基于视图的下界近似查询算法

由定理 3.5 可知，基于视图的最优下界近似查询问题 CLA 和基于视图的最优完全下界近似查询问题 CLA^c 都是 NP-完全问题。所以在当前计算复杂性理论体系下，CLA 和 CLA^c 均不存在多项式时间精确算法。尽管问题的计算复杂度较高难以设计高效的精确算法，但是本文在仔细分析 CLA 和 CLA^c 问题特征的基础上设计了具有近似度保证的近

表 5　最优上界近似查询算法 CUAsim

Algorithm：最优上界近似查询算法 CUAsim

Input: 视图集合 $\mathcal{V}$ 和模式图 Q。
Output: Q 的基于 $\mathcal{V}$ 的最优上界近似查询（如果存在）。

1. **for each** V in $\mathcal{V}$ **do** compute the image I_Q[V] of V in Q;
2. $V_u := \bigcup_{V \in \mathcal{V}} V_{I_Q[V]}$; $E_u := \bigcup_{V \in \mathcal{V}} E_{I_Q[V]}$; Let Q_u be (V_u, E_u);
3. **if** $V_u \neq \emptyset$ **then return** Q_u;
4. **return** "no";　/* 不存在 Q 的基于 $\mathcal{V}$ 的上界近似查询 */

似算法和实用的启发算法。这些算法是基于下界近似查询的小型模型性质设计的。给定视图集合 $\mathcal{V}$ 和模式图 Q，基于视图的下界近似查询的小型模型性质如下。其中，$\widehat{Q}$ 是具有与 Q 相同节点的完全图。

引理 3.3 (小型模型性质): 给定视图集合 $\mathcal{V}$ 和模式图 Q，对于任意 Q 的基于 $\mathcal{V}$ 的下界近似查询 Q_l（或完全下界近似查询 Q_l^c），存在 $\widehat{Q}$ 的一个与 Q_l（或 Q_l^c）等价的子图 Q_s (或 Q_s^c 并且 $V_{Q_s^c} = V_Q$)，满足：

- Q_s（或 Q_s^c）也是一个 Q 的基于 $\mathcal{V}$ 的下界近似查询（或完全下界近似查询）；并且
- Q_s（或 Q_s^c）与 Q 比 Q_l（或 Q_l^c）与 Q 的结构更相近，即 $\mathrm{clo}(Q_s, Q) \leq \mathrm{clo}(Q_l, Q)$（或 $\mathrm{clo}(Q_s^c, Q) \leq \mathrm{clo}(Q_l^c, Q)$）。　□

由引理 3.3 可知，只需要从 Q 的子图中即可查找到 Q 的下界近似查询，从而避免了从无穷多种可能中计算 Q 的下界近似查询。基于引理 3.3，本文为最优下界近似查询问题 CLA^c 提出了具有近似度保证的近似算法，为最优完全下界近似查询问题 CLA^c 提出了实用的启发算法。下面首先介绍 CLA^c 问题的算法。

3.5.1　最优完全下界近似查询算法

最优完全下界近似查询问题 CLA^c 的最优完全下界近似查询算法如表 6 所示，记为 CLAsim^c。算法 CLAsim^c 的输入为视图集合 $\mathcal{V}$ 和模式图 Q。如果存在 Q 的基于 $\mathcal{V}$ 的完全下界近似查询，则 CLAsim^c 输出最优的完全下界近似查询。算法 CLAsim^c 的主要工作流程为：（1）CLAsim^c 首先为 $\mathcal{V}$ 中每个视图计算其在 Q 的完全图 $\widehat{Q}$ 中的镜像。（2）而后，CLAsim^c 根据镜像中的节点和边构建 $\widehat{Q}$ 的子图 Q_l。（3）最后，CLAsim^c 验证以

表 6　最优完全下界近似查询算法 CLAsim^c

Algorithm：**最优完全下界近似查询算法 CLAsim^c**

Input: 视图集合 $\mathcal{V}$ 和模式图 Q。
Output: Q 的基于 $\mathcal{V}$ 的最优完全下界近似查询（如果存在）。

1. **for each** V in $\mathcal{V}$ **do**
2. 　　compute image $I_{\widehat{Q}}$[V] of V in $\widehat{Q}$;
3. 　　neg(V) := $E_{I_{\widehat{Q}}[V]} \setminus E_Q$; pos(V) := $E_{I_{\widehat{Q}}[V]} \cap E_Q$;
4. $U := \emptyset$;
5. **while** $E_Q \nsubseteq \bigcup_{V \in U} E_{I_{\widehat{Q}}[V]}$ **and** $\bigcup_{V \in \mathcal{V}}$ pos(V) $\nsubseteq \bigcup_{V' \in U} E_{I_{\widehat{Q}}[V']}$ **do**
6. 　　find V in $\mathcal{V} \setminus U$ *minimizing* $\rho(V) = \frac{|neg(V)|}{|pos(V) \setminus \bigcup_{V' \in U} E_{I_{\widehat{Q}}[V']}|}$;
7. 　　$U := U \cup \{V\}$;
8. $V_l := \bigcup_{V \in U} V_{I_{\widehat{Q}}[V]}$; $E_l := \bigcup_{V \in U} E_{I_{\widehat{Q}}[V]}$; Let Q_l be (V_l, E_l);
9. **if** $E_Q \subseteq E_{Q_l}$ **then return** Q_l;
10. **return** "no";　/* 未找到 Q 的基于 $\mathcal{V}$ 的完全下界近似查询 */

下条件是否成立：如果 Q_l 包含了 Q 中所有边，则 CLAsim^c 返回 Q_l 作为 Q 的基于 $\mathcal{V}$ 的最优完全下界近似查询；否则，算法返回“No”，因为无法找到 Q 的基于 $\mathcal{V}$ 的完全下界近似查询。

具体而言，对于视图集合 $\mathcal{V}$ 中的每个视图 V，算法 CLAsim^c 计算其在 Q 的完全图 $\widehat{Q}$ 中的镜像 $I_{\widehat{Q}}$[V]，并且生成两个额外数据结构：集合 neg(V) 和集合 pos(V)（第 1-3 行）。其中，neg(V) 为镜像 $I_{\widehat{Q}}$[V] 中不属于 Q 的边的集合，pos(V) 为镜像 $I_{\widehat{Q}}$[V] 中既属于 $I_{\widehat{Q}}$[V] 又属于 Q 的边的集合。

而后，算法 CLAsim^c 迭代地从 $\mathcal{V}$ 中识别出与 Q 相关的视图（第 4-7 行）。在每轮迭代步骤中，CLAsim^c 从 $\mathcal{V}$ 中所有直至当前轮次仍未被选中过的视图中选出 ρ(V) 值最小的视图 V（第 6-7 行）。其中，对于每个视图 V，其 ρ(V) 值定义如下：

$$\rho(V) = \frac{|neg(V)|}{|pos(V) \setminus \bigcup_{V' \in U} E_{I_{\widehat{Q}}[V']}|} \tag{3.1}$$

当所有已经选中的视图在 $\widehat{Q}$ 中的镜像能够包含 Q 中的所有边，迭代步骤停止（第 5 行）。

直观理解，ρ(V) 是视图 V 中包含的不属于 Q 的边的个数除以如果选中 V 能够新覆盖 Q 中边的个数得到的平均数。算法 CLAsim^c 在每轮迭代步骤中都选择 ρ(V) 值最小的视图 V，从而更多覆盖 Q 中的边并且更少引入 Q 中不存在的边。结合引理 3.3，本文将在之后证明上述迭代步骤能够逐步生成与 Q 拓补结构更相近的下界近似查询。

最后，算法 CLAsim^c 根据迭代步骤中选中的所有视图在 $\widehat{Q}$ 中镜像中的节点和边构建图 Q_l（第 8 行）。CLAsim^c 最终验证以下条件是否成立：如果 Q_l 包含了 Q 中所有边，则算法返回 Q_l 作为 Q 的基于 $\mathcal{V}$ 的最优完全下界近似查询；否则，算法返回“No”，因为无法找到 Q 的基于 $\mathcal{V}$ 的完全下界近似查询（第 9-10 行）。

例 3.10: 以模式图 Q_1 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。算法 CLAsim^c 为视图集合 $\mathcal{V}$ 中每个视图 V_i ($i \in [1,4]$) 计算其在 Q_1 的完全图 $\widehat{Q_1}$ 中的镜像（第 1-3 行）。而后，算法 CLAsim^c 执行迭代步骤迭代地从 $\mathcal{V}$ 中选取出与 Q_1 相关的视图。在第一次迭代步骤中，CLAsim^c 选取视图 V_1 并将其并入视图集合 U 中，因为 $\rho(\mathsf{V}_1) = 0$ 相较于 $\rho(\mathsf{V}_2) = 0, \rho(\mathsf{V}_3) = \frac{1}{2}$ 和 $\rho(\mathsf{V}_4) = +\infty$ 值最小。在第二次迭代步骤中，CLAsim^c 选取视图 V_2 并将其并入视图集合 U 中，因为 $\rho(\mathsf{V}_2) = 0$ 相较于 $\rho(\mathsf{V}_3) = \frac{1}{2}$ 和 $\rho(\mathsf{V}_4) = +\infty$ 值最小。在第三次迭代步骤中，CLAsim^c 选取视图 V_3 并将其并入视图集合 U 中，因为 $\rho(\mathsf{V}_2) = 1$ 相较于 $\rho(\mathsf{V}_4) = +\infty$ 值最小。此时，算法 CLAsim^c 判定停止条件 $E_{Q_1} \subseteq \bigcup_{\mathsf{V}_i \in U(i \in [1,3])} E_{I_{\widehat{Q}_1}[\mathsf{V}_i]}$ 成立，迭代步骤至此停止（第 4-7 行）。最后，算法 CLAsim^c 构建图 Q_l，如图 7 所示。算法最终经过验证得知 Q_l 包含了 Q_1 中所有边。所以，算法 CLAsim^c 返回 Q_l 作为 Q_1 的基于 $\mathcal{V}$ 的最优完全下界近似查询（第 8-10 行）。 □

根据定理 3.5（2）可知，CLA^c 问题不存在任何常数近似度保证的近似算法。然而，本文将证明算法 CLAsim^c 是 CLA^c 问题的具有多项式近似度保证的近似算法。对于 $E_{\widehat{Q}} \setminus E_Q$ 中的每条边 e，本文用 occ(e) 表示 $\mathcal{V}$ 中在 $\widehat{Q}$ 中的镜像包含 e 的视图的个数，即 occ(e) = $|\{\mathsf{V} \mid e \in E_{I_{\widehat{Q}}[\mathsf{V}]}, \mathsf{V} \in \mathcal{V}\}|$。

定理 3.6: 算法 CLAsim^c 是近似度为 $\max_{e \in E_{\widehat{Q}} \setminus E_Q} \mathsf{occ}(e) \cdot \ln(\max_{\mathsf{V} \in \mathcal{V}} |E_{I_{\widehat{Q}}[\mathsf{V}]} \cap E_Q|)$ 的近似算法。如果存在 Q 的基于 $\mathcal{V}$ 的完全下界近似查询，则算法 CLAsim^c 可以在 $O(|\mathcal{V}||Q|^2)$-时间内返回 Q 的基于 $\mathcal{V}$ 的最优完全下界近似查询。 □

证明 根据定理 3.1和引理 3.1，可以证明如果存在 Q 的基于 $\mathcal{V}$ 的完全下界近似查询，则算法 CLAsim^c 可以在 $O(|\mathcal{V}||Q|^2)$-时间内返回 Q 的基于 $\mathcal{V}$ 的最优完全下界近似查询。这同时证明了定理 3.3（b）的正确性。下面证明算法 CLAsim^c 近似度的正确性。

本文用 Q_{OPT} 表示 Q 的基于 $\mathcal{V}$ 的最优完全下界近似查询。根据引理 3.1 可知，一定存在 $\mathcal{V}$ 的一个视图子集 U_{OPT}，使得 Q_{OPT} 能够通过合并 U_{OPT} 中所有视图在 $\widehat{Q}$ 中的镜像计算得出，即 $E_{Q_{\mathrm{OPT}}} = \bigcup_{\mathrm{V}\in U_{\mathrm{OPT}}} E_{I_{\widehat{Q}}[\mathrm{V}]}$。分析可知以下等式成立：

$$\mathrm{clo}(Q_l, Q) = |\cup_{\mathrm{V}\in U} \mathrm{neg}(\mathrm{V})| \tag{3.2}$$

$$\mathrm{clo}(Q_{\mathrm{OPT}}, Q) = |\cup_{\mathrm{V}\in U_{\mathrm{OPT}}} \mathrm{neg}(\mathrm{V})| \tag{3.3}$$

此外，对于任意一个视图集合 U'，以下不等式成立：

$$|\cup_{\mathrm{V}\in U'} \mathrm{neg}(\mathrm{V})| \le \sum_{\mathrm{V}\in U'} |\mathrm{neg}(\mathrm{V})| = \sum_{e\in\cup_{\mathrm{V}\in U'}\mathrm{neg}(\mathrm{V})} \mathrm{occ}(e) \le \max_{e\in E_{\widehat{Q}}\setminus E_Q}\mathrm{occ}(e)\cdot|\cup_{\mathrm{V}\in U'}\mathrm{neg}(\mathrm{V})| \tag{3.4}$$

所以，根据等式 3.2、等式 3.3 和不等式 3.4 可以推导得出以下不等式成立：

$$\frac{\mathrm{clo}(Q_l, Q)}{\mathrm{clo}(Q_{\mathrm{OPT}}, Q)} \le \frac{\sum_{\mathrm{V}\in U}|\mathrm{neg}(\mathrm{V})|}{|\bigcup_{\mathrm{V}\in U_{OPT}}\mathrm{neg}(\mathrm{V})|} \le \frac{\max_{e\in E_{\widehat{Q}}\setminus E_Q}\mathrm{occ}(e)\cdot\sum_{\mathrm{V}\in U}|\mathrm{neg}(\mathrm{V})|}{\sum_{\mathrm{V}\in U_{\mathrm{OPT}}}|\mathrm{neg}(\mathrm{V})|} \tag{3.5}$$

另外，可以看出算法 CLAsim^c 中的 while 循环步骤实际构建了一个从“计算视图集合 U 从而最小化 $\sum_{\mathrm{V}\in U}\mathrm{neg}(\mathrm{V})$ 值问题”到 *minimum weighted set cover*（MWSC）问题 [74] 的具有近似度保证的归约。该归约过程分别将模式图 Q 的边集 E_Q，$\mathcal{V}$ 中视图 V 的“正向”边集合 pos(V)，$\mathcal{V}$ 中视图 V 的“逆向”边集合包含的边的个数 |neg(V)|，分别对应于 MWSC 问题中的论域，集合，和集合的权重；并将 $\mathcal{V}$ 对应于 MWSC 问题中所有集合的组合。已知 MWSC 问题存在近似度为 $\ln(\max_{\mathrm{V}\in\mathcal{V}}|E_{I_{\widehat{Q}}[\mathrm{V}]}\cap E_Q|)$ 的近似算法 [74]。由近似算法理论可知，以上被归约问题的近似算法也具有与归约问题（MWSC）相同的近似复杂度，所以以下不等式成立：

$$\frac{\sum_{\mathrm{V}\in U}|\mathrm{neg}(\mathrm{V})|}{\sum_{\mathrm{V}\in U_{\mathrm{OPT}}}|\mathrm{neg}(\mathrm{V})|} \le \frac{\ln(\max_{\mathrm{V}\in\mathcal{V}}|E_{I_{\widehat{Q}}[\mathrm{V}]}\cap E_Q|)\cdot\sum_{\mathrm{V}\in U_*}|\mathrm{neg}(\mathrm{V})|}{\sum_{\mathrm{V}\in U_{\mathrm{OPT}}}|\mathrm{neg}(\mathrm{V})|} \le \ln(\max_{\mathrm{V}\in\mathcal{V}}|E_{I_{\widehat{Q}}[\mathrm{V}]}\cap E_Q|) \tag{3.6}$$

其中，U_* 是归约问题（MWSC）的最优解。根据不等式 3.5 和 3.6，可以得到算法 CLAsim^c 近似度如下：

$$\frac{\mathrm{clo}(Q_l, Q)}{\mathrm{clo}(Q_{\mathrm{OPT}}, Q)} \le \max_{e\in E_{\widehat{Q}}\setminus E_Q}\mathrm{occ}(e)\cdot\ln(\max_{\mathrm{V}\in\mathcal{V}}|E_{I_{\widehat{Q}}[\mathrm{V}]}\cap E_Q|) \tag{3.7}$$

所以，算法 CLAsim^c 是 CLA^c 问题的 $\max_{e\in E_{\widehat{Q}}\setminus E_Q}\mathrm{occ}(e)\cdot\ln(\max_{\mathrm{V}\in\mathcal{V}}|E_{I_{\widehat{Q}}[\mathrm{V}]}\cap E_Q|)$-近似算法。 □

表 7　最优下界近似查询算法 CLAsim

Algorithm：最优下界近似查询算法 CLAsim

Input: 视图集合 $\mathcal{V}$ 和模式图 Q。
Output: Q 的基于 $\mathcal{V}$ 的最优下界近似查询（如果存在）。

1. **for each** V in $\mathcal{V}$ **do** compute image $I_{\widehat{Q}}$[V] of V in $\widehat{Q}$ of Q;
2. $V_l := \bigcup_{V\in\mathcal{V}} V_{I_{\widehat{Q}}[V]}$; $E_l := \bigcup_{V\in\mathcal{V}} E_{I_{\widehat{Q}}[V]}$; Let Q_l be (V_l, E_l);
3. **while** there exist u, u' in V_{Q_l} with $(u, u') \in E_Q \setminus E_{Q_l}$ **do**　/* Q_l 还不是 Q 的诱导子图 */
4. 　**for each** V in $\mathcal{V}$ with u, u' in $V_{I_{\widehat{Q}}[V]}$ **do**
5. 　　remove the view image $I_{\widehat{Q}}$[V] from Q_l;
6. 　**if** both u and u' remain in V_{Q_l} of Q_l **then**
7. 　　$u_0 := \arg\min_{v\in\{u,u'\}} f(v)$;
8. 　　remove from Q_l all view images $I_{\widehat{Q}}$[V] that contain u_0;
9. **if** $E_{Q_l} \neq \emptyset$ **then return** Q_l;
10. **return** "no";　/* 未找到 Q 的基于 $\mathcal{V}$ 的下界近似查询 */

3.5.2　最优下界近似查询算法

本章将设计用于解决最优下界近似查询问题 CLA 的算法。由定理 3.3（c）可知，给定视图集合 $\mathcal{V}$ 和模式图 Q，下界近似查询的存在性问题即判定是否存在 Q 的基于 $\mathcal{V}$ 的下界近似查询已经是 NP-难问题。而优化问题比存在性问题更为复杂。根据定理 3.5 可知，计算 Q 的基于 $\mathcal{V}$ 的最优下界近似查询问题是 NP-难问题，并且不存在常数近似度保证的近似算法。尽管问题的计算复杂度较高难以设计高效的精确算法，但是本文在仔细分析 CLA 问题特征的基础上设计了实用的启发算法。

最优下界近似查询 CLA 问题的最优下界近似查询算法如表 7 所示，记为CLAsim 。算法CLAsim 的主要工作流程为：（1）CLAsim 首先为 $\mathcal{V}$ 中每个视图计算其在 Q 的完全图 $\widehat{Q}$ 中的镜像，并根据镜像中的节点和边构建 $\widehat{Q}$ 的子图 Q_l（第 1-2 行）。（2）之后，CLAsim 迭代地从构建 Q_l 的视图集合中去除掉不相关的视图，直至 Q_l 成为 Q 的一个诱导子图（第 3-8 行）。（3）最后，CLAsim 判定如果 Q_l 的边集不为空，则算法返回 Q_l 作为 Q 的基于 $\mathcal{V}$ 的最优下界近似查询；否则，算法返回“No”，因为无法找到 Q 的基于 $\mathcal{V}$ 的下界近似查询（第 9-10 行）。

具体而言在步骤（2）中，算法CLAsim 按如下方法迭代地从构建 Q_l 的视图集合中去除掉不相关的视图：在每轮迭代步骤中，CLAsim 首先检查 Q_l 是否已经是 Q 的一个诱导子图，即检查 Q 中是否存在一条边 (u,u')，该边的两个端点 $u,u' \in V_{Q_l}$ 但是 (u,u') 不属于 Q_l 的边集 E_{Q_l}（第 3 行）。如果存在这样的边 (u,u')，则说明 Q_l 还不是 Q 的诱导子图。此时，算法CLAsim 从构建 Q_l 的视图集合中找出在 $\widehat{Q}$ 中的镜像同时包含节点 u 和 u' 的视图，并把这些视图从构建 Q_l 的视图集合中去除（第 4-5 行）。而后，CLAsim 再次检查 Q_l 是否仍然同时包含节点 u 和 u'（第 6 行）。如果包含，则CLAsim 计算节点 u 和 u' 中哪个节点的 $f(v)$ 值较小（第 7 行），其中 $f(v)=|N(v)|+|H(v)|$，并且

$$N(v)=\{(u,u')\,|\,u,u' \in V_{Q_l(v)},(u,u') \in E_Q \setminus E_{Q_l(v)}\} \tag{3.8}$$

$$H(v)=\{(u,u')\,|(u,u') \in E_Q \cap (E_{Q_l} \setminus E_{Q_l(v)})\} \tag{3.9}$$

公式 3.8 和公式 3.9 中，$Q_l(v)$ 是一个点集为 $\bigcup_{\mathsf{V}\in\mathcal{V}\setminus S(v)} V_{I_{\widehat{Q}}[\mathsf{V}]}$，边集为 $\bigcup_{\mathsf{V}\in\mathcal{V}\setminus S(v)} E_{I_{\widehat{Q}}[\mathsf{V}]}$）的模式图，其中 $S(v)=\{\mathsf{V}\,|\,v \in V_{I_{\widehat{Q}}[\mathsf{V}]},\mathsf{V} \in \mathcal{V}\}$。

直观理解，$N(v)$ 包含 Q_l 中因为去除掉所有镜像中含有节点 v 的视图后所产生的所有新的“坏边”，$H(v)$ 包含 Q_l 中去除掉所有镜像中含有节点 v 的视图后所去除掉的所有“好边”。显然，如果 $f(v)$ 值越小，则从 Q_l 中去除掉所有与 v 相关的视图对于 Q_l 与 Q 结构距离 $\mathrm{clo}(Q_l,Q)$ 的影响越小。所以，算法CLAsim 从两个节点 $\{u,u'\}$ 中选取出 $f(v)$ 值较小的节点，并将构建 Q_l 的镜像中含有这个节点的视图全部去除（第 8 行）。

例 3.11: 以模式图 Q_2 和视图集合 $\mathcal{V}$ 为例，如图 7 所示。算法 CLAsim 为视图集合 $\mathcal{V}$ 中每个视图 V_i $(i \in [1,4])$ 计算其在 Q_2 的完全图 $\widehat{Q_2}$ 中的镜像，并根据 4 个镜像中的节点和边构建 Q_l（第 1-2 行）。之后，算法验证得知 Q_l 还不是 Q_2 的一个诱导子图。因为 Q_2 中存在一条边 $(\mathsf{PM},\mathsf{BA})$，该边的两个端点 PM 和 BA 在 Q_l 中，但是边 $(\mathsf{PM},\mathsf{BA})$ 不在 Q_l 中（第 3 行）。为了将 Q_l 构建为 Q_2 的诱导子图，算法 CLAsim 执行迭代步骤从构建 Q_l 的视图中迭代地去除掉“坏”的视图。计算过程中，CLAsim 首先需要从 Q_l 中去除掉镜像同时包含节点 PM 和 BA 的所有视图。经查找可知不存在这样的视图，所以此步不需要去除任何视图（第 4-5 行）。而后，CLAsim 需要从 Q_l 中去除掉镜像包含节点 PM 和 BA 其中之一的所有视图。经计算得知 $f(\mathsf{BA})=0$ 小于 $f(\mathsf{PM})=2$（$N(\mathsf{PM})=\emptyset$，$H(\mathsf{PM})=\{(\mathsf{PM},\mathsf{SD}),(\mathsf{PM},\mathsf{UD})\}$）。所以，算法 CLAsim 从 Q_l 中去除掉镜像包含节点 BA 的所有视图，即视图 V_4（$I_{\widehat{Q_2}}[\mathsf{V}_4]$ 包含边 $(\mathsf{UD},\mathsf{BA})$）（第 6-8 行）。剩余的 Q_l 如图 7 所示。此时，Q_l 为 Q_2 的诱导子图。算法 CLAsim 判定 Q_l 的边集不为空，并将其返回作为 Q_2 的基于

$\mathcal{V}$ 的最优下界近似查询（第 9 行）。 □

算法正确性和复杂性分析。算法 CLAsim 的正确性由定理 3.1 和引理 3.1 保证。通过建立从模式图 Q 的所有节点到 $\mathcal{V}$ 中所有视图的镜像的倒排索引，可以保证算法 CLAsim 在 $O(|\mathcal{V}||Q|^2)$ 时间内执行完毕计算过程。

3.6 基于视图的近似查询理论扩展

本文在章节 3.1 至章节 3.5 提出了一套基于视图的近似图匹配查询理论及方法。本文在论述该方法工作原理时是以图模拟为图匹配查询语义。然而，基于视图的近似图匹配查询方法是通用方法，适用于任意一种图匹配查询语义。为了验证该方法的通用性，本章首先将该方法应用于子图同构图匹配查询语义（章节 3.6.1- 3.6.3），并在最后抽象出基于视图的近似图匹配查询方法的通用计算框架（章节 3.6.4）。

本章首先针对子图同构语义设计了基于视图的近似图匹配查询方法。章节 3.6.1 和章节 3.6.2 对在子图同构语义下，基于视图的近似图匹配查询方法的基本性质和八个基本问题进行了理论分析。章节 3.6.3 设计了基于视图的图匹配查询算法。

3.6.1 基本性质

本章分析了在子图同构语义下基于视图的近似图匹配查询方法的基本性质，包括查询包含和基于视图的图匹配查询问题的基本性质。其中，需要说明的是在子图同构语义下基于视图的图匹配查询方法在本文之前未被相关工作提出和研究过。如章节 2.3.1 介绍，相关工作只提出了在图模拟语义下基于视图的图匹配查询方法 [46, 47]。

（完全）上界/下界包含。基于子图同构语义的四种包含关系定义与基于图模拟语义的四种包含关系定义一致。其性质刻画如下所示。

定理 3.7: 对于模式图 Q，Q_u 和 Q_l，

(1) $Q \sqsubseteq_{\mathcal{U}} Q_u$ 当且仅当 $Q_u \lhd Q$；

(2) $Q \sqsubseteq^c_{\mathcal{U}} Q_u$ 当且仅当 $Q_u \lhd Q$ 并且 $V_Q = V_{I_Q[Q_u]}$；

(3) $Q_l \sqsubseteq_{\mathcal{L}} Q$ 当且仅当存在 Q 的诱导子图 Q_s 满足 $Q_s \lhd Q_l$ 并且 $V_{Q_l} = V_{I_{Q_l}[Q_s]}$；

(4) $Q_l \sqsubseteq^c_{\mathcal{L}} Q$ 当且仅当 $Q \lhd Q_l$ 并且 $V_{Q_l} = V_{I_{Q_l}[Q]}$； □

基于视图的图匹配查询。与图模拟语义下基于视图的图匹配查询问题类似，本文用镜像这一概念刻画子图同构语义下基于视图的图匹配查询问题的基本性质。

定理 3.8: 对于视图集合 $\mathcal{V}$ 和模式图 Q,

(1) 模式图 Q 可以被视图集合 $\mathcal{V}$ 回答当且仅当 $E_Q = \bigcup_{V \in \mathcal{V}} E_{I_Q[V]}$ 成立;

(2) 判定模式图 Q 是否可以被视图集合 $\mathcal{V}$ 回答是 NP-完全问题。 □

与图模拟语义下基于视图的图匹配查询问题存在多项式时间算法不同，子图同构语义下基于视图的图匹配查询问题的计算复杂度很高，是 NP-完全问题。

3.6.2 基本问题和计算复杂性分析

本章将对在子图同构语义下基于视图的近似图匹配查询的存在性问题和最优化问题进行理论分析。

近似查询的存在性。本章首先对在子图同构语义下，基于视图的上界近似查询，完全上界近似查询，下界近似查询和完全下界近似查询的存在性问题，即 EUA，EUA^c，ELA 和 ELA^c 问题进行分析。这些问题有以下性质。

定理 3.9: 给定视图集合 $\mathcal{V}$ 和模式图 Q,

(1) 存在一个 Q 的基于 $\mathcal{V}$ 的上界近似查询当且仅当存在视图 $V \in \mathcal{V}$ 使得图匹配结果 $V(Q) \neq \emptyset$;

(2) 存在一个 Q 的基于 $\mathcal{V}$ 的完全上界近似查询当且仅当 $V_Q = \bigcup_{V \in \mathcal{V}} V_{I_Q[V]}$;

(3) 存在一个 Q 的基于 $\mathcal{V}$ 的完全下界近似查询当且仅当 $E_Q \subseteq \bigcup_{V \in \mathcal{V}} E_{I_{\bar{Q}}[V]}$;

(4) EUA，EUA^c，ELA 和 ELA^c 都是 NP-完全问题。 □

与图模拟语义下基于视图的近似图匹配查询问题不同，子图同构语义下基于视图的上界和下界近似图匹配查询的存在性问题已经是 NP-完全问题。因为判定是否存在 Q 的基于 $\mathcal{V}$ 的近似图匹配查询问题包含了判定一个视图 V 与模式图 Q 是否基于子图同构语义匹配的问题，而判定基于子图同构语义匹配是 NP-完全问题。

图模拟语义下基于视图的下界近似查询的存在性问题 ELA 是 NP-难问题。尽管子图同构语义下的 ELA 问题需要执行子图同构计算步骤，但是其计算复杂度并未超越 NP 问题。另外，与图模拟语义不同，子图同构语义下基于视图的下界近似查询的存在性问题 ELA 和完全下界近似查询的存在性问题 ELA^c 具有相同的计算复杂度。

最优近似查询。利用章节 3.3.2 中在图模拟语义下定义的结构距离函数 clo，本章对在子图同构语义下，基于视图的最优上界近似查询，最优完全上界近似查询，最优下界近似查询和最优完全下界近似查询问题，即 CUA，CUA^c，CLA 和 CLA^c 进行理论分析。这些

问题有以下性质。

定理 3.10: 给定视图集合 $\mathcal{V}$ 和模式图 Q,

(1) 模式图 $Q_u(\bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]}, \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_Q[\mathsf{V}]})$ 是 Q 的基于 $\mathcal{V}$ 的最优上界近似查询；

(2) 如果 $\bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]} = V_Q$，则 $Q_u^c(\bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]}, \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_Q[\mathsf{V}]})$ 是 Q 的基于 $\mathcal{V}$ 的最优完全上界近似查询；

(3) CUA，CUA^c，CLA 和 CLA^c 问题的判定问题都是 NP-完全问题。 □

尽管在子图同构语义下基于视图的近似图匹配查询方法都需要执行子图同构（NP-完全）计算步骤，然而在子图同构语义下计算复杂度最高的 ELA，CLA 和 CLA^c 问题仍然保持了和相应在图模拟语义下 ELA，CLA 和 CLA^c 问题同样的计算复杂度。

3.6.3 算法设计

本章将分别设计在子图同构语义下，基于视图的最优（完全）上界近似查询算法和最优（完全）下界近似查询算法。而后设计在子图同构语义下基于视图的图匹配查询算法。

最优上界和下界近似查询算法。本章设计在子图同构语义下，基于视图的最优上界近似查询算法，最优完全上界近似查询算法，最优下界近似查询算法和最优完全下界近似查询算法，分别记为 CUAiso^c，CUAiso，CLAiso^c 和 CLAiso。这些算法是基于章节 3.4 和章节 3.5 中提出的图模拟语义下相对应的最优近似查询算法的修改。只需修改一点即可，即用子图同构语义代替图模拟语义计算视图在模式图中的镜像。这些算法的正确性由章节 3.6.1 和章节 3.6.2 中的性质、定理和子图同构语义下类似引理 3.2 和引理 3.3 的小型模型性质保证。

子图同构语义基于视图的图匹配查询算法。根据定理 3.8，本章设计子图同构语义的基于视图的图匹配查询算法，记为 QAViso，作为在子图同构语义下基于视图的近似图匹配查询方法的支撑算法。算法 QAViso 的输入为视图集合 $\mathcal{V}$ 和模式图 Q，算法首先验证 Q 是否可以被 $\mathcal{V}$ 回答。如果可以，则算法 QAViso 返回一个 Q 的基于 $\mathcal{V}$ 的查询计划 $\mathcal{P}$。对于任意数据图 G，查询计划 $\mathcal{P}(G)$ 只需访问 $\mathcal{V}$ 中视图 V 在 G 中的镜像 $I_G[\mathsf{V}]$ 即可计算得出 Q 在 G 中的图匹配结果 $Q(G)$。

查询计划 $\mathcal{P}$ 是一个操作序列 $T_1 = \delta_1, \ldots, T_n = \delta_n$，使得对于任意数据图 G, $T_n(G) = Q(G)$。其中，操作 δ_i 为以下三种操作之一，并且在操作的执行过程中只需访问视图在数据图中的镜像。

表 8　子图同构语义基于视图的图匹配查询算法 QAViso

Algorithm：子图同构语义基于视图的图匹配查询算法 QAViso

Input: 视图集合 $\mathcal{V}$ 和模式图 Q。
Output: Q 的基于 $\mathcal{V}$ 的查询计划 $\mathcal{P}$（如果 Q 可以被 $\mathcal{V}$ 回答）。

1. $\mathcal{P}$:= [ ]; $S := \emptyset$;
2. **for each** V in $\mathcal{V}$ **do if** $\mathsf{V}(Q) \neq \emptyset$ **then** $S := S \cup \{\mathsf{V}\}$;
3. **if** $\bigcup_{\mathsf{V}\in S} E_{I_Q[\mathsf{V}]} \neq E_Q$ **then return** "no";　/* 假设 $S = \{\mathsf{V}_1, ..., \mathsf{V}_m\}$ */
4. initialize $\mathcal{P}$ to $[T_1 = I_G[\mathsf{V}_1]]$; $V := \emptyset$;
5. **for** i in $[1, m-1]$ **do**
6. 　append $T_{i+1} = T_i \bowtie I_G[\mathsf{V}_{i+1}]$ to $\mathcal{P}$;
7. 　add common nodes in T_i and $I_G[\mathsf{V}_{i+1}]$ to V;
8. append $T_{m+1} = \sigma(T_m, V, d_Q)$ to $\mathcal{P}$;
9. append $T_{m+2} = \mathsf{mat}(Q, T_{m+1})$ to $\mathcal{P}$;
10. **return** $\mathcal{P}$;

(a) Join $G_1 \bowtie G_2$。其中 G_1 和 G_2 为两个图。$G_1 \bowtie G_2$ 返回图 $(V_{G_1} \cup V_{G_2}, E_{G_1} \cup E_{G_2})$ 中所有既包含 G_1 中节点又包含 G_2 中节点的连通分量。

(b) Filter $\sigma(G, V, r)$。其中 G 是图，V 是 G 的节点集的子集，r 是一个正整数。$\sigma(G, V, r)$ 返回 G 的一个诱导子图 $G_{V,r}$。其中，$G_{V,r}$ 的节点集是由 G 中与 V 中任意节点之间距离不超过 r 的节点构成的集合。

(c) Match $\mathsf{mat}(Q, G)$。其中 Q 是模式图，G 是数据图。$\mathsf{mat}(Q, G)$ 计算得出 Q 在 G 中基于子图同构语义的图匹配结果 $Q(G)$。

子图同构语义的基于视图的图匹配查询算法 QAViso 如表 8 所示。该算法为 Q 生成一个基于 $\mathcal{V}$ 的查询计划 $\mathcal{P}$，其主要计算步骤如下。

算法 QAViso 首先判定 Q 是否可以被 $\mathcal{V}$ 回答。如果可以，则根据定理 3.8 找出 $\mathcal{V}$ 中所有与 Q 相关的视图（第 2-3 行）。而后，算法 QAViso 执行三个主要步骤生成查询计划 $\mathcal{P}$。（i）首先，QAViso 将所有相关视图在数据图中的镜像用 Join 操作合并起来（第 4-7 行）。（ii）而后，QAViso 执行 Filter 操作从步骤（i）生成的合并图中过滤出一个子图。该子图的节点包括以 Join 操作时涉及的公共节点为中心以 Q 的直径 d_Q 为半径内的

所有节点（第 8 行）。（iii）最后，QAViso 返回 Q 在步骤（ii）生成的过滤子图中基于子图同构的图匹配结果（第 9-10 行）。

备注。可以为算法 QAViso 设计多种优化技术，用于优化从视图集合 $\mathcal{V}$ 中找出与 Q 相关视图的步骤，从而减少后续合并、过滤生成最终图匹配结果过程的计算量。章节 2.3.1 详细介绍了 [46] 提出的在图模拟语义下基于视图的图匹配查询算法 QAV 中采用的两种视图选取优化策略。算法 QAViso 可以采取相同的优化设计思路优化视图选取步骤，以提高整体查询计算效率。

3.6.4 基于视图的近似查询方法的通用计算框架

本文提出了一套通用的基于视图的近似图匹配查询方法，适用于任意一种图匹配查询语义。本文在章节 3.4、章节 3.5 和章节 3.6.3 中分别设计了图模拟语义和子图同构语义的基于视图的近似图匹配查询方法。基于这些算法设计，本章将抽象出基于视图的近似图匹配查询方法的通用计算框架，如下所示。

（1）给定视图集合 $\mathcal{V}$ 和模式图 Q，首先调用基于视图的图匹配查询算法（如图模拟语义的 QAV [46] 算法、子图同构语义的 QAViso 算法等），判定 Q 是否可以被 $\mathcal{V}$ 精确回答。

（2）如果 $\mathcal{V}$ 可以精确回答 Q，则调用基于视图的图匹配查询算法（如 QAV [46]、QAViso 等），生成 Q 的基于 $\mathcal{V}$ 的精确查询计划，使得查询引擎只访问已经存储的 $\mathcal{V}$ 中视图在 G 中的镜像即可得到 Q 在 G 中的图匹配查询结果。

（3）如果 $\mathcal{V}$ 不可以精确回答 Q，则调用基于视图的近似图匹配查询算法（如 CUAsim^c、CUAsim 、CLAsim^c、CLAsim 、CUAiso^c、CUAiso 、CLAiso^c、CLAiso 等），判定是否存在 Q 的基于 $\mathcal{V}$ 的上界和下界近似查询。如果存在，则找到 Q 的基于 $\mathcal{V}$ 的最优上界和下界近似查询 Q_u 和 Q_l。

（4）Q_u 和 Q_l 的生成方法保证其肯定可以被 $\mathcal{V}$ 精确回答。通过调用基于视图的图匹配查询算法（如 QAV [46]、QAViso 等），生成 Q_u 和 Q_l 的基于 $\mathcal{V}$ 的精确查询计划，即可得到 Q 在 G 中的上界和下界近似图匹配查询结果。

3.7　实验评估

本章采用两个真实数据集，设计有效性测试和效率测试两组实验来评估基于视图的近似图匹配查询方法的性能。

3.7.1　实验设置

本章首先介绍实验设置和数据集等实验相关信息。

数据图。本章采用两个真实数据集来评估方法的性能。

（1）知识图谱（DBpedia）版本号为 DBpedia 201504 [75]。其包含 4.43M 个节点，8.43M 条边和 735 个节点标签数据。

（2）YouTube 视频网络图（YouTube）是从视频网站 YouTube 上获取的表示视频之间推荐关系的网络图 [76]。其包含 2.03M 个视频节点，12.22M 条视频之间的推荐边和 398 个节点标签数据。其中，从视频节点 x 到视频节点 y 的边表示如果一个用户喜欢观看视频 x，则他很可能喜欢观看视频 y。

模式图。本章通过实现一个模式图自动生成器来生成模式图。自动生成器由 3 个参数控制，从而生成不同的模式图：$|V_Q|$ 控制模式图的节点个数，$|E_Q|$ 控制模式图边的个数，标签函数 l_v 从实验用数据图的字表 Σ 中为每个节点选取一个标签。本章用 $(|V_Q|,|E_Q|)$ 表示模式图 $Q(V_Q,E_Q)$ 的大小。本章通过调节参数 $|V_Q|$ 从 3 逐步变化至 11，参数 $|E_Q|$ 从 5 逐步变化至 13，总共生成了 100 个模式图作为实验的查询输入。

视图。根据 [46, 47] 中所述方法，本章为每个数据图生成了 60 个视图。这些视图有以下四个大小：(2,1)，(3,2)，(4,3) 和 (4,4)。相同大小的视图具有不同的拓补结构。（a）对于 DBpedia，视图在 DBpedia 中的镜像平均包含 72K 个节点和边，所有视图镜像占用物理存储空间大小为整个 DBpedia 物理存储空间的 32.58%。（b）对于 YouTube，视图在 YouTube 中的镜像平均包含 80K 个节点和边，所有视图镜像占用物理存储空间大小为整个 DBpedia 物理存储空间的 34.29%。

算法。本章用 C++ 实现了以下算法：（1）本文提出的基于视图的最优（完全）上界/下界近似查询算法：图模拟语义的 CUAsim^c、CUAsim 、CLAsim^c、CLAsim 算法和子图同构语义的 CUAiso^c、CUAiso 、CLAiso^c、CLAiso 算法；（2）本文提出的子图同构语义的基于视图的图匹配查询算法 QAViso，作为子图同构语义的基于视图的近似图匹配查询方法的支撑算法；（3）图模拟语义的基于视图的图匹配查询算法 QAVsim [46]；（4）传

表 9　由 $\mathcal{V}$ 精确及近似回答模式图所占百分比

% of views used	25%	50%	75%	100%
DBpedia:%-ap: sim(sub)	65(53)%	74(64)%	85(78)%	95(88)%
YouTube:%-ap: sim(sub)	72(61)%	80(69)%	88(81)%	98(90)%
DBpedia:%-ap^c: sim(sub)	53(41)%	64(52)%	75(68)%	88(82)%
YouTube:%-ap^c: sim(sub)	58(51)%	70(64)%	83(77)%	92(85)%
DBpedia:%-answerable	0(0)%	3(0)%	8(3)%	12(8)%
YouTube:%-answerable	0(0)%	5(1)%	10(4)%	17(10)%

统的用于计算基于图模拟和子图同构的图匹配查询算法 gSim [13] 和 QAViso（使用 C++ Boost Graph 库）。

所有实验都是在一台配置为 Intel Core(TM) Duo 3.00GHz CPU and 16GB of memory 的机器上测试。每组实验重复 10 次，并取平均数记录于以下实验结果章节中。

3.7.2　有效性测试

本章测试基于视图的近似图匹配查询方法的有效性，即评估采用该方法相对于直接执行图匹配查询方法对于图匹配查询引擎查询效率和查询结果的影响。

（1）基于视图的近似图匹配查询的存在性测试。通过变化使用的视图占生成视图总个数的百分比，测试模式图中（a）存在基于视图的上界近似查询或下界近似查询的模式图占所有测试模式图的百分比，记为%-ap，和（b）存在基于视图的完全上界近似查询或完全下界近似查询的模式图占所有测试模式图的百分比，记为%-ap^c。另外，测试模式图中（c）可以被视图精确回答的模式图占所有测试模式图的百分比，记为%-answerable。测试结果如表 9 所示。

实验发现如下所示：（i）50% 以上的模式图存在基于可用视图的近似查询，其中大部分模式图存在完全近似查询。观察可知，当 75% 的视图被使用时，DBpedia 上 85% 的模式图存在基于图模拟语义的近似查询，YouTube 上 81% 的的模式图存在基于子图同构语义的近似查询；DBpedia 上 75% 的模式图存在基于图模拟语义的完全近似查询，YouTube 上 77% 的模式图存在基于子图同构语义的近似查询。（ii）基于图模拟语义的模式图比基于子图同构语义的模式图更容易存在基于视图的（完全）近似查询。（iii）极少数模式图可以被视图精确回答。尽管使用 100% 的视图，在 DBpedia 上只有 12% 的

模式图在图模拟语义下可以被视图精确回答；在 YouTube 上只有 17% 的模式图在图模拟语义下可以被视图精确回答。当考虑子图同构语义时，上述百分比值更低。以上实验结果进一步验证了对基于视图的近似图匹配查询方法开展研究的必要性。

（2）基于视图的最优近似图匹配查询的准确性测试。本章设计三个准确性评价标准和两组实验，来评估基于视图的最优近似图匹配查询方法生成的最优近似查询的图匹配查询结果相对精确图匹配查询结果的准确性，从查询结果的准确性方面评估该方法的有效性。

本章设计三个准确性评价标准如下：（a）F-measure $F(Q, Q', G)$ 衡量 Q 的上界近似查询或下界近似查询 Q' 的图匹配查询结果相对于 Q 在数据图 G 中图匹配查询结果的准确性；（b）强 F-measure $F_s(Q, Q_u, Q_l, G)$ 和（c）弱 F-measure $F_w(Q, Q_u, Q_l, G)$ 衡量 Q 的一对上界和下界近似查询 (Q_u, Q_l) 组合的图匹配查询结果相对于 Q 在数据图 G 中图匹配查询结果的准确性。具体而言，对于任意两个集合 S 和 S'，定义其精确率为 $\mathsf{prec}(S', S) = \frac{|S' \cap S|}{|S'|}$，召回率为 $\mathsf{recall}(S', S) = \frac{|S' \cap S|}{|S|}$。则 F-measure、strong F-measure、weak F-measure 定义如下：

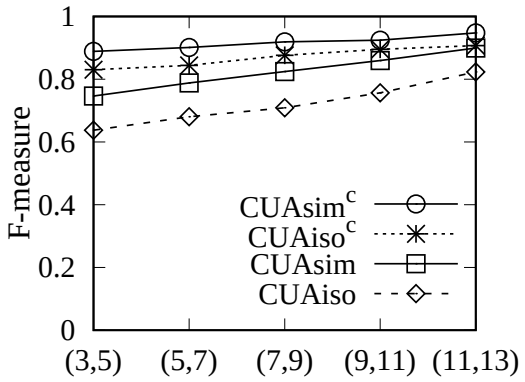
$$F(Q, Q', G) = \frac{2\mathsf{prec}(Q'(G), Q(G)) \cdot \mathsf{recall}(Q'(G), Q(G))}{\mathsf{prec}(Q'(G), Q(G)) + \mathsf{recall}(Q'(G), Q(G))} \tag{3.10}$$

$$F_s(Q, Q_u, Q_l, G) = \frac{2\mathsf{prec}(Q_u(G), Q(G)) \cdot \mathsf{recall}(Q_l(G), Q(G))}{\mathsf{prec}(Q_u(G), Q(G)) + \mathsf{recall}(Q_l(G), Q(G))} \tag{3.11}$$

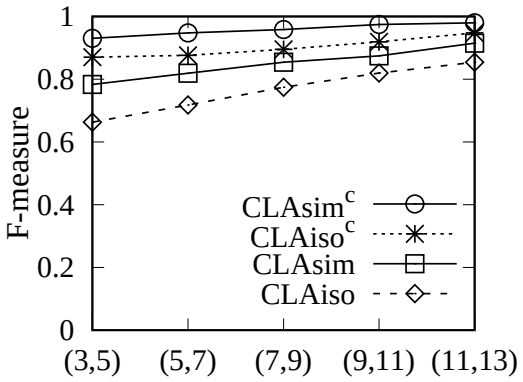
$$F_w(Q, Q_u, Q_l, G) = \frac{2\mathsf{prec}(Q_l(G), Q(G)) \cdot \mathsf{recall}(Q_u(G), Q(G))}{\mathsf{prec}(Q_l(G), Q(G)) + \mathsf{recall}(Q_u(G), Q(G))} \tag{3.12}$$

其中，$F(Q, Q', G)$ 是传统的 F-measure。$F_s(Q, Q_u, Q_l, G)$ 和 $F_w(Q, Q_u, Q_l, G)$ 是传统 F-measure 的扩展。$F_s(Q, Q_u, Q_l, G)$ 用 $Q_u(G)$ 衡量精确率 prec，用 $Q_l(G)$ 衡量召回率 recall；而 $F_w(Q, Q_u, Q_l, G)$ 用 $Q_l(G)$ 衡量精确率 prec，用 $Q_u(G)$ 衡量召回率 recall。直观理解，使用 Q_l 倾向于产生更高的 prec 值，使用 Q_u 倾向于产生更高的 recall 值。所以，weak F-measure 使用“最好”的 Q_u 和 Q_l 的组合来衡量准确性。与之相反，strong F-measure 使用“最坏”的 Q_u 和 Q_l 的组合来衡量准确性。存在特殊情况，当 Q_u 是 Q 的完全上界近似查询并且 Q_l 是 Q 的完全下界近似查询时，$\mathsf{prec}(Q_l(G), Q(G)) = \mathsf{recall}(Q_u(G), Q(G)) = F_w(Q, Q_u, Q_l, G) = 1$。

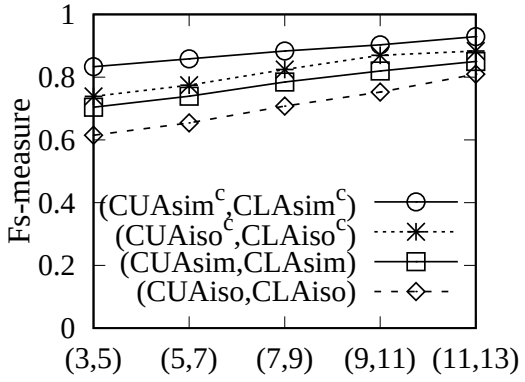
（*a*）Q 对准确性的影响。通过改变 Q 的大小 $|Q|$，使其从 (3, 5) 逐步变化至 (11, 13)，分别使用衡量标准 $F(Q, Q_u, G)$，$F(Q, Q_l, G)$，$F_w(Q, Q_u, Q_l, G)$ 和 $F_s(Q, Q_u, Q_l, G)$ 在数据图



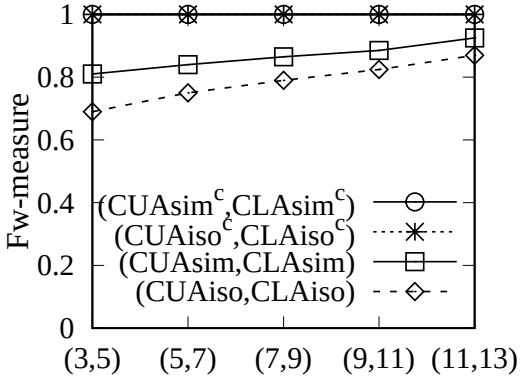
(a) 变化 $|Q|$（上界）



(b) 变化 $|Q|$（下界）



(c) 变化 $|Q|$（上界和下界）

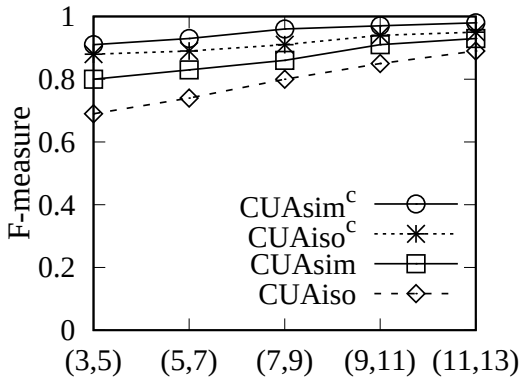


(d) 变化 $|Q|$（上界和下界）

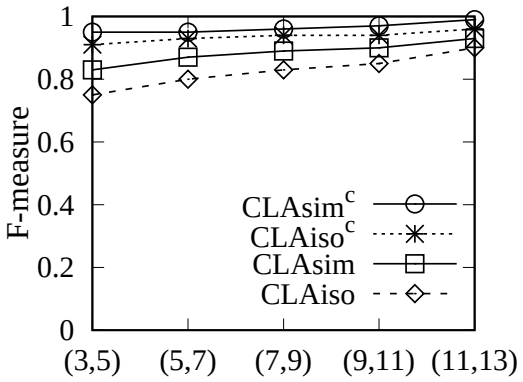
图 8　基于视图的最优近似查询方法有效性测试：变化 $|Q|$，DBpedia

DBpedia 和 YouTube 上评估模式图 Q 的大小变化对方法生成的 Q 的上界近似查询 Q_u 和下界近似查询 Q_l 的准确性的影响。数据图 DBpedia 上的实验结果如图 8a，8b，8c和 8d 所示，数据图 YouTube 上的实验结果如图 9a，9b，9c和 9d 所示。

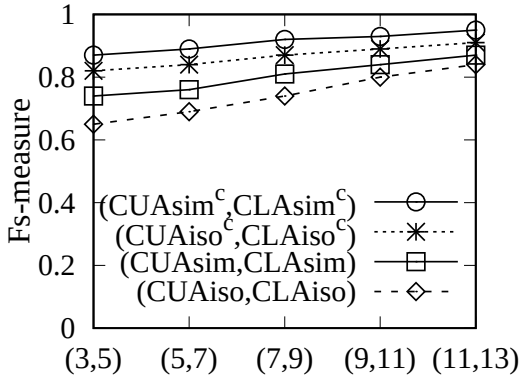
实验发现如下所示：（i）本文提出的基于视图的近似图匹配查询方法生成的 Q 的上界近似查询 Q_u 和下界近似查询 Q_l 可以有效的近似 Q 在数据图中的图匹配查询结果。例如在数据图 DBpedia 上，对于图模拟语义，Q_u 和 Q_l 的 F-measure 值在所有参数设置下均高于 0.7；对于子图同构语义，Q_u 和 Q_l 的 F-measure 值在所有参数设置下均高于 0.6。（ii）基于视图的完全近似查询比基于视图的近似查询准确率高。例如，在数据图 DBpedia 上对于子图同构语义，当 $|Q|$ 为 (11, 13) 时，完全上界近似查询的 F-measure 值为 0.89，大于上界近似查询的 F-measure 值 0.82。（iii）如准确性评价标准定义介绍中所述，用“最好”的 Q_u 和 Q_l 的组合可以得到更高的准确率（weak F-measure）。实验发现不论在数据图 DBpedia 上还是数据图 YouTube 上，不论对于图模拟语义还是子图同



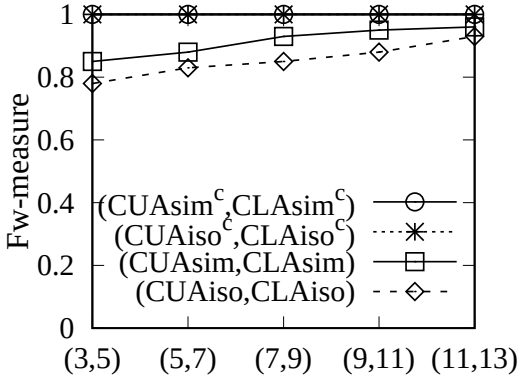
(a) 变化 $|Q|$（上界）



(b) 变化 $|Q|$（下界）



(c) 变化 $|Q|$（上界和下界）

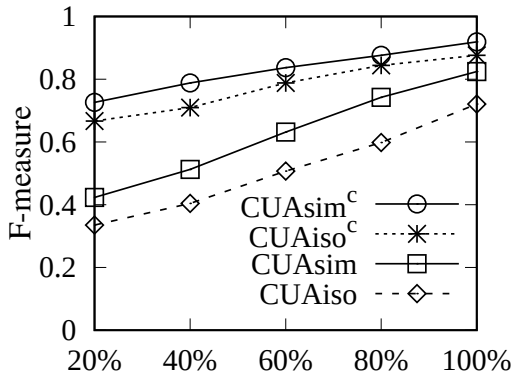


(d) 变化 $|Q|$（上界和下界）

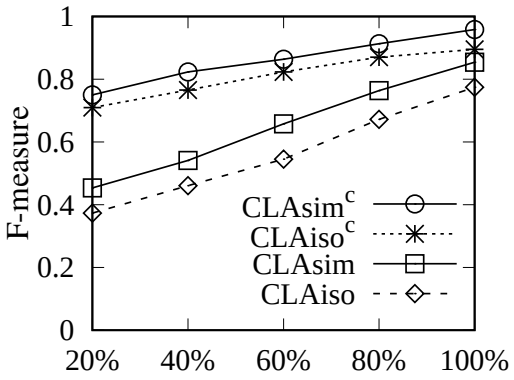
图 9　基于视图的最优近似查询方法有效性测试：变化 $|Q|$，YouTube

构语义，Q_u 和 Q_l 的 weak F-measure 值在所有参数设置下均高于单独使用 Q_u 或 Q_l 所得出的 F-measure 值。例如，在数据图 DBpedia 上对于图模拟语义，当 $|Q|$ 为 $(11, 13)$ 时，weak F-measure 值为 0.93，而单独使用 Q_u 或 Q_l 所得出的 F-measure 值均低于 0.9。（vi）实验发现尽管使用“最坏”的 Q_u 和 Q_l 的组合得出的准确率（strong F-measure）依然很高，说明上界近似查询 Q_u 和下界近似查询 Q_l 的确能够有效的近似 Q 在数据图中的图匹配查询结果。例如，在数据图 DBpedia 上对于图模拟语义，当 $|Q|$ 为 $(3, 5)$ 时，strong F-measure 值高于 0.7。（v）实验发现不论对于图模拟语义还是子图同构语义，基于视图的上界近似查询和下界近似查询的准确率不随 Q 的大小变化而发生显著变化。这条实验规律证明了该方法的通用性。

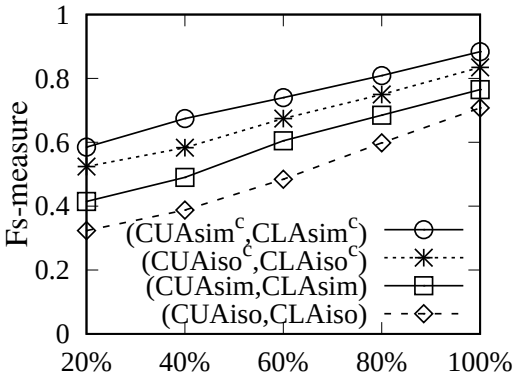
（*b*）$\mathcal{V}$ 对准确性的影响。通过改变视图的使用数量 $||\mathcal{V}||$，使其从生成视图总个数的 20% 逐步变化至 100%，并固定 Q 的大小 $|Q|$ 为 $(7, 9)$，分别使用衡量标准 $F(Q, Q_u, G)$，$F(Q, Q_l, G)$，$F_w(Q, Q_u, Q_l, G)$ 和 $F_s(Q, Q_u, Q_l, G)$ 在数据图 DBpedia 和 YouTube 上评估视



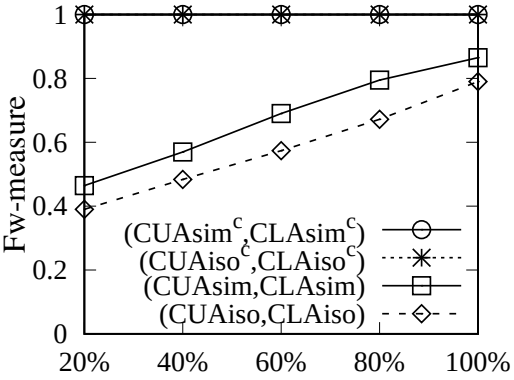
(a) 变化 $\|\mathcal{V}\|$（上界）



(b) 变化 $\|\mathcal{V}\|$（下界）



(c) 变化 $\|\mathcal{V}\|$（上界和下界）

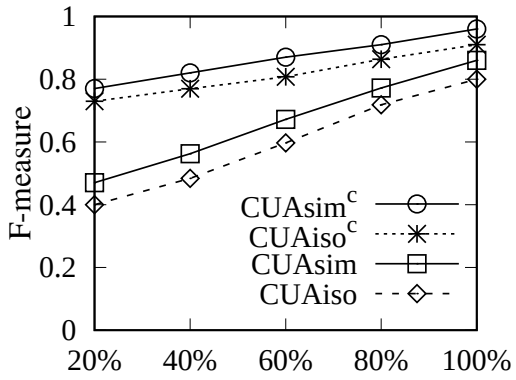


(d) 变化 $\|\mathcal{V}\|$（上界和下界）

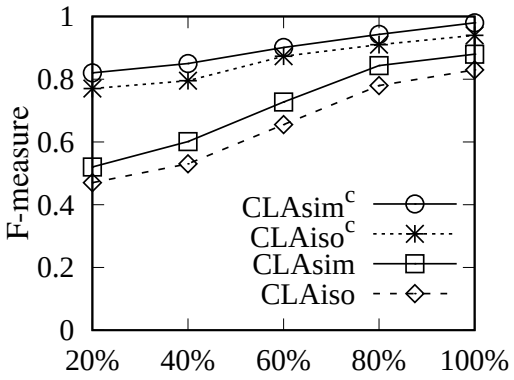
图 10　基于视图的最优近似查询方法有效性测试：变化 $\|\mathcal{V}\|$，DBpedia

图集合 $\mathcal{V}$ 的使用数量变化对方法生成的 Q 的上界近似查询 Q_u 和下界近似查询 Q_l 的准确性的影响。数据图 DBpedia 上的实验结果如图 10a，10b，10c和 10d 所示，数据图 YouTube 上的实验结果如图 11a，11b，11c和 11d 所示。

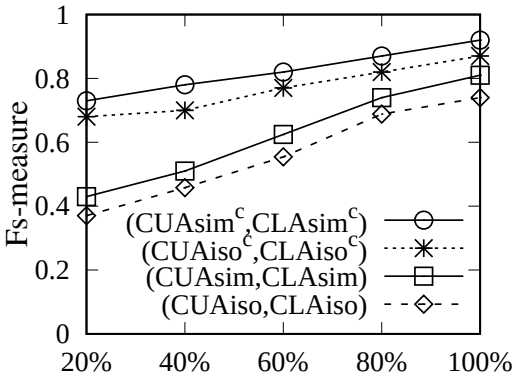
实验发现如下所示：（i）在所有情况下，当使用的视图数量 $\|\mathcal{V}\|$ 越多时，上界近似查询 Q_u 和下界近似查询 Q_l 的准确率越高。例如，在数据图 YouTube 上对于图模拟语义，当只有 20% 的视图被使用时，下界近似查询 Q_l 的 F-measure 值为 0.52；当 100% 的视图被使用时，Q_l 的 F-measure 值达到了 0.88。这是因为当更多视图被使用时，方法生成的基于视图的上界近似查询 Q_u 和下界近似查询 Q_l 与模式图 Q 的拓补结构越相近，则 Q_u 和 Q_l 的图匹配查询结果与 Q 在数据图中的图匹配查询结果越相近。（ii）实验发现由 Q_u 和 Q_l 组合得出的 weak F-measure 值高于分别单独使用 Q_u 和 Q_l 得出的 F-measure 值。这些值均高于由 Q_u 和 Q_l 组合得出的 strong F-measure 值。这条实验规律与变化参数 $|Q|$ 实验中发现的实验规律一致。



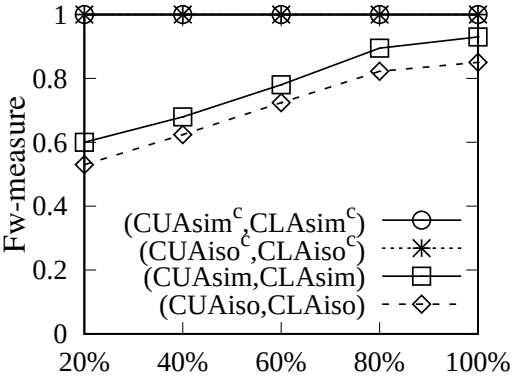
(a) 变化 $\|\mathcal{V}\|$（上界）



(b) 变化 $\|\mathcal{V}\|$（下界）



(c) 变化 $\|\mathcal{V}\|$（上界和下界）

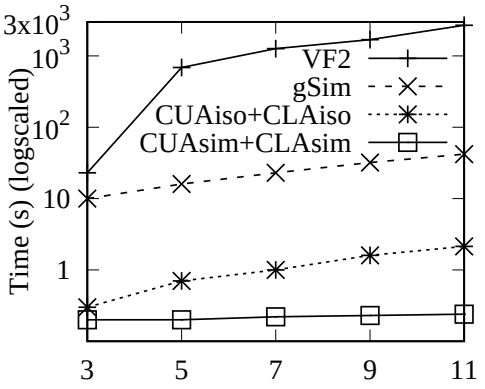


(d) 变化 $\|\mathcal{V}\|$（上界和下界）

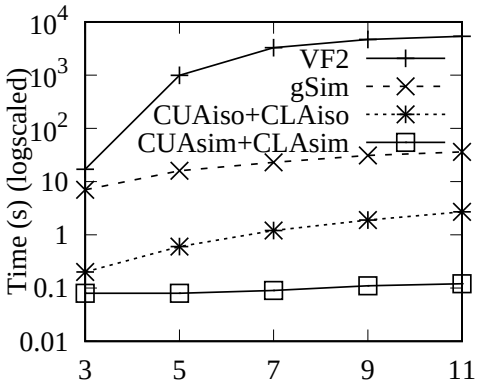
图 11　基于视图的最优近似查询方法有效性测试：变化 $\|\mathcal{V}\|$，YouTube

（3）基于视图的最优近似图匹配查询的加速比测试。本章设计实验测量了采用基于视图的近似图匹配查询方法计算图匹配查询结果相对于直接使用图匹配查询算法计算图匹配查询结果的加速比。实验测量和比较了利用视图集合 $\mathcal{V}$ 生成 Q 的上界近似查询 Q_u 和下界近似查询 Q_l，以及计算 Q_u 和 Q_l 的图匹配查询结果（算法 QAVsim 和 QAViso），和直接访问数据图计算模式图 Q 的图匹配查询结果（算法 gSim 和 VF2）这两个过程的计算时间。为了减少噪音，实验中只统计准确率高于 0.6 的上界和下界近似查询的平均计算处理时间。数据图 DBpedia 上的实验结果如图 12a 和 12c 所示，数据图 YouTube 上的实验结果如图 12b 和 12d 所示。

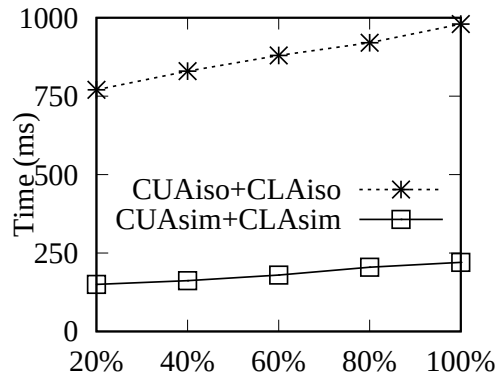
实验发现如下所示：利用视图计算上界近似查询和下界近似查询的图匹配查询结果的效率远高于直接在数据图中计算模式图的图匹配查询结果的效率。比如在数据图 YouTube 上，当使用 100% 的视图时，对于子图同构语义，生成上界近似查询 Q_u 和下界近似查询 Q_l 以及计算 Q_u 和 Q_l 的图匹配查询结果的总时间，比算法 VF2 的执行时间快



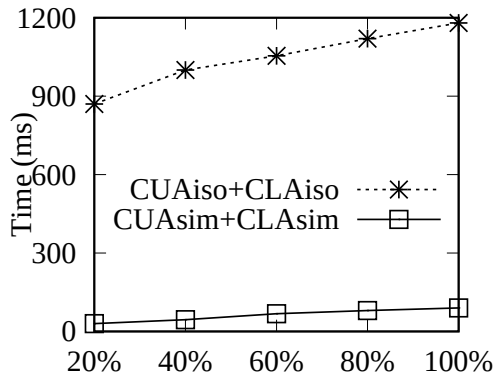
(a) DBpedia： 变化 $|Q|$



(b) YouTube： 变化 $|Q|$



(c) DBpedia： 变化 $\|\mathcal{V}\|$



(d) YouTube： 变化 $\|\mathcal{V}\|$

图 12　基于视图的最优近似查询方法加速比测试

90 到 2000 倍；对于图模拟语义，生成上界近似查询 Q_u 和下界近似查询 Q_l 以及计算 Q_u 和 Q_l 的图匹配查询结果的总时间，比算法 gSim 的执行时间快 80 到 300 倍。在数据图 DBpedia 上可以观测到同样的实验规律。

3.7.3　效率测试

本章测试基于视图的近似图匹配查询方法中所设计的所有算法的效率。实验发现当 100% 的视图被使用时，在两个数据图上对于所有模式图，所有算法均能在 2.7s 内执行完毕。

3.7.4　实验总结

实验结果表明，（1）本文提出的基于视图的近似图匹配查询方法生成的上界近似查询和下界近似查询可以有效的近似模式图在数据图中的查询结果。当较少数量的视图被

使用时，约 65% 的模式图存在基于图模拟的近似查询，约 53% 的模式图存在基于子图同构的近似查询。（2）本文提出的方法对图匹配查询过程的效率提升有显著效果。当处理百万级真实数据时，对于图模拟语义，方法生成的上界近似查询和下界近似查询的准确率（F_w）在 0.79 以上，并且可以在 0.24s 内执行完毕整个查询过程，而直接运行基于图模拟的图匹配查询算法时间花费平均为 42s；对于子图同构语义，方法生成的上界近似查询和下界近似查询的准确率（F_w）在 0.86 以上，并且可以在 2.7s 内执行完毕整个查询过程，而直接运行基于子图同构的图匹配查询算法时间花费平均为 5382s。（3）本文设计的算法效率很高：所有算法在所有实验参数设置下均能在 2.7s 内运行完毕。

3.8　相关工作分析

本课题的相关工作有两方面：基于视图的图匹配查询和查询近似。

基于视图的图匹配查询。与本课题最相关的工作是由 [46, 47] 提出的基于视图的图匹配查询方法。本文在章节 2.3.1 中详细介绍了这项工作。本课题与这项工作有以下几处不同：（1）本课题将基于视图的精确图匹配查询扩展为近似图匹配查询。当利用视图无法得到模式图的精确图匹配查询结果时，本课题提出的方法可以利用视图找到模式图的近似图匹配查询结果。（2）本课题对子图同构语义下基于视图的图匹配查询问题进行了研究并设计了算法，这一问题在本课题之前未被相关工作提出和研究过。

查询近似。与本课题相关的另一个工作是查询近似方法。这项工作包括以下三个方向研究内容：（a）数据驱动的查询近似。该方法首先为数据集构建数据摘要，而后在数据摘要上为输入查询执行查询过程 [77, 78]。（b）查询驱动的查询近似。该方法将计算花费较大的输入查询转换为计算花费较小的查询, 使得转换后的查询在数据集中的查询结果是原始查询在数据集中查询结果的近似解 [79–81]。（c）启发近似。该方法通过启发式规则对查询搜索空间进行剪枝从而减少计算花费，最终得到精确度在可接受范围内的非准确查询结果 [17, 82, 83]。

本课题与这项工作有以下几处不同：（1）本课题提出了基于视图的近似图匹配查询方法，该方法可以适用于任意一种图匹配查询语义。然而，现有工作主要关注于关系查询的查询近似 [77, 78, 80, 81]。（2）与数据驱动的查询近似和启发近似不同 [17, 77, 78, 82, 83]，本课题关注于查询驱动的查询近似，不依赖于数据集的存在。（3）本课题的研究内容是基于视图的查询近似，这个思路在之前查询驱动的研究工作中未被涉及 [80, 81]。

3.9 小结

本文提出了基于视图的近似图匹配查询方法，该方法适用于任意一种图匹配查询语义。本文利用该方法为模式图生成基于视图的上界近似查询和下界近似查询，并分析了基于视图的上界和下界近似查询的性质。本文对该方法中的八个基本问题，即基于视图的（完全）上界/下界近似查询的存在性/最优化问题，从计算复杂性和可近似性两个方面进行了理论分析。根据理论分析结论，本文设计了高效的精确算法、具有近似度保证的近似算法和实用的启发算法计算基于视图的最优（完全）上界和下界近似查询。本文在论述该方法工作原理时以图模拟为图匹配查询语义，并将该方法扩展至子图同构图匹配查询语义以验证方法的通用性。基于上述研究，本文抽象出基于视图的近似图匹配查询方法的通用计算框架。本文的实验结果验证了基于视图的近似图匹配查询方法的有效性和高效性。总而言之，本文提出的方法将基于视图的精确图匹配查询扩展为近似图匹配查询，提高了视图的命中率和利用率，从而提高图匹配查询引擎的效率。

第四章　图匹配查询增量计算框架

在实际应用中，用于查询出最优 k 个结果的 top-k 查询方法一般更具实用性。例如在 Web 搜索中，搜索引擎会查询出与用户输入的关键词最匹配的 k 个（如 Google，k=10）网页并将其按顺序排列于搜索结果的第一页。本文首先提出 top-k 图匹配查询语义，用于从数据图中查找出与模式图匹配的最优 k 个匹配子图。相对于传统图匹配查询语义，该图匹配查询语义更具实用价值，适用于社交推荐和团队搜索等需要找到最优 k 个满足实际需求的查询结果的应用。另外，图匹配查询通常是在高度动态的环境下执行的。因为实际应用中的数据集规模通常较大，并且随着时间的推移不断演变。而且，对于所有类型查询引擎和所有类型数据集，迭代设计查询是一个常见的过程，即用户根据上一轮查询结果满意程度多次反复调整查询输入，以期得到预期的查询结果。由于动态环境对高效查询引擎的要求日益增长，本文提出并研究动态图匹配查询问题，考虑连续混合的模式图更新和数据图更新。针对动态图匹配查询问题，本文提出了统一的图匹配查询增量计算框架，该框架可以处理连续的单独或混合的模式图和数据图更新。这是对模式图更新的第一次研究，也是迄今为止所考虑的最广泛和最实用的动态设置。本文提出的图匹配查询增量计算框架是通用方法，可以适用于 top-k 图匹配查询方法和一般图匹配查询方法以及任意一种图匹配查询语义。基于图匹配查询的增量计算框架，本文设计了高效的增量算法和查询优化技术。最后，在真实数据集论文引用网络图 Citation 和视频网络图 YouTube 和合成数据图 Synthetic 上，本文通过实验分析证明了 top-k 图匹配查询语义的实用价值，并且证明了图匹配查询增量计算框架及相关增量算法能够显著提高图匹配查询的效率。

4.1　引言

在实际应用中，用于查询出最优 k 个结果的 top-k 查询语义一般更具实用性。例如在 Web 搜索中，当用户输入查询关键词，Web 搜索引擎能够从互联网上查找到千万个甚至数亿个与关键词匹配的 Web 网页。然而，将搜索结果全部返回给用户显然不可行。为了使返回给用户的查询结果具有可读性，主流 Web 搜索引擎无一例外的采用 top-k 查询方法 [84, 85]。例如，Google 搜索引擎查询出与用户输入的关键词最匹配的前 10 个网页并将其按顺序排列于搜索结果的第一页。而后，再将第 11-20 个匹配网页按顺序排列于搜索结果的第二页，以此类推。经过研究发现，用户的行为模式倾向于只查看第一页

的搜索结果 [86]，并且统计结果显示 Google 用户对于搜索结果第一页的 10 个网页的点击量占所有搜索结果网页点击量的 91.7% [87]。这些事实说明查询引擎能够返回实用结果的重要性。相对于千万甚至数亿个查询结果，用户往往只关注并且只能够访问最优的 k 查询结果。然而，本文通过对图匹配查询各种匹配语义分析得知（章节 2.1），图匹配查询的查询结果实用价值较低。基于子图同构的图匹配查询，其查询结果是数据图中与模式图完全相同的子图，然而这些匹配子图的数目可以达到指数多个；基于图模拟的图匹配查询，其查询结果是数据图中一个与模式图匹配的子图，然而该匹配子图的规模可以达到与数据图相同的规模；基于强模拟的图匹配查询，其查询结果是数据图中 $|V|$ 个与模式图匹配的子图。其中，$|V|$ 为数据图的节点个数，匹配子图的规模与模式图的规模相近。显然，在三种典型的图匹配查询语义中，强模拟语义的图匹配查询结果最具实用性。然而，本文发现强模拟查询语义的实用价值仍然较低。由于现实生活中数据图规模通常较大，社交网络图等应用中节点规模已经达到千万级 [2]，这导致强模拟语义返回的结果依然较多。而且，强模拟查询语义不能表达一些实际应用的需求，比如查找到指定匹配节点个数和指定匹配子图规模的查询结果。

另外，同 Web 搜索引擎 [88] 和其他结构化和非结构化查询引擎（参见综述 [48]）一样，图匹配查询面临一个关键问题：图匹配查询通常是在高度动态的环境下执行的。实际应用中的数据集规模通常较大，并且随着时间的推移不断演变。例如在社交网络（如 Facebook）等新应用中，2017 年月平均活跃用户达到了 21.2 亿，相比 2016 年增长了 13.9% [2]。另外，数据处于高度变化状态，据统计 Facebook 每天新增用户超过 50 万，每分钟产生 51 万新评论，29.3 万用户状态更新, 每天新增数据量达到 4PB [3, 4]。而且，对于所有类型查询引擎和所有类型数据集，迭代设计查询是一个常见的过程，即用户根据上一轮查询结果满意程度多次反复调整查询输入，以期得到预期的查询结果 [32, 33]。然而，本文通过对图匹配查询的各种匹配语义分析得知（章节 2.1），图匹配查询语义的计算复杂度普遍较高。基于子图同构语义的图匹配查询，该查询语义计算复杂度最高，因为子图同构问题为 NP-完全问题；基于图模拟语义的图匹配查询，该查询语义计算复杂度最低，为二次方多项式时间；基于强模拟语义的图匹配查询，该查询语义为三次方多项式时间。这些事实说明即使采用复杂度最低的图匹配查询语义（图模拟语义），每当数据图和模式图发生更新，重新执行图匹配查询计算过程这种更新处理方法计算开销太大，是绝对不可行的。我们需要提出新的动态环境下的图匹配查询模式来提高查询效率，而该模式应该是查询语义无关的，即其可以使现有的所有图匹配查询语义受益。本

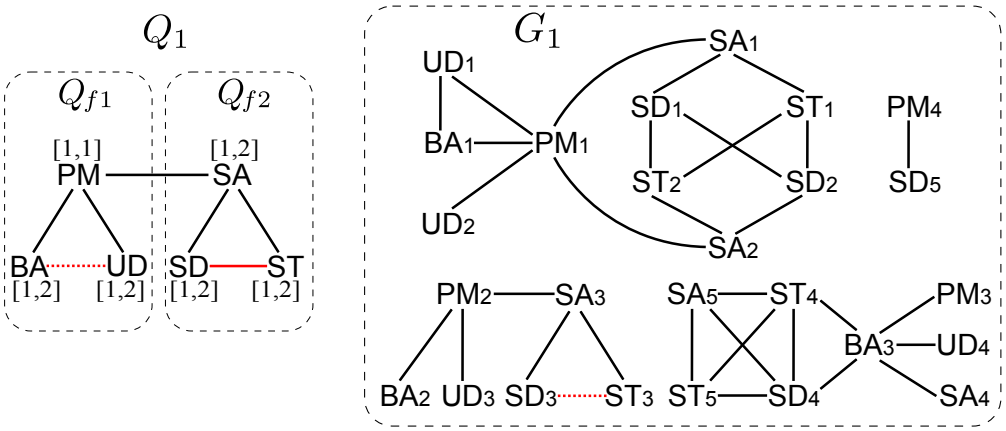


图 13　查询动态推荐网络图

文将通过实际应用中的两个例子阐述前文中提出的所有问题。

例 4.1: 以一个推荐网络图 G_1 为例 [20]，如图 13所示。G_1 中每个节点表示一个具有专业知识的技术人员，节点上的标签表示不同的专业，比如项目经理（PM ）、软件架构设计师（SA ）、软件开发工程师（SD ）、软件测试工程师（ST ）、界面设计工程师（UD ）以及业务分析师（BA ）；G 中每条边表示两个技术人员之间存在过良好的合作关系，比如边 $(\mathrm{PM}_1, \mathrm{SA}_1)$ 表示PM_1 曾经和SA_1 高效的合作并完成过项目开发。

一个人力资源部经理想要为他所在的软件公司组建两个开发团队致力于即将开展的企业软件开发项目。为了组建能够胜任软件开发业务需求并且可以良好合作的开发团队，他将团队组建需求用一个模式图 Q_1（不包括虚线边）来表示，并借助于推荐网络图 G_1（不包括虚线边）来查询出满足需求的团队，Q_1 如图 13 所示（虚线方格内）。他想要找到的目标团队需要（1）包括一个PM ，还包括BA 、UD 、SA 、SD 和 ST 各一个到两个，并且（2）PM 能够和 SAs 、BAs 和 UDs 都有高效的合作关系，另外SDs 和STs 之间可以高效合作并且分别能够与SAs 高效合作。 □

例 4.1 中所述问题实际为团队构建应用领域的一个问题描述。相关研究为团队构建问题提出了很多解决方法 [89–92]。但是这些解决方法只能表达出例 4.1 中所述的团队职能需求和团队成员个数需求（查询条件（1）），但是无法表达出团队结构需求（查询条件（2））。而满足团队结构需求则是团队可以高效合作的关键和保证。由于图匹配查询方法可以保证模式图和查询结果的结构匹配关系，从而可以为团队构建应用提供良好的解决方案。然而，通过分析图匹配查询的匹配语义可以发现，现有图匹配查询语义无法准确表达出比如找到最优的一个匹配结果和限制匹配结果中节点个数等实际应用需求。

本文将提出一种基于强模拟语义的 top-k 图匹配查询语义，用于从数据图中查找出与模式图匹配的最优 k 个匹配子图。另外，本文为该查询语义引入能够控制匹配结果规模的参数，包括匹配节点个数和匹配子图结构半径参数，从而适用于社交推荐和团队搜索等需要找到最优 k 个满足实际需求的查询结果的应用。

图匹配查询通常是在高度动态的环境下执行的，如例 4.2 所示。

例 4.2: 继续以例 4.1 中模式图 Q_1 和数据图 G_1 为例，如图 13 所示。

（1）分析可知，由于 Q_1 查询约束条件较严格在 G_1 中只能找到一个查询结果。为了组建两个团队，人力资源部经理继续修改模式图 Q_1，通过减少一条边 $\Delta Q_1 = (\mathsf{SD}, \mathsf{ST})^-$ 期望找到更多查询结果。

（2）另外，数据图通常会随着时间的推移不断演变。例如，数据图 G_1 在某时刻增加了一条边 $\Delta G_1 = (\mathsf{SD}_3, \mathsf{ST}_3)^+$。

（3）模式图和数据图很可能会同时发生更新。例如，模式图 Q_1 的更新 ΔQ_1 和数据图 G_1 的更新 ΔG_1 同时发生。 □

由于动态环境对高效查询引擎的要求日益增长，本文将提出并研究动态图匹配查询问题，考虑连续混合的模式图更新和数据图更新。增量计算技术即在数据动态更新情况下通过复用已有查询结果计算数据更新后结果。其目标是通过最大化减少冗余计算从而提高查询效率技术。本文将提出一个统一的图匹配查询增量计算框架，该框架可以处理连续的单独或混合的模式图和数据图更新。并且，本文提出的增量计算框架是通用方法，可以适用于 top-k 和一般图匹配查询方法以及任意一种图匹配查询语义。基于图匹配查询的增量计算框架，本文设计了高效的增量算法和查询优化技术。这是对模式图更新的第一次研究，也是迄今为止所考虑的最广泛和最实用的动态设置。

本章主要内容提纲。基于对以上两个核心问题的研究，本章节研究内容提纲如下所示。

（1） 本文提出基于组合模拟的 top-k 图匹配查询语义，用于从数据图中查找出与模式图匹配的最优 k 个匹配子图。另外，本文为该查询语义引入能够控制匹配结果规模的参数，包括匹配节点个数和匹配子图结构半径参数，从而适用于社交推荐和团队搜索等需要找到最优 k 个满足实际需求的查询结果的应用。（章节 4.2）。

（2） 本文为 top-k 图匹配查询语义设计了批处理算法。本文研究了模式图的可满足性，这是图匹配查询语义加入规模控制参数后出现的一致性验证问题。本文还提出了两个优化技术，快速计算半径变化球内匹配结果和基于密度的过滤技术，以提高计算效率（章节 4.3）。

（3） 为了应对高度动态环境对图匹配查询的要求，本文提出动态图匹配查询问题，考虑连续混合的模式图更新和数据图更新。这是对模式图更新的第一次研究，也是对模式图和数据图连续混合更新的首次研究。本文研究的图匹配查询的动态环境是迄今为止所考虑的最广泛和最实用的动态设置（章节 4.4）。

（4） 本文首先对动态图匹配查询问题的计算复杂性进行了理论分析。尽管由理论分析得知问题的计算复杂度较高，本文设计了基于模式图划分和标识影响区域的增量计算策略，从而将模式图和数据图的更新影响范围最小化和局域化。基于增量计算策略，本文提出统一的图匹配查询增量计算框架，可以处理连续的单独或混合的模式图和数据图的连续更新。该计算框架是通用计算方法，适用于 top-k 图匹配查询和一般图匹配查询方法以及任意一种图匹配查询语义。（章节 4.5）。

（5） 基于图匹配查询的增量计算框架，本文设计了统一的增量算法来处理单独或混合的模式图和数据图更新。本文在基于传统的针对 top-k 批处理算法的提前终止优化技术基础上，提出了针对 top-k 增量算法的提前返回优化技术（章节 4.6）。

（6） 本文采用论文引用网络图 Citation 和视频网络图 YouTube 两个真实数据集和合成数据图 Synthetic 来评估方法性能，数据集包含百万级节点数据和千万级边数据。经实验验证得知（a）top-k 图匹配查询语义能够在实际应用中查找到有意义的查询结果；并且（b）增量算法效率相对批处理算法效率有显著提升：即使当模式图更新百分比达到 36%，数据图更新百分比达到 30.2%，模式图和数据图混合更新百分比达到（25.5%, 20%），连续模式图更新百分比达到 28.5%，连续数据图更新百分比达到 20.7%，连续混合模式图和数据图更新百分比达到（18%, 14.5%），增量算法相对批处理算法查询效率都有提升，验证了方法的有效性（章节 4.7）。

为了论述简洁，本文只将部分定理证明列于正文中，其他定理证明详见附录 B。

4.2 Top-k 图匹配查询

本章首先提出基于组合模拟的图匹配查询语义。而后，本章提出基于组合模拟的 top-k 图匹配查询语义，用于从数据图中查找出与模式图匹配的最优 k 个基于组合模拟语义匹配的子图。

4.2.1 基于组合模拟的图匹配查询

本文在章节 2.1对模式图、数据图、图匹配查询进行了详细介绍。本章首先对这些基本概念进行回顾，并且扩展传统的模式图为其增加匹配节点个数要求。而后，在无向

图上定义组合模拟图匹配查询语义。

定义 4.1 (数据图): 数据图 $G(V_G, E_G, l_G)$ 是由节点集和边集构成的无向图。其中，V_G 为节点集，E_G 为边集，l_G 为标签函数将 V_G 中任一节点映射至标签集合 Σ 中的一组标签。□

定义 4.2 (模式图): 模式图 $Q(V_Q, E_Q, l_Q, f_Q)$ 是由节点集和边集构成的无向图。其中，（1）V_Q 为节点集，E_Q 为边集；（2）l_Q 为标签函数将 V_Q 中任意一个节点映射至节点标签集合 Σ 中的一个标签；并且（3）f_Q 为容量函数将 V_Q 中任意一个节点 $u \in V_Q$ 映射至一个闭区间 $[x, y]$，其中 $x \leq y$ 并且 x 和 y 均为非负整数。 □

直观理解，$f_Q(u)$ 为模式图中的节点 u 指定了一个整数区间，限定了数据图中与 u 匹配的节点个数的范围。对于传统的图匹配查询，一般在模式图的边上存在限定匹配结果容量的标识 [27, 93, 94]，而非节点上。在上下文已知的情况下，本文将数据图 $G(V_G, E_G, l_G)$ 简写为 $G(V_G, E_G)$，将模式图 $Q(V_Q, E_Q, l_Q, f_Q)$ 简写为 $Q(V_Q, E_Q)$。

图模拟是定义于有向图上的图匹配查询语义 [13]，本文在章节 2.1.2 详细介绍了图模拟匹配查询语义的定义。本章将在有向图上的图模拟匹配语义的基础上定义无向图上的图模拟匹配查询语义。

定义 4.3 (图模拟 (无向图)): 给定模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$，Q 与 G 通过图模拟匹配，记为 $Q \prec G$，当且仅当存在节点集合之间的二元关系 $R \subseteq V_Q \times V_G$，使得
（1）对于每对节点 $(u, v) \in R$, 节点 u 和节点 v 有相同的标签，即 $l_Q(u) = l_G(v)$；并且
（2）对于 Q 中每个节点 u，在 G 中都存在一个节点 v，使得（a）$(u, v) \in R$，并且（b）对于 Q 中节点 u 的每个邻接节点 u'，在 G 中都存在 v 的邻接节点 v'，使得 $(u', v') \in R$。

其中，G 中会存在多个与 Q 匹配的二元关系，但只存在一个最大二元匹配关系 R_M，使得对于 G 中所有与 Q 匹配的二元关系 P，P 都是 R_M 的子集，即 $P \subseteq R_M$。 □

下面介绍球结构和匹配图的定义。

定义 4.4 (球结构): 对于数据图 G 中一个节点 v，和一个非负整数 r，以 v 为球心以 r 为半径的球 $\hat{G}[v, r]$ 是 G 的一个子图，使得
（1）对于球 $\hat{G}[v, r]$ 中的所有节点 v'，v' 和 v 之间的最短距离小于等于 r，即 $\mathsf{dist}(v, v') \leq r$。
（2）球 $\hat{G}[v, r]$ 含有 G 中所有相同节点的所有邻接边。 □

定义 4.5 (匹配图): 对于图模拟匹配语义，如果 $Q \prec G$，则存在最大二元匹配关系 $R_M \subseteq V_Q \times V_G$。$Q$ 在 G 中的匹配图 G_s 为 G 中的一个子图。其中，

（1）一个节点 $v \in V_s$ 当且仅当节点 v 属于最大二元匹配关系 R_M，并且

（2）G_s 含有 G 中所有相同节点的所有邻接边。 □

直观理解，基于最大二元匹配关系 R_M 的匹配图 G_s 是数据图 G 的以 R_M 中所有节点为节点集的诱导子图。

下面将介绍组合模拟图匹配查询语义。该语义在图模拟匹配语义的基础上通过扩展传统的模式图增加了匹配节点个数要求，通过引入球结构增加了结果的局域性要求。

定义 4.6 (组合模拟): 给定模式图 $Q(V_Q, E_Q)$、数据图 $G(V_G, E_G)$ 和正整数半径 r，则 Q 与 G 通过组合模拟语义匹配，记为 $Q \lhd_r G$，当且仅当 G 中存在一个球结构 $\hat{G}[v,t]$（$t \in [1,r]$, $t \in Z$），使得

（1）$Q \prec \hat{G}[v,t]$, 即 Q 与 $\hat{G}[v,t]$ 通过图模拟匹配。其中，最大二元匹配关系为 R_M，基于 R_M 的匹配图为 G_s；并且

（2）对于 Q 中任意一个节点 u，G_s 中存在节点 v 与其匹配，即 $(u,v) \in R_M$。这些匹配节点的总个数满足区间 $f_Q(u)$ 限制。

其中，G_s 被称为模式图 Q 在 G 中的完美匹配子图。 □

直观理解，（1）模式图 Q 可以表达实际应用中对于匹配结果的结构限制和容量限制条件，并且（2）模式图 Q 在数据图 G 中的完美匹配子图 G_s 满足以下限制条件：（a）G_s 与 Q 基于图模拟匹配，并且 G_s 是 Q 的位于一个球结构 $\hat{G}[v,t]$（$t \in [1,r]$）内的匹配图；另外，（b）G_s 满足模式图 Q 中的所有节点容量限制要求。

例 4.3: 给定例 4.1 中模式图 Q_1 和数据图 G_1，如图 13 所示，给定半径 $r = 2$。分析可知 Q_1 和 G_1 基于组合模拟语义匹配，也就是 $Q_1 \lhd_r G_1$，因为（a）G_1 中存在一个球结构 $\hat{G}[\mathrm{PM}_1, 2]$，其中存在一个 Q_1 的完美匹配子图，即 G_1 中包含节点 PM_1 的连通分量，将模式图 Q_1 中的节点 PM 、BA 、UD 、SA 、SD 和 ST 分别匹配至球 $\hat{G}[\mathrm{PM}_1, 2]$ 中的节点 $\{\mathrm{PM}_1\}$、$\{\mathrm{BA}_1\}$、$\{\mathrm{UD}_1, \mathrm{UD}_2\}$、$\{\mathrm{SA}_1, \mathrm{SA}_2\}$、$\{\mathrm{SD}_1, \mathrm{SD}_2\}$ 和 $\{\mathrm{ST}_1, \mathrm{ST}_2\}$。并且（b）球 $\hat{G}[\mathrm{PM}_1, 2]$ 中的匹配节点满足 Q_1 中所有节点容量限制要求。 □

备注。（1）组合模拟匹配语义与图模拟 [13] 和强模拟 [14] 匹配语义不同之处在于：组合模拟能够表达匹配节点个数限制要求，并且组合模拟是无向图上的图匹配查询语义。（2）与强模拟匹配语义中球结构采用固定的半径（模式图的直径）不同，组合模拟匹配语义在用户指定的最大球结构半径内的所有球中查找完美匹配子图，从而可以控制匹配结果的规模，增强组合模拟图匹配查询语义的实用性。

4.2.2　基于组合模拟的 top-k 图匹配查询

给定模式图 Q，数据图 G 和两个正整数 r 和 k，基于组合模拟语义的 top-k 图匹配查询，记为 kGPM(Q,G,k)，即在 G 中找到 k 个密度最大的 Q 的完美匹配子图，并根据密度大小将其按顺序排列成一个完美匹配子图序列 L_k。

这里图 $H(V_H,E_H)$ 的密度 den_H 为 $|E_H|/|V_H|$。其中，$|E_H|$ 为图 H 中边的个数，$|V_H|$ 为图 H 中节点的个数。这种密度计算方法是数据挖掘相关应用中的常用计算方法 [95, 96]。直观理解，den_H 越大，说明图 H 的节点之间的边更多，结构更具多样性。在社交推荐和团队搜索等实际应用中，组合模拟匹配语义利用球结构将匹配节点之间的距离限制在 $2*r$ 内，并且利用模式图的节点容量要求限制匹配结果的节点数目；Top-k 查询语义利用密度计算方法查找出匹配节点之间联系更紧密的匹配结果，从而使得基于组合模拟的 top-k 图匹配查询方法更具实用性。

例 4.4: 给定例 4.1 中模式图 Q_1 和数据图 G_1，如图 13 所示，给定半径 $r=2$。设定 $k=1$，因为绝大多数团队构建问题的查询方法只能够找到最优的查询结果 [89–92]。利用现有团队构建问题的查询方法在 G_1 中查找满足例 4.1 中所述需求的项目团队。结果如下所示：

（1）通过最小化查询结果的直径 [89]，查询得到由以下节点集合构成的团队：{BA_3, PM_3, UD_4, SA_4, SD_4, ST_4}；

（2）通过最小化查询结果中所有节点对之间的距离之和 [92] 查询得到和（1）相同团队。

（3）通过最大化查询结果的密度 [90]，查询得到以 G_1 中包含节点 PM_1 的连通分量和包含节点 BA_3 的连通分量中所有节点（节点 UD_2、PM_3、UD_4 和 SA_4 除外）构成的团队。

可以看出，这些方法查询得到的团队只能够满足例 4.1 中所述的职能需求（查询条件（1）），然而不能满足团队结构需求。分析可知，方法（1）和方法（2）查找到的团队只由节点 BA_3 连接，方法（3）查找到的团队成员之间合作关系也很松散。

当利用基于组合模拟的 top-k 图匹配查询方法时，可以查找到例 4.3 中所述团队，其密度为 1.4。该团队满足例4.1 中的所有查询条件。该方法的查询结果质量优于以上所有团队查找到的结果。　□

4.3　Top-k 图匹配查询算法

本章为 top-k 图匹配查询语义设计了批处理算法。本章首先研究了模式图的可满足性问题，而后提出了两个优化技术。本章在最后介绍了批处理算法的具体工作流程。

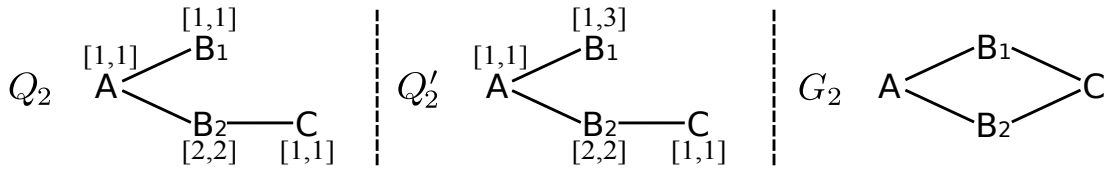


图 14　模式图可满足性示例

4.3.1　模式图可满足性

与图模拟 [13] 和图模拟扩展匹配语义 [14, 28] 不同，组合模拟匹配语义的节点容量控制参数会导致出现模式图不可能与任何数据图匹配的情况，如例 4.5 所示。

例 4.5:（1）对于图 14 中的模式图 Q_2，通过验证可知不存在任何数据图 G 与 Q_2 通过组合模拟语义匹配，即 $Q_2 \lhd_r G$，因为（a）对于 G 中任意节点 v，如果 v 与 Q_2 中节点 B_2 匹配，那么 v 必定与 Q_2 中节点 B_1 匹配，所以（b）B_1 的匹配节点容量区间的上界不应该小于 B_2 的匹配节点容量区间的下界。

（2）相反的，图 14 中的模式图 Q_2' 是可满足的，因为存在数据图 G_2 使得 $Q_2 \lhd_r G_2$。另外，图 13 中的模式图 Q_1 也是可满足的，因为存在数据图 G_1 使得 $Q_1 \lhd_r G_1$。 □

本文称模式图 Q 是可满足的当且仅当存在一个数据图 G 与 Q 通过组合模拟语义匹配，即 $Q \lhd_r G$。判定一个模式图的可满足性这一过程可以在较快的多项式时间内完成。

定理 4.1: 模式图 Q 的可满足性判定可以在 $O(|Q|^2)$ 时间内计算完毕。 □

将模式图 Q 既看做模式图也看做数据图，可以为 Q 在 Q 自身中计算基于图模拟的最大二元匹配关系 R_M。模式图 Q 是可满足的当且仅当对于任意 $(u, v) \in R_M$，其中 u 上的容量区间为 $[x_u, y_u]$，v 上的容量区间为 $[x_v, y_v]$，$x_v \leq y_u$ 成立。分析可知 R_M 的规模最大为 $|Q|^2$，并且通常模式图都很小。所以模式图的可满足性判定计算开销很小。

由定理 4.1提供正确性保证，本文在后续章节中只考虑可满足的模式图。

4.3.2　批处理算法

本章首先为批处理算法设计两个优化技术：快速计算半径变化球内匹配结果和基于密度的过滤技术。而后，本章将介绍基于组合模拟的 top-k 图匹配查询方法的批处理算法。

半径变化球内匹配结果。基于组合模拟的 top-k 图匹配查询是从数据图 G 的所有球结构 $\hat{G}[v, t]$（$v \in V$，$t \in [1, r]$）内查找出密度最大的 k 个匹配结果。然而，构建 $r|V|$ 个球结构并依次在其中计算图匹配查询结果这一过程计算开销太大。经分析得知，上述过程包含

很多冗余计算。只需要构建 $|V|$ 个半径为 r 的球结构 $\hat{G}[v,r]$（$v \in V$）并在这些球内计算图匹配结果，并利用球 $\hat{G}[v,r]$ 内的匹配结果增量计算出内层球 $\hat{G}[v,t]$（$t \in [1, r-1]$）中的图匹配结果，即可得到最终查询结果。

定理 4.2: 给定模式图 Q，数据图 G 中球结构 $\hat{G}[v,r]$ 和球结构 $\hat{G}[v,t]$（$t \in [1, r-1]$），（1）如果 $Q \prec \hat{G}[v,t]$，那么 $Q \prec \hat{G}[v,r]$ 成立；并且（2）如果 G_s 是球 $\hat{G}[v,r]$ 中 Q 的基于图模拟最大二元匹配关系 R_M 的匹配图，G'_s 是球 $\hat{G}[v,t]$ 中 Q 的基于图模拟最大二元匹配关系 R'_M 的匹配图，则 $R'_M \subset R_M$，G'_s 是 G_s 的子图。 □

当已经计算得到 Q 在 $\hat{G}[v,r]$ 中基于图模拟语义的匹配图 G_s 时，为了计算 Q 在 $\hat{G}[v,t]$（$t \in [1, r-1]$）中的完美匹配子图，需要执行以下步骤：（1）首先在 G_s 中标识出属于球 $\hat{G}[v,t]$ 的子图 G_s^t，这一标识过程可以在构建球 $\hat{G}[v,r]$ 的过程中实现，而不需要额外计算开销。（2）而后，检测 G_s^t 是否已经是 Q 在 $\hat{G}[v,t]$ 中基于图模拟的匹配图。如果不是，则将 G_s^t 中与 Q 不匹配的节点和边依次去除直至剩余子图 G'_s 与 Q 基于图模拟匹配。这一计算过程可以通过调用高效的图匹配增量算法实现 [50]。（3）最后，检测模式图上所有节点的容量区间是否能够被满足。如果满足，则 G'_s 是 Q 在 $\hat{G}[v,r]$ 中基于组合模拟语义匹配的完美匹配子图。

密度过滤技术。本文设计基于密度的过滤技术过滤掉一部分不可能包含最终 top-k 结果的球结构，从而进一步减少构建球结构和执行图匹配查询过程的次数，提高图匹配查询效率。密度过滤技术的核心是判定一个球结构是否包含最终 top-k 结果中的一个结果。

给定球结构 $\hat{G}[v,r]$，可以计算出 $\hat{G}[v,r]$ 中所有子图的密度上界 $\mathsf{den}_{\hat{G}_s}$，其中 $\hat{G}_s$ 是球 $\hat{G}[v,r]$ 的子图。如果密度上界 $\mathsf{den}_{\hat{G}_s}$ 大于当前发现的第 k 优匹配结果的密度，则球 $\hat{G}[v,r]$ 中可能存在最终 top-k 匹配结果中的一个结果。反之，则球 $\hat{G}[v,r]$ 中不可能存在最终 top-k 匹配结果。直接过滤掉球 $\hat{G}[v,r]$ 从而可以避免构建球结构和计算匹配结果等冗余计算。

密度过滤技术的关键问题是如何快速高效地为 G 中每个球结构计算其密度上界 $\mathsf{den}_{\hat{G}_s}$。调研可知即使当前最优的最大密度子图算法时间复杂度都为效率较低的三次方多项式时间 $O(|\hat{G}[v,r]|^3)$ [95]。本文将利用 [96] 提出的一个重要结论来解决这一问题。

引理 4.1: 给定图 H，den_{H_c} 为 H 中最大核 H_C 的密度，den_{H_d} 为 H 中最大密度子图 H_d 的密度。则（1）$\mathsf{den}_{H_c} \leq \mathsf{den}_{H_d} \leq 2 * \mathsf{den}_{H_c}$ 成立；并且（2）存在算法可以在多项式时间 $O(|E_H|)$ 内计算得出 den_{H_c} [96]。 □

表 10　基于组合模拟的 top-k 图匹配查询算法 batch

Algorithm：基于组合模拟的 **top-k** 图匹配查询算法 batch

Input: 模式图 $Q(V_Q, E_Q, l_Q)$，数据图 $G(V_G, E_G, l_G)$，正整数 r 和 k。
Output: 模式图 Q 在数据图 G 中的 top-k 完美匹配子图。

1. **if** Q is unsatisfiable **then return** *nil*;
2. $L_k := \emptyset$;
3. **for** each ball $\hat{G}[v, r]$ in G **do**
4. compute the maximum core $\hat{G}_C$ of the ball $\hat{G}[v, r]$;
5. **if** 2*$\text{den}_{\hat{G}_C}$ ≤ the density of the k-th result in L_k **then**
6. **continue** ;
7. $G_s := \text{ugSim}\ (Q, \hat{G}[v, r])$;
8. If G_s satisfies capacity bounds on Q **then** Insert G_s into L_k;
9. **for** each ball $\hat{G}[v, t]$ with $t \in [1, r-1]$ **do**
10. $G'_s := \text{incSim}\ (G_s, Q, \hat{G}[v, t])$;
11. If G'_s satisfies capacity bounds on Q **then** Insert G'_s into L_k;
12. **return** $L_k[0 : k-1]$;

Procedure：图模拟查询过程 $\text{ugSim}(Q, \hat{G}[v, r])$

Input: 模式图 $Q(V_Q, E_Q, l_Q)$ 和球 $\hat{G}[v, r]$。
Output: 模式图 Q 在球 $\hat{G}[v, r]$ 中的匹配图 G_s。

1. **for** each $u \in V_Q$ **do**
2. $\text{sim}(u) := \{w | w \text{ is in } \hat{G}[v, r] \text{ and } l_Q(u) \in l_G(w)\}$;
3. **while** there are changes **do**
4. **for** each edge (u, u') in E_Q and each node $w \in \text{sim}(u)$ **do**
5. **if** there is no edge (w, w') in $\hat{G}[v, r]$ with $w' \in \text{sim}(u')$ **then**
6. $\text{sim}(u) := \text{sim}(u) \setminus \{w\}$;
7. **if** $\text{sim}(u) = \emptyset$ **then return** $\emptyset$;
8. $R_M := \{(u, w) | u \in V_Q, w \in \text{sim}(u)\}$;
9. Construct the match graph G_s *w.r.t.* R_M;
10. **return** G_s;

图 H 中的最大核 H_C 是 H 的子图，其节点度至少为 ρ，其中 ρ 是最大可能值。根据引理 4.1，本文使用 $2 * \text{den}_{H_c}$ 作为 H 的密度上界过来滤不可能存在最终 top-k 结果的球。

基于以上两个优化技术，本文提出基于组合模拟的 top-k 图匹配查询的批处理算法。

基于组合模拟的 top-k 图匹配查询批处理算法 batch。算法伪代码如表 10 所示，该算法输入为模式图 Q 和数据图 G，正整数 r 和 k。算法输出密度最大的 top-k 完美匹配子图。算法具体工作流程如下。算法 batch 首先判定 Q 是否为可满足的（第 1 行）。如果 Q 是可满足模式图，则算法 batch 为 G 中每一个球结构 $\hat{G}[v,r]$ 计算其最大核 $\hat{G}_C$，而后采用密度过滤优化技术判定是否可以过滤掉该球（第 3-6 行）。如果满足过滤条件，则 batch 跳过当前球；如果不满足过滤条件，则 batch 在当前球 $\hat{G}[v,r]$ 中通过调用图模拟查询过程 ugSim（第 7 行）并执行节点容量区间检测（第 8 行），计算出 Q 在 $\hat{G}[v,r]$ 中基于结构模拟匹配语义的完美匹配子图。其中，过程 ugSim 是图模拟算法 [13, 28] 在无向图上的扩展。最后，算法 batch 在球 $\hat{G}[v,r]$ 的每个内层球 $\hat{G}[v,t]$ 中，通过调用增量计算过程 incSim 和执行节点容量区间检测计算出 Q 的完美匹配子图（第 9-11 行）。其中，过程 incSim 是图匹配查询增量算法的扩展 [50]。

图模拟（无向图）查询过程 ugSim。过程伪代码如表 10 所示，该过程输入为模式图 Q 和球结构 $\hat{G}[v,r]$，输出球 $\hat{G}[v,r]$ 中模式图 Q 基于图模拟匹配语义的匹配图。对于 Q 中任意一个节点 $u \in V_Q$，ugSim 首先计算得到 G 中包含 u 的标签的候选匹配节点集合 sim(u)，即 $l_Q(u) \in l_G(w)$（第 1-2 行）。而后，该算法不断循环为每个模式图节点 u，从 sim(u) 中去除经验证不匹配的数据图节点（第 3-7 行）。对于一个数据图节点 w，除非满足以下验证条件，否则算法将从 sim(u) 中将 w 去除：如果在 Q 中 u 有一个邻居节点 u'，那么在 G 中存在一个 w 的子节点 w' 使得 $w' \in \text{sim}(u')$ 成立。最终，ugSim 根据最大二元匹配关系 R_M 构建出匹配图 G_s 并将其作为查询结果返回 (第 8-10 行)。

算法正确性和复杂性分析。算法 batch 的正确性由以下性质保证：（1）由过程 ugSim 的正确性保证，可以按照与图模拟算法正确性证明相同的方法证明得出 [13]；由过程 incSim 的正确性保证，可以按照与图匹配增量算法正确性证明相同的方法证明得出 [50]。（2）由定理 4.2 和引理 4.1 保证。

模式图可满足性判定步骤花费时间 $O(|Q|^2)$，为所有外层球计算基于组合模拟语义的图匹配查询结果花费时间 $O(|V||Q||G|)$，为所有内层球增量计算基于组合模拟语义的图匹配查询结果花费时间 $O(r|V||V_Q||E|)$，为 $|V|$ 个外层球计算其最大核的密度花费时间 $O(|V||E|)$。所以，算法 batch 时间复杂度为 $O(|Q|^2+|V||Q||G|+r|V||V_Q||E|)$。然而，因为密度

过滤优化技术会显著提高查询效率，并且 $O(r|V||V_Q||E|)$ 是增量算法时间复杂度，而增量算法实际运行时间远低于时间复杂度所表示的最坏情况时间花费。所以，算法 batch 的实际运行时间远低于其算法时间复杂度所表示的最坏情况时间花费。

4.4 动态 top-k 图匹配查询

本章提出动态图匹配查询问题，考虑连续混合的模式图更新和数据图更新。本章在定义动态图匹配查询问题时和在第 4.5、4.6章论述动态图匹配查询方法工作原理时是以基于组合模拟的 top-k 图匹配查询语义为基础。然而，本文提出的动态图匹配查询方法是通用方法，适用于 top-k 图匹配查询和一般图匹配查询以及任意一种图匹配查询语义。

定义 4.7 (模式图更新): 模式图更新是由单元模式图更新构成的集合，记为 ΔQ。单元模式图更新分为以下五种：(1) 插入边操作，在模式图 Q 中两个节点之间插入一条边，(2) 删除边操作，从模式图 Q 中删除一条连接两个节点的边，(3) 插入节点操作，在模式图 Q 中插入一个节点，(4) 删除节点操作，在模式图 Q 中删除一个节点和所有邻接边，以及 (5) 区间更新操作，更新模式图 Q 的一个节点容量区间。默认所有操作均保持模式图 Q 为连通图。 □

定义 4.8 (数据图更新): 数据图更新是由单元数据图更新构成的集合，记为 ΔG。单元数据图更新分为四种，和模式图前四种单元更新定义相似。与模式图更新不同之处在于，数据图更新不需要保持数据图的连通性。 □

定义 4.9 (基于组合模拟的动态 top-k 图匹配查询): 给定模式图 Q、数据图 G、正整数 r 和 k、Q 在 G 中的 top-k 个完美匹配子图序列 $L_k(Q,G)$、模式图更新 ΔQ 和数据图更新 ΔG，基于组合模拟的动态 top-k 图匹配查询，记为 kDGPM$(Q,G,k,L_k,\Delta Q,\Delta G)$ ，即在 $G \oplus \Delta G$ 中找到 k 个密度最大的 $Q \oplus \Delta Q$ 的完美匹配子图，

其中，$\oplus$ 表示为 Q 执行更新操作 ΔQ，为 G 执行更新操作 ΔG。$Q \oplus \Delta Q$ 表示更新后的模式图，$G \oplus \Delta G$ 表示更新后的数据图。值得注意的是以上问题定义中考虑的动态环境包括连续的单独或混合的模式图和数据图更新，是迄今为止图匹配查询问题所考虑的最广泛和最实用的动态设置。 □

表 11　增量查询方法符号表

符号名称	符号描述
Q，G	模式图，数据图
$\hat{G}[v,r]$	数据图 G 中一个以节点 v 为球心以正整数 r 为半径的球结构
$L_k(Q,G)$	Q 在 G 中的 top-k 个完美匹配子图序列
ΔQ，ΔG	模式图更新，数据图更新
$\oplus$	为 Q 执行更新操作 ΔQ，为 G 执行更新操作 ΔG
$Q_h = \{Q_{fi}, C\}$	模式图分块：h 个模式图区块和割边集
AffBs	更新影响区域：受更新影响的球结构集合
$M(Q_{fi}, \hat{G}[v,r])$	模式图区块 Q_{fi} 在球结构 $\hat{G}[v,r]$ 中的最大二元匹配关系
$\tilde{M}(Q,G)$	区块-球匹配结果（辅助数据结构）
FS, BS	区块匹配状态结构，球匹配状态结构（辅助数据结构）
FBM	区块-球-匹配结果索引，包括 FS 和 BS （辅助数据结构）
BF, UP	球过滤结构，更新调度结构（辅助数据结构）

4.5　图匹配查询通用增量计算框架

本章首先分析基于组合模拟的动态 top-k 图匹配查询问题中存在的困难和动态图匹配查询方法的设计原则，而后提出一个统一的图匹配查询增量计算框架。为了方便查阅，本章将本文使用的符号总结在表 11 中。

4.5.1　理论分析

根据定理 4.2 可知，如果模式图 Q 与球 $\hat{G}[v,t]$（$t \in [1, r-1]$）基于图模拟语义匹配，那么 Q 肯定与球 $\hat{G}[v,r]$ 基于图模拟语义匹配，而且 Q 在 $\hat{G}[v,t]$ 中的查询结果可以由 Q 在 $\hat{G}[v,r]$ 中的查询结果通过简单的计算得出。所以，动态 top-k 图匹配查询问题只需重点关注 G 中所有半径为 r 的球 $\hat{G}[v,r]$ 即可。在后续章节内容中，本文在论述球结构相关操作时默认球半径为 r。

增量计算技术即在数据动态更新情况下通过复用已有查询结果计算数据更新后结果。其目标是通过最大化减少冗余计算从而提高查询效率技术。本文在章节 2.3.2 对增量计算理论和技术进行了详细介绍。本章将首先对增量计算理论中的基本概念进行回顾，而后利用增量计算理论对动态 top-k 图匹配查询问题从增量计算角度进行理论分析。

增量计算复杂性分析 [48, 49]。回顾章节 2.3.2 可知，相对于传统计算复杂性理论用算法的整个输入来评估算法的时间复杂度，[48, 49] 提出对于增量算法，应该用算法输入和输出变化大小的总和 |CHANGED| 来衡量算法的时间复杂度，其中 |CHANGED| 表示处理数据更新需要花费的固有计算成本。

如果一个增量计算问题存在时间复杂度可以表示为 |CHANGED| 的多项式函数的算法，则称该问题为有界增量问题。反之，则称该问题为无界增量问题。基于组合模拟的动态 top-k 图匹配查询问题是无界增量问题，同其他基于图模拟扩展语义的图匹配查询问题类似 [50, 51]。

定理 4.3: 基于组合模拟的动态 top-k 图匹配查询问题是无界增量问题。即使当特殊情况 $k = 1$ 并且只处理单元模式图更新或单元数据图更新时，该问题依然是无界增量问题。□

下面通过示例说明模式图更新和数据图更新对图匹配查询结果的影响。

例 4.6: 以例 4.1中模式图更新 ΔQ_1 和数据图更新 ΔG_1 为例，如图 13所示。

（1）对于 ΔQ_1，因为球结构 $\hat{G}[\mathsf{PM}_1, 2]$ 与 Q 匹配，所以 $\hat{G}[\mathsf{PM}_1, 2]$ 可能也与 $Q_1 \oplus \Delta Q_1$ 匹配。所以需要为 $\hat{G}[\mathsf{PM}_1, 2]$ 重新计算 $Q_1 \oplus \Delta Q_1$ 在其中的图匹配结果。对于 G_1 中除 $\hat{G}[\mathsf{PM}_1, 2]$ 外的所有球结构中，原本与 Q_1 不匹配的节点由于模式图产生更新 ΔQ_1，从而能够与 $Q_1 \oplus \Delta Q_1$ 匹配，进而生成完美匹配子图，所以需要为这些球结构重新计算 $Q_1 \oplus \Delta Q_1$ 在其中的图匹配结果。

（2）对于 ΔG_1，Q_1 在 $G_1 \oplus \Delta G_1$ 中的匹配结果在原有 Q_1 在 G_1 的匹配结果基础上增加了一个完美匹配子图，即包含节点 PM_2 的联通分量。 □

本章最后探讨为动态图匹配查询问题设计高效增量算法面临的困难与挑战以及需要遵守的设计原则。

（1）模式图更新和数据图更新的影响范围。基于定理 4.3 和例 4.6，分析可知（a）单元模式图更新的影响范围一般很大，可能导致先前图匹配查询结果整体发生改变，因此需要重新构建所有球结构并重新计算其中的图匹配查询结果。（b）单元数据图更新的影响范围也可能是全局的，所以也需要重新构建所有球结构并重新计算其中的图匹配查询结果。因此，设计动态图匹配查询算法的关键是识别出模式图和数据图更新的影响范围并尽量将影响范围局域化和最小化。

（2）维护辅助数据结构。增量算法通常会存储和维护一部分辅助数据结构用于存储 Q 在 G 中的中间计算结果和最终查询结果 [49, 50]。如何设计轻量级的辅助数据结构对于增

量算法的设计至关重要。一个直观的想法是存储 Q 在 G 的所有球结构中的最大二元匹配关系集合 $M(Q,G)$，正如现有的增量图匹配查询算法处理数据更新时设计的辅助数据结构一样 [50, 51]。然而，如例 4.6所示，模式图更新 ΔQ 的影响范围是全局的。如果存储 $M(Q,G)$ 作为辅助数据结构，对于模式图边/节点删除操作，需要重新构建所有球结构并重新计算其中的图匹配查询结果，即整个 $M(Q,G)$。因此，存储 $M(Q,G)$ 对于提高动态查询效率很可能毫无帮助，更不用说像常规增量算法所设计的只存储先前查询结果作为辅助数据结构 [48, 49]，即只存储 Q 在 G 中的 top-k 完美匹配子图。

（3）支持连续混合的模式图和数据图更新。一个实用的增量图匹配查询方法应该支持连续的单独或混合的模式图和数据图更新。这进一步增加了设计辅助数据结构和设计高效增量算法的难度。

4.5.2　通用增量计算框架

通过章节 4.5.1 的分析可以看到设计动态图匹配查询增量算法所面临的困难与挑战。尽管如此，在深入分析问题性质的基础上，本章设计了统一的图匹配查询增量计算框架。该框架可以处理连续的单独或混合的模式图和数据图更新，并且利用（i）模式图分块策略和（ii）标识更新影响区域策略来局域化模式图和数据图更新的影响范围，从而降低计算查询结果和维护辅助数据结构的运算成本。

（I）模式图分块策略。本文称 $\{Q_{f1}(V_{f1},E_{f1}),\ldots,Q_{fh}(V_{fh},E_{fh}),C\}$ 是模式图 $Q(V_Q,E_Q)$ 的一个 h-分块，记为 Q_h，当且仅当以下条件成立：（1）$\bigcup_{i=1}^{h} V_{fi} = V_Q$，（2）对于任意 $i \neq j \in [1,h]$，$V_{fi} \cap V_{fj} = \emptyset$ 成立，（3）E_{fi} 是 Q 中两个端点都属于 V_{fi} 的边的集合，并且（4）$C = E_Q \setminus (E_{f1} \cup \ldots \cup E_{fh})$。其中，本文称 Q_{fi}（$i \in [1,h]$）是 Q 的一个区块，C 是 Q 的割边集。

分析可知，通过模式图分块可以将 Q 上的模式图更新转换为在一个区块 Q_{fi} 上的更新或者在割边集 C 上的更新。模式图分块策略可以将模式图更新的影响范围局域化。此外，图模拟匹配语义在模式图分块策略下有优异性质，如下所示。

定理 4.4: 给定模式图 Q 的一个 h-分块 $\{Q_{f1},\ldots,Q_{fh},C\}$。对于数据图 G 中任意一个球结构 $\hat{G}$，如果区块 Q_{fi}（$i \in [1,h]$）在 $\hat{G}$ 中基于图模拟匹配的最大二元匹配关系为 M_i，并且 Q 在 $\hat{G}$ 中基于图模拟匹配的最大二元匹配关系为 M，则 $M \subseteq \bigcup_{i=1}^{h} M_i$ 成立。　□

本文称 M_i 为 Q 在球 $\hat{G}$ 中的部分二元匹配关系，或部分匹配关系。根据图模拟语义的性质 [13]，$\bigcup_{i=1}^{h} M_i$ 实际是 M 的中间结果。一旦计算得到 Q 在 $\hat{G}$ 中基于图模拟匹配的

最大二元匹配关系 M，即可通过容量区间检测等简单计算得到 Q 在 $\hat{G}$ 中基于组合模拟的图匹配查询结果。

也就是说，基于模式图分块策略 Q_h，可以存储并维护一个区块-球匹配结果集合作为增量算法的辅助数据结构，即模式图 Q 的所有区块在数据图 G 的所有球中基于图模拟匹配的最大二元匹配关系的集合 $\tilde{M}(Q,G)$。而且，辅助数据结构 $\tilde{M}(Q,G)$ 的存储空间成本很低，如实验章节中所示。

通过存储 $\tilde{M}(Q,G)$，对于每个球结构 $\hat{G}$，可以通过访问 $\tilde{M}(Q,G)$ 得到 Q 的所有区块在 $\hat{G}$ 中基于图模拟匹配的最大二元匹配关系的集合，即 $\bigcup_{i=1}^{h} M_i$。所以，当模式图的区块 Q_{fi} 发生更新时，只需要更新 M_i 并保持其他部分不变即可。也就是说，为了计算 $Q \oplus \Delta Q(\hat{G})$，只需要计算 $Q_{fi} \oplus \Delta Q(\hat{G})$，并将其与其他部分合并即可得到 $Q \oplus \Delta Q(\hat{G})$。更好的情况是，如果模式图更新是作用于割边集 C 上的更新，则不需要更新任何 M_i，只需要简单的组合计算将所有区块 M_i 合并即可得到 $Q \oplus \Delta Q(\hat{G})$。

为了实现高效的增量计算过程，我们期望（1）模式图的所有区块大小能够保持一致从而平衡更新的分布，并且（2）最小化组合所有区块得到更新查询结果所需要的计算开销。因此，本章定义并研究模式图优化分块问题。

给定模式图 Q 和一个正整数 h，模式图优化分块问题即找到 Q 的一个 h-分块使得 $\max(|V_{fi}|)$（$i \in [1,h]$）和 $|C|$ 同时最小。模式图优化分块问题是双目标优化问题。直观理解，其将模式图划分为大小基本相等的 h 个区块，并最小化割边集的大小。然而，该问题计算复杂性很高，如下所示。

定理 4.5: 模式图优化分块问题是NP-完全问题。即使当特殊情况 $h = 2$ 时，该问题依然是NP-完全问题。 □

尽管该问题的计算复杂性很高，但是，Q 和 h 在实际应用中通常很小 [28]，比如 $|Q| = 15$，$h = 3$。因此，本章设计了用于解决模式图优化分块问题的启发式算法，记为PFrag，算法伪代码如表 12 所示。算法PFrag 通过将模式图优化分块问题归约到被广泛研究的 (k,v)-*balanced partition* 问题 [97]，即可利用后者的算法解决模式图优化分块问题。(k,v)-*balanced partition* 问题即将图 H 的节点分为 k 个部分，使得每个部分的节点个数不大于 $v \cdot \frac{|V|}{k}$，并且不同部分之间的割边数目最小。该问题虽然不存在近似算法，但是存在很多高效的启发式算法 [98]。

模式图优化分块算法 PFrag。算法伪代码如表 12所示，该算法输入为模式图 Q 和一个正整数 h，输出模式图 Q 的一个 h-分块 $\{Q_{f1},\ldots,Q_{fh},C\}$。算法 PFrag 通过变化 v 值循环

表 12　模式图优化分块算法 PFrag

Algorithm：模式图优化分块算法 PFrag

Input: 模式图 $Q(V_Q, E_Q)$ 和正整数 h。
Output: 模式图 Q 的一个 h-分块 $\{Q_{f1}, \ldots, Q_{fh}, C\}$。

1. $k := h$; $\nu := h$; $M_Q := |V_Q|$; $M_C := 0$;
2. $(Q^{\nu}_{f1}, \ldots, Q^{\nu}_{fh}, C^{\nu}) :=$ BalanceP(k, ν);
3. $M^{\nu}_Q := \max\{|V_{Q^{\nu}_{f1}}|, \ldots, |V_{Q^{\nu}_{fh}}|\}$; $M^{\nu}_C := |C^{\nu}|$;
4. **while** $\max\{M_Q, M_C\} > \max\{M^{\nu}_Q, M^{\nu}_C\}$ **do**
5. $\quad Q_{f1} := Q^{\nu}_{f1}$, …, $Q_{fh} := Q^{\nu}_{fh}$;
6. $\quad C := C^{\nu}$; $M_Q := M^{\nu}_Q$; $M_C := M^{\nu}_C$;
7. $\quad$ **if** $M^{\nu}_Q \geq M^{\nu}_C$ **then** $\nu := \frac{\nu}{2}$ **else** $\nu := \frac{3\nu}{2}$;
8. $\quad (Q^{\nu}_{f1}, \ldots, Q^{\nu}_{fh}, C) :=$ BalanceP(k, ν);
9. $\quad M^{\nu}_Q := \max\{|Q^{\nu}_{f1}|, \ldots, |Q^{\nu}_{fh}|\}$; $M^{\nu}_C := |C^{\nu}|$;
10. **return** $\{Q_{f1}, \ldots, Q_{fh}, C\}$;

调用 (k, ν)-*balanced partition* 问题（$\nu \geq 1$）的启发算法 BalanceP(k, ν)，从而计算得到模式图 Q 一个 h-分块，并且 h-分块的所有区块 Q_{fi} 大小基本相等。具体而言，算法 PFrag 用 M_Q 表示当前最大区块的节点数目，用 M_C 表示当前割边集的数目。首先，算法 PFrag 将 M_Q 赋值为 $|V_Q|$，将 M_C 赋值为 0（第 1 行）。算法 PFrag 调用算法 BalanceP ，并且设定参数 k 和 ν 均为 h。也就是不为 Q 将要划分出的区块大小设定任何限制（第 2 行）。而后，算法 PFrag 执行循环步骤迭代的判定是否能够通过二分搜索调整参数 ν，从而改进新生成的 h-分块的质量（第 4-9 行）。如果当前最大区块的大小 M^{ν}_Q 不小于当前割边集的大小 M^{ν}_C，则调用算法 BalanceP ，并且设定参数 k 为 h，ν 为 $\frac{\nu}{2}$；反之则设定参数 k 为 h，ν 为 $\frac{3\nu}{2}$（第 7 行）。如果新生成的 Q 的 h-分块的质量不能再被改进，则 PFrag 返回该 h-分块作为最终 Q 的划分结果（第 10 行）。

算法正确性和复杂性分析。算法 PFrag 的正确性由算法 BalanceP 的正确性保证 [97, 98]。算法PFrag 的时间复杂度为 $O(\log h \cdot t_{\text{BalanceP}})$。其中，$t_{\text{BalanceP}}$ 是 (k, ν)-*balanced partition* 问题计算算法的时间复杂度。因为 $\nu \geq 1$，所以 PFrag 最多调用 $\log h$ 次算法 BalanceP 。

(II) 标识更新影响区域策略（AffBs ）。在模式图分块策略的基础上，本文进一步通过标识更新的影响区域将更新计算限制在一个区域（AffBs ）内，从而局域化模式图和数据图更新的影响范围，避免冗余计算。

本文称对于增量算法 $\mathcal{A}$ 而言，数据图 G 中的一个球结构 $\hat{G}$ 是受更新影响的，当且仅当在增量计算过程中，$\mathcal{A}$ 必须要访问球 $\hat{G}$ 才能得到更新后查询结果。本文称球 $\hat{G}$ 为受更新影响的球结构，记为AffB 。则更新影响区域即为所有受更新影响的球结构集合，记为AffBs 。

事实上，AffBs 是在模式图和数据图发生更新后，可能包含更新后图匹配查询结果的球结构集合。所以只需要访问 AffBs 而不需要访问其他球结构即可得到最终查询结果，从而避免冗余计算，提高查询效率。具体而言，（1）对于模式图更新 ΔQ，标识更新影响区域策略使得可以避免在每个球中为每个模式图区块更新部分匹配关系；（2）对于数据图更新 ΔG，基于组合模拟图匹配语义中的结果本地化性质，标识更新影响区域策略可以将数据图更新影响范围限制在一组结构被 ΔG 改变的球结构中。

(III) 图匹配查询通用增量计算框架。基于模式图分块策略和标识更新影响区域策略，本文提出统一的增量计算框架来处理连续的单独或混合的模式图和数据图更新。本文在本章和第 4.6 章论述该增量计算框架工作原理时是以基于组合模拟的 top-k 图匹配查询语义为基础。然而，该增量计算框架是图匹配查询的通用计算框架，可以适用于 top-k 图匹配查询和一般图匹配查询方法以及任意一种图匹配查询语义。

给定模式图 Q 的 h-分块 Q_h，数据图 G，两个正整数 r 和 k，和辅助数据结构（将在章节 4.6介绍），比如模式图所有区块在所有球结构（半径 r）中的部分匹配关系。给定模式图更新 ΔQ 和数据图更新 ΔG，基于统一增量计算框架的算法 dynamic 包括以下三个主要步骤。

(1)标识更新影响区域 AffBs。对于单独的模式图更新 ΔQ 和数据图更新 ΔG，算法 dynamic 调用两个不同计算过程分别处理 ΔQ 和 ΔG。对于混合的 ΔQ 和 ΔG，算法 dynamic 将两个计算过程生成的 AffBs 取并集。

(2) 更新 AffBs 中的部分匹配关系。对于受 ΔQ 影响的球，dynamic 增量计算得到更新后的模式图区块在 AffBs 中的部分匹配关系；对于受 ΔG 影响的球，dynamic 重新计算所有模式图区块在 AffBs 中的部分匹配关系；对于受 ΔQ 和 ΔG 共同影响的球，dynamic 采用和处理 ΔG 同样的更新方式。在更新部分匹配关系的同时，dynamic 同时更新维护辅助数据结构 FBM，以处理即将连续产生的单独或混合的模式图和数据图更新。

（3）合并部分匹配关系。最后，算法 dynamic 在 AffBs 的一部分可能产生最终查询结果的球中将所有部分匹配关系合并，并在这些球的内层球中计算查询结果，从而计算得到最终 top-k 完美匹配子图。

4.6　增量查询算法

本章将对增量算法 dynamic 进行详细介绍，包括（1）其使用的辅助数据结构，（2）算法 dynamicP 和算法 dynamicG 分别处理模式图和数据图更新，以及（3）算法 dynamic 通过将 dynamicP 和 dynamicG 集成为一个算法统一处理模式图和数据图更新。

4.6.1　辅助数据结构

辅助数据结构分为两类：第一类用于维护部分匹配关系，第二类用于处理模式图更新。给定模式图 $Q(V_Q, E_Q)$ 的一个 h-分块 $\mathcal{Q}_h = \{Q_{f1}, \ldots, Q_{fh}, C\}$，数据图 $G(V_G, E_G)$ 和模式图更新 ΔG。

（I）第一类辅助数据结构如下所示。

（1）区块匹配状态结构（FS）由 2^h 个布尔向量 $(b_1, \ldots, b_h)$ 构成，每个布尔向量被称为类型代码（tc），其中 b_i（$i \in [1, h]$）是 0 或 1。回顾章节 4.5.2 中模式图分块策略论述部分可知，h 通常很小，一般为 2 到 5 之间的数值。。

增量算法使用 FS 将 G 中所有球的匹配状态根据模式图分块情况 $\mathcal{Q}_h$ 分为 2^h 类。对于一个类型代码为 $(b_1, \ldots, b_h)$ 的球，b_i 为 1 当且仅当模式图区块 Q_{fi} 与该球基于图模拟语义匹配

（2）球匹配状态结构（BS）由 $|V_G|$ 个三元组 $(bid, cflag, den)$ 构成。其中，bid 是球的 id；$cflag$ 是该球中最后处理的单元模式图更新的 id（最初设置为 0）; den 是该球中所有子图的密度上界。增量算法使用 BS 存储 G 中所有球的基本信息。

（3）区块-球匹配结果是 Q 在 G 中所有部分匹配关系的集合，即 $\bigcup_{i\in[1,h], v\in V} M(Q_{fi}, \hat{G}[v, r])$，记为 $\tilde{M}(Q, G)$。其中，$M(Q_{fi}, \hat{G}[v, r])$ 为模式图区块 Q_{fi} 在球 $\hat{G}[v, r]$ 中基于图模拟匹配的最大二元匹配关系。

增量算法使用 $\tilde{M}(Q, G)$ 存储 Q 的所有区块在 G 的所有球中的部分匹配关系。相对于简单的按顺序存储 $\tilde{M}(Q, G)$ 中所有部分匹配关系，增量算法根据模式图区块和球之间的匹配状态来组织 $\tilde{M}(Q, G)$ 中部分匹配关系的存储。

（4）区块-球-匹配结果索引（FBM）将 FS 和 BS 链接在一起，构成区块-球-匹配结果索引。而后，将 FBM 链接至 $\tilde{M}(Q,G)$。构建细节如下所示。

对于 BS 中每个球结构 $\hat{G}[v,r]$ 的记录，（a）将 FS 中该球对应的类型代码记录链接至 BS 中该球的记录；（b）将 BS 中该球的记录链接至 $\tilde{M}(Q,G)$ 中该球对应的部分匹配关系集合 $M(Q_{fi},\hat{G}[v,r])$（$i \in [1,h]$）。其中，如果该球对应的类型代码中的一位 $b_i = 1$，则 $M(Q_{fi},\hat{G}[v,r])$ 非空。

直观理解，FBM 根据模式图区块和数据图中球之间的匹配状态来索引 $\tilde{M}(Q,G)$ 中存储的部分匹配关系。

例 4.7: 以图 13 中模式图 Q_1 和数据图 G_1 为例（均不包括虚线边），给定 $r = 2$，$k = 2$，$h = 2$。则其增量算法的辅助数据结构 FBM 和 $\tilde{M}(Q,G)$ 如图 15 所示。

（1）模式图 Q_1 被算法PFrag 划分为两个区块 Q_{f1} 和 Q_{f2}，所以 FS 中有 $2^2 = 4$ 个类型代码。

（2）对于类型代码 tc 为 $(1,1)$ 的球，例如球 $\hat{G}[\mathsf{PM}_1,2]$，区块 Q_{f1} 和区块 Q_{f2} 均与该球基于图模拟语义匹配。另外，球 $\hat{G}[\mathsf{PM}_2,2]$ 的 tc 为 $(1,0)$, 球 $\hat{G}[\mathsf{BA}_3,2]$ 的 tc 为 $(0,1)$，球 $\hat{G}[\mathsf{PM}_4,2]$ 的 tc 为 $(0,0)$。为了便于分析，本章在后续讨论中只考虑这四个球结构。 □

以上设计的辅助数据结构具有如下优异性质。

定理 4.6: 使用辅助数据结构 $\tilde{M}(Q,G)$ 和 FBM，给定模式图更新 ΔQ 和数据图更新 ΔG，增量算法 dynamic 处理更新 ΔQ 和 ΔG 的时间由 Q、$\tilde{M}(Q,G)$ 和 AffBs 确定，而不直接取决于 G。 □

本章将通过为增量算法 dynamic 设计具体的计算过程并分析其时间复杂度来证明定理 4.6。

（II）第二类辅助数据结构如下所示。

（1）球过滤结构（BF）由 2^h 个布尔向量 $(b_1,\ldots,b_h)$ 构成，其中每个布尔向量被称为过滤代码（fc），并且对应于区块匹配状态结构 FS 中的一个类型代码 tc。其中，BF 中的过滤代码 fc 的每个 b_i（$i \in [1,h]$）都被初始化为 1，并针对 ΔQ 中的每个单元模式图更新 δ 进行更新：（a）当 δ 是区块 Q_{fi} 上的删除边操作或删除节点操作时，BF 中所有过滤代码 fc 的第 i 位更新为 0；否则，BF 保持不变。

（2）更新调度结构（UP）由 $h+1$ 个堆栈 $T(Q_{f1})$，…，$T(Q_{fh})$，$T(C)$ 构成。堆栈 $T(Q_{fi})$（$i \in [1,h]$）用于记录迄今为止到达的所有模式图更新集合 ΔQ_1，…，ΔQ_N 中的所有单元

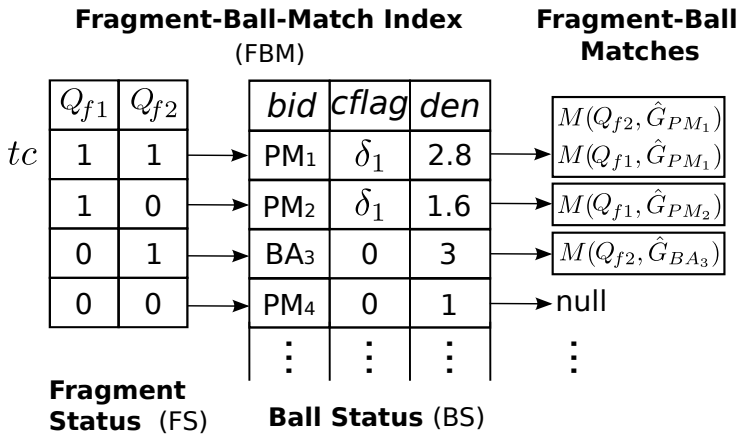


图 15　辅助数据结构示例（I）：FBM，$\tilde{M}(Q,G)$

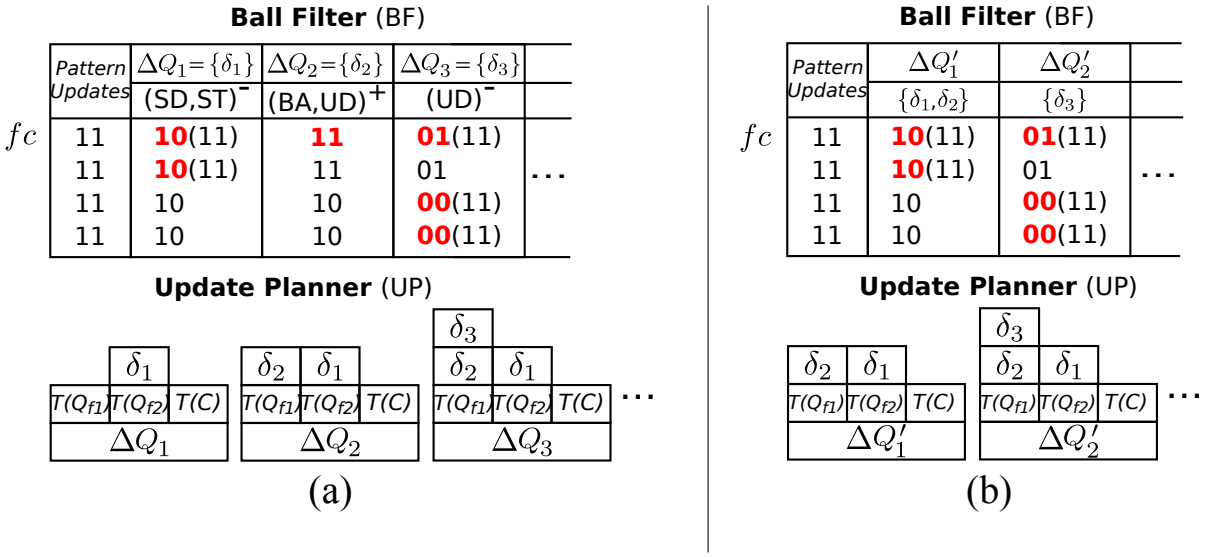


图 16　辅助数据结构示例（II）：BF，UP

更新中，针对区块 Q_{fi} 的单元模式图更新。同样的，堆栈 $T(C)$ 用于记录针对割边集 C 的单元模式图更新。所有堆栈的初始状态均为空。每个堆栈均根据每个到来的模式图更新集合中的每个单元更新进行动态更新。

4.6.2　模式图更新增量算法

根据章节 4.5.2 的算法设计框架，本章将设计算法 dynamicP 来处理模式图更新 ΔQ，并设计提前返回优化技术用于提高增量算法的效率。

（I）标识更新影响区域。本章首先设计算法 dynamicP 的计算过程 IdABall，通过利用辅助数据结构 FBM 和 BF 标识更新影响区域 AffBs，即所有受更新影响的球结构。

计算过程 IdABall。给定模式图 Q 的一个 h-分块和模式图更新 ΔQ，（1）IdABall 首先根据 ΔQ 中的所有单元更新来更新 BF。（2）对于每个 $j \in [1, 2^h]$，IdABall 为 FBM 中 FS 的每个类型代码 tc_j 和 BF 的每个过滤代码 $fc_{j\Delta}$ 执行按位与（&）操作，即 $tc_j \& fc_{j\Delta}$。（3）最后，如果 $tc_j \& fc_{j\Delta} = fc_{j\Delta}$ 成立，则 IdABall 通过访问 FBM 中的 BS 将所有类型代码为 tc_j 的球结构全部标注为更新影响球 AffBs，而后将过滤代码 $fc_{j\Delta}$ 重置为 $(1, \ldots, 1)$。当以下条件成立时，则以上判定条件 $tc_j \& fc_{j\Delta} = fc_{j\Delta}$ 成立：（a）$fc_{j\Delta}$ 的第 i 位（$i \in [1, h]$）为 0，即 Q_{fi} 上存在删除边/节点操作，则可能在数据图中产生更多的匹配节点，或者（b）$fc_{j\Delta}$ 和 tc_j 的第 i 位（$i \in [1, h]$）都是 1，即类型代码为 tc_j 的球已经与 Q_{fi} 匹配，并且 Q_{fi} 上没有删除边/节点操作。

例 4.8: 以例 4.7 中的输入和辅助数据结构为例，球过滤结构 BF 如图 16 所示。

（1）模式图更新 ΔQ_1 包括一个针对区块 Q_{f2} 的删除边操作 δ_1=(SD,ST)$^-$。则 BF 中四个过滤代码均由 $fc\,(1,1)$ 更新为 $fc_{\Delta}\,(1,0)$，如图 16(a) 中 BF 第二列所示。IdABall 识别类型代码为 $tc\,(1,1)$ 的球 $\hat{G}[\mathrm{PM}_1, 2]$ 和类型代码为 $tc\,(1,0)$ 的球 $\hat{G}[\mathrm{PM}_2, 2]$ 为更新影响区域 AffBs，因为 $tc(1,1) \& fc_{\Delta}(1,0) = fc_{\Delta}(1,0)$ 和 $tc(1,0) \& fc_{\Delta}(1,0) = fc_{\Delta}(1,0)$ 成立。而后，IdABall 将 BF 中相应的过滤代码重置为 $(1,1)$。

（2）考虑另一种情况，当模式图更新 $\Delta Q_1'$ 包括一个针对区块 Q_{f2} 的删除边操作 δ_1=(SD,ST)$^-$ 和另一个针对区块 Q_{f1} 的插入边操作 $\delta_2 = (\mathrm{BA}, \mathrm{UD})^+$。BF 更新如图 16(b) 中 BF 第二列所示。IdABall 识别出与上面相同的更新影响区域 AffBs。 □

计算过程 IdABall 的正确性由以下定理保证。

定理 4.7: 对于任意一个 G 中的球 $\hat{G}[v, r]$，如果更新后的模式图 $Q \oplus \Delta Q$ 在 $\hat{G}[v, r]$ 中存在完美匹配子图，则球 $\hat{G}[v, r]$ 肯定是由计算过程 IdABall 标识出的受更新影响的球。 □

延迟更新策略。为了减少计算量，算法 dynamicP 只更新 $\tilde{M}(Q, G)$ 中与 AffBs 相关的部分匹配关系用于计算 top-k 完美匹配子图 $L_k(Q \oplus \Delta Q, G)$。然而，虽然被过滤掉的球中不可能包含 $L_k(Q \oplus \Delta Q, G)$ 中结果，但是可能会包含下一轮模式图更新 $\Delta Q'$ 后的结果，即 $L_k(Q \oplus \Delta Q \oplus \Delta Q', G)$ 中结果。但是，由于算法 dynamicP 在处理更新 ΔQ 时判定不需要更新被过滤掉的球在 $\tilde{M}(Q, G)$ 中的部分匹配关系，所以，在处理下一轮更新 $\Delta Q'$ 时，对于上一轮被过滤掉的球，无法利用其在 $\tilde{M}(Q, G)$ 中的部分匹配关系执行增量计算。因此，算法 dynamicP 需要设计一个可以智能维护 $\tilde{M}(Q, G)$ 的策略，使得既不影响标识更新影响区域策略的效果，也不影响增量计算的工作流程。

要实现这个目标，算法 dynamicP 需要维护迄今为止针对 Q 的所有单元更新，并且尽可能的推迟处理 ΔQ 中的单元更新，并且不影响处理下一轮到来的更新 $\Delta Q'$，这就是延迟更新策略。

算法 dynamicP 使用辅助数据结构 UP 和 BS 中的 $cflag$ 属性来实现延迟更新。在处理当前更新 ΔQ 时，对于每个球结构 $\hat{G}$，$\hat{G}$[cflag] 记录 $\hat{G}$ 中已经处理的最后一个单元模式图更新的 id，这个值初始化为 0。当处理下一轮更新 $\Delta Q'$ 时，对于受 $\Delta Q'$ 影响的任何一个 AffB $\hat{G}$ 和任何一个模式图区块 Q_{fi}，算法 dynamicP 通过调用计算过程 IncMatch（稍后介绍）将 $M(Q_{fi}, \hat{G})$ 更新为 $M(Q_{fi} \oplus \Delta Q'_{fi}, \hat{G})$。其中，$\Delta Q'_{fi}$ 由存储在堆栈 $T(Q_{fi})$ 中单元更新 id 大于 BS 中 $\hat{G}$[cflag] 值的所有单元更新构成。

例 4.9: 继续以例 4.8 中内容为例。（1）$\hat{G}[\mathrm{PM}_1, 2]$ 和 $\hat{G}[\mathrm{PM}_2, 2]$ 都是 AffBs，并且 UP 的更新如图 16(a) 所示。

（a）UP 已经根据 $\Delta Q_1 = \{\delta_1\}$ 相应更新为如图 16(a) 所示状态。算法 dynamicP 增量计算 Q_{f2} 上执行更新 δ_1 后在 $\hat{G}[\mathrm{PM}_1, 2]$ 和 $\hat{G}[\mathrm{PM}_2, 2]$ 中的部分匹配关系，并将两个球在 BS 中的相应 $cflag$ 属性值设置为 δ_1，更新状态如图 15 所示。

（b）之后，模式图更新 $\Delta Q_2 = \{\delta_2 = (\mathrm{BA}, \mathrm{UD})^+\}$ 产生。计算过程 IdABall 更新 BF 和 UP，状态如图 16(a) 所示。IdABall 将类型代码为 tc $(1, 1)$ 的球标识为 AffBs，即球 $\hat{G}[\mathrm{PM}_1, 2]$。

（c）最后，模式图更新 $\Delta Q_3 = \{\delta_3 = (\mathrm{UD})^-\}$ 产生。IdABall 将类型代码为 tc $(1, 1)$, $(0, 1)$ 和 $(0, 0)$ 的球标识为 AffBs。以 $\hat{G}[\mathrm{BA}_3, 2]$ 为例，该球是 AffB。通过访问 UP，算法 dynamicP 增量维护 Q_{f1} 上执行更新 $\{\delta_2, \delta_3\}$ 后在球 $\hat{G}[\mathrm{BA}_3, 2]$ 中的部分匹配关系，以及 Q_{f2} 上执行更新 $\{\delta_1\}$ 后在球 $\hat{G}[\mathrm{BA}_3, 2]$ 中的部分匹配关系。

（2）当 $\Delta Q'_i$ 包含多个单元模式图更新时，BF 和 UP 的更新如图 16(b) 所示。　□

（II）更新区块-球匹配结果。本节设计算法 dynamicP 的计算过程 IncMatch，用于更新 $\tilde{M}(Q, G)$ 中与 AffBs 和 ΔQ 相关的部分匹配关系。

计算过程 IncMatch。给定模式图 Q 的 h-区块，数据图 G，区块-球匹配结果 $\tilde{M}(Q, G)$，模式图更新 ΔQ，UP 和根据 ΔQ 识别出的 AffBs。对于 AffBs 中的每个球 $\hat{G}$ 和每个区块 Q_{fi}，IncMatch 将其在 $\tilde{M}(Q, G)$ 中的部分匹配关系由 $M(Q_{fi}, \hat{G})$ 更新为 $M(Q_{fi} \oplus \Delta Q_{fi}, \hat{G})$。回顾延迟更新策略可知 ΔQ_{fi} 包括 UP 中积累的迄今为止未处理的针对区块 Q_{fi} 的单元模式图更新。下面将介绍在不同情况下 IncMatch 更新 $M(Q_{fi}, \hat{G})$ 的方法。

（*1*）ΔQ_{fi} 中存在删除边/节点操作。在这种情况下，IncMatch 需要构建 G 中的 AffB 球

表 13　模式图插入边操作增量计算过程 patEIns

Procedure：模式图插入边操作增量计算过程 patEIns

Input: 部分匹配关系 $M(Q_{fi}, \hat{G}[v,r])$ 和模式图插入边操作 $\delta = (u,u')^+$。

Output: 更新后的部分匹配关系 $M(Q_{fi} \oplus \delta, \hat{G}[v,r])$。

```
1.  RMv := ∅;
2.  for each u ∈ V_Q do sim(u) := {w | (u,w) ∈ M(Q_fi, Ĝ[v,r])};
3.  for each node w ∈ sim(u) do
4.      if there exists no (w,w') ∈ E_Ĝ[v,r] with w' ∈ sim(u') then
5.          RMv.push([u,w]);
6.  for each node w' ∈ sim(u') do
7.      if there exists no (w',w) ∈ E_Ĝ[v,r] with w ∈ sim(u) then
8.          RMv.push([u',w']);
9.  while RMv ≠ ∅ do
10.     [u,w] := RMv.pop(); sim(u) := sim(u) \ {w};
11.     for each (u,u') ∈ E_Qfi do
12.         for each (w,w') ∈ E_Ĝ[v,r] with w' ∈ sim(u') do
13.             if there is no (w',w'') ∈ E_Ĝ[v,r] with w'' ∈ sim(u) then
14.                 RMv.push ([u',w']);
15. if there is a node u ∈ V_Qfi with |sim(u)| = 0 then sim(·) := ∅;
16. M(Q_fi ⊕ δ, Ĝ[v,r]) := {(u,w) | u ∈ V_Q, w ∈ sim(u)};
17. return M(Q_fi ⊕ δ, Ĝ[v,r]);
```

$\hat{G}[v,r]$。IncMatch 通过调用计算过程 ugSim 计算出 $Q_{fi} \oplus \Delta Q_{fi}$ 在 $\hat{G}[v,r]$ 中的最大二元匹配关系，这一过程时间复杂度为 $O(|Q_{fi} \oplus \Delta Q_{fi}||\hat{G}[v,r]|)$。

（2）ΔQ_{fi} 中不存在删除边/节点操作。在这种情况下，IncMatch 将 ΔQ_{fi} 中的单元更新按不同更新类型分类，并按以下方法单独处理每类更新。

（i）ΔQ_{fi} 或 C 中的区间更新操作。在这种情况下，不需要对任何与 AffBs 和 ΔQ_{fi} 相关的部分匹配关系进行更新计算。因为在这种情况下 $M(Q_{fi} \oplus \Delta Q_{fi}, \hat{G}) = M(Q_{fi}, \hat{G})$ 成立。只需要在合并计算过程中执行容量区间检查以及与内部球相关的简单计算即可。

（ii）ΔQ_{fi} 中的插入边操作。在这种情况下，IncMatch 通过调用计算过程 patEIns 增量处理插入边更新操作。

计算过程 patEIns。给定部分匹配关系 $M(Q_{fi}, \hat{G}[v,r])$（也写作 sim(·)）和模式图插入边操作 $\delta = (u,u')^+$，patEIns 增量计算得到更新后的部分匹配关系 $M(Q_{fi} \oplus \delta, \hat{G}[v,r])$，计算过程伪代码如表 13 所示。patEIns 与图匹配查询的动态数据图增量查询算法类似 [50]。patEIns 首先识别出受 Q_{fi} 上的插入边操作直接影响而需要从 sim(·) 中删除的数据图节点，并将他们与匹配的模式图节点一起作为二元组压入堆栈 RMv 中（第 3-8 行）。而后，patEIns 迭代的从 sim(·) 中识别并删除受先前删除节点影响而需要去除的数据图节点（第 9-14 行）。patEIns 通过使用堆栈 RMv 实现上述迭代计算过程。在迭代过程中，如果存在一个模式图节点 u 在数据图中的匹配节点集合 sim(u) 为空，则将 sim(·) 设置为空集 $\emptyset$（第 15 行）。最后，patEIns 返回更新后的模式图区块 $Q_{fi} \oplus \delta$ 在数据图中更新后的部分匹配关系 sim(·)（第 16-17 行）。

例 4.10: 以例 4.9 中情况 (1)-(b) 为例。给定模式图更新操作 $\delta_2 = (\text{BA}, \text{UD})^+$，和部分匹配关系 $M(Q_{f1}, \hat{G}_{\text{PM}_1})$。其中，$M(Q_{f1}, \hat{G}_{\text{PM}_1})$ 包括区块 Q_{f1} 中的模式图节点 PM 、BA 和UD，分别匹配至数据图节点 {PM_1 }、{BA_1 } 和 {UD_1,UD_2 }。为了计算更新后的部分匹配关系 $M(Q_{f1} \oplus \delta_2, \hat{G}_{\text{PM}_1})$，patEIns 删除直接受 δ_2 影响的数据图节点UD_2，并发现不需要删除任何其他节点。 □

（ii）ΔQ_{fi} 中的插入节点操作。通过扩展计算过程 patEIns，可以用与处理插入边操作类似的方式处理插入节点操作。给定插入节点操作 $\delta = (u,(u,u'))^+$，其中 u 是新插入的节点。为了计算更新后的部分匹配关系 $M(Q_{fi} \oplus \delta, \hat{G}[v,r])$，计算过程 IncMatch 首先计算出 $\hat{G}[v,r]$ 中与 u 具有相同标签的数据图节点集合 sim(u) 作为候选匹配节点集合。然后，IncMatch 调用 patEIns($M(Q_{fi}, \hat{G}[v,r]), \hat{G}[v,r], (u,u')$) 得到更新后的部分匹配关系。

更新 FBM。在更新完毕 $\tilde{M}(Q,G)$ 后，算法 dynamicP 为所有 AffBs 中的球结构更新其在 FBM 中的相应信息，包括根据更新后的部分匹配关系更新 FBM 与 $\tilde{M}(Q,G)$ 之间的链接，以及更新 BS 中的相应cflag 属性信息。更新 FBM 操作可以在 $O(||\text{AffBs}||)$ 时间内执行完毕。

(III) 合并区块-球匹配结果。最后，本节设计算法 dynamicP 的计算过程 combine，用于合并 AffBs 中更新后的部分匹配关系，从而计算得到 top-k 完美匹配子图序列 $L_k(Q \oplus \Delta Q, G)$。分析可知，AffBs 中只有与 $Q \oplus \Delta Q$ 的所有模式图区块都匹配的球才可能产生最终结果 $L_k(Q \oplus \Delta Q, G)$，也就是 AffBs 中只有一部分球需要执行合并计算过程来得到最终结果。

计算过程 combine。对于每个 AffB 球结构 $\hat{G}[v,r]$，计算过程 combine 通过调用 patEIns($\bigcup_{i\in[1,h]} M(Q_{fi},\hat{G}[v,r]),\hat{G}[v,r],C\oplus\Delta C$) 增量计算 $Q\oplus\Delta Q$ 在 $\hat{G}[v,r]$ 中的最大匹配关系。其中 ΔC 包括 ΔQ 中所有针对割边集 C 的插入边和删除边更新。而后，combine 检查合并后的最大匹配关系是否满足更新后的容量区间限制。如果满足，则构建基于最大匹配关系的完美匹配子图。之后，基于 $Q\oplus\Delta Q$ 在 $\hat{G}[v,r]$ 中的最大匹配关系，combine 为球 $\hat{G}[v,r]$ 的每个内层球计算其中的最大匹配关系，并执行容量区间检查和构建完美匹配子图。combine 最后返回 $Q\oplus\Delta Q$ 在 G 中的 top-k 完美匹配子图序列。

例 4.11: 继续以例 4.9(1) 中内容为例。当 dynamicP 根据 ΔQ_3 更新完毕 AffBs 中的部分匹配关系后，dynamicP 判定球 $\hat{G}[\text{PM}_1,2]$、$\hat{G}[\text{PM}_2,2]$ 和 $\hat{G}[\text{BA}_3,2]$ 进入最终合并计算过程。（1）对于球 $\hat{G}[\text{PM}_1,2]$ 和 $\hat{G}[\text{PM}_2,2]$，因为 $C\oplus\Delta C=\{(\text{PM},\text{SA})\}$，combine 计算得知对于每个节点$\text{PM}_j$，都存在一个节点$\text{SA}_i$ 与其连接，反之对于每个节点SA_j，都存在一个节点PM_i 与其连接，并且匹配结果满足模式图的容量区间限制。另外，combine 计算得知球 $\hat{G}[\text{PM}_1,2]$ 的内层球 $\hat{G}[\text{PM}_1,1]$ 和球 $\hat{G}[\text{PM}_2,2]$ 的内层球 $\hat{G}[\text{PM}_2,1]$ 中均不存在完美匹配子图。所以，combine 返回球 $\hat{G}[\text{PM}_1,2]$ 和球 $\hat{G}[\text{PM}_2,2]$ 中的两个完美匹配子图。（2）对于球 $\hat{G}[\text{BA}_3,2]$，combine 无法找到与节点PM_j 连接的节点SA_j，也无法找到与节点SA_j 连接的节点PM_j。所以，combine 判定球 $\hat{G}[\text{BA}_3,2]$ 中不存在完美匹配子图。 □

（IV）提前返回优化技术。本节设计了提前返回优化技术，用于提高算法 dynamicP 的查询效率。提前返回优化技术是针对 top-k 增量算法的优化技术，类似于 top-k 批处理算法的提前终止优化技术 [99]。下面首先对提前返回性质进行定义。

提前返回性质。给定模式图 Q 和模式图更新 ΔQ，如果一个增量算法在任何数据图 G 上能够尽早输出查询结果 $L_k(Q\oplus\Delta Q,G)$，而无需为 AffB 中每个球更新部分匹配结果，并将剩余更新工作调度至后台执行，则称该增量算法具有提前返回性质。

定理 4.8: 基于组合模拟的动态 top-k 图匹配查询问题存在具有提前返回性质的增量算法。 □

下面证明增量算法 dynamicP 具有提前返回性质。回顾章节 4.3.2 中提出了用于提高批处理算法 batch 效率的密度过滤优化技术。增量算法 dynamicP 也使用球结构的密度上界来过滤掉 AffBs 中一部分不可能存在最终 top-k 结果的球结构。具体而言，给定模式图 Q 和数据图 G，dynamicP 利用辅助数据结构 BS 中的 den 属性维护了 G 中每个球的密度上界，即 $\hat{G}[\text{den}]$。根据引理 4.1 可以计算得到球结构的密度上界。因此，给定模

式图更新 ΔQ，如果当前发现的第 k 优完美匹配子图的密度大于 AffBs 中所有未被处理的球结构密度上界的最大值，则算法 dynamicP 输出当前最优 k 个完美匹配子图作为最终查询结果 $L_k(Q \oplus \Delta Q, G)$，同时继续在后台更新 AffBs 中所有未被处理的球结构的部分匹配关系。

值得注意的是提前返回优化技术对于模式图更新是有效的，但不适用于数据图更新以及包含插入带有新标签节点操作的模式图更新。

（V）完整模式图更新增量算法。给定辅助数据结构 $\tilde{M}(Q,G)$，FBM，BF 和 UP。对于模式图更新 ΔQ，算法 dynamicP 通过依次调用计算过程 IdABall，IncMatch 和 combine 计算并提前返回更新后的模式图 $Q \oplus \Delta Q$ 在数据图 G 中的 top-k 完美匹配子图，并且同时更新和维护辅助数据结构。

模式图更新增量算法 dynamicP。算法伪代码如表 14 所示。对于每个模式图更新 ΔQ，算法 dynamicP 首先将查询结果序列 L_k 设置为空。并调用 IdABall 根据 ΔQ 标识出 AffBs（第 1-2 行）。而后，dynamicP 将 AffBs 中的球根据 BS 中球的密度上界 $\hat{G}$[den] 按非升序排列（第 3 行），并按此顺序依次访问 AffBs 中每个球（第 4-10 行）。每当访问至下一个 AffB 球 $\hat{G}[v,r]$ 时，dynamicP 首先检查 L_k 中是否已经有 k 个完美匹配子图，并且 L_k 中第 k 个密度最大的完美匹配子图的密度是否大于 $\hat{G}[v,r]$[den]（第 5 行）。如果大于，则 dynamicP 立即输出当前 L_k 中前 k 个完美匹配子图作为最终查询结果（第 6 行），然后通过调用 IncMatch 继续在后台更新 AffBs 中所有未被处理的球在 $\tilde{M}(Q,G)$ 中的部分匹配关系（第 7 行）；否则，dynamicP 先调用 IncMatch 来更新部分匹配关系，再调用 combine 将对应于所有模式图区块的所有部分匹配关系合并以获得球 $\hat{G}[v,r]$ 中和其内部球中的完美匹配子图集合 S_{G_s}（第 7-8 行）。之后，dynamicP 将 S_{G_s} 中的完美匹配子图插入 L_k 中（第 9 行）。

算法正确性和复杂性分析。算法 dynamicP 的正确性由以下计算过程的正确性保证：（a）IdABall（根据定理 4.7）；（b）IncMatch，其正确性由计算过程 ugSim 的正确性（章节 4.3 中证明）和计算过程 patEIns 的正确性保证。patEIns 是正确的因为其只删除 $M(Q_{fi}, \hat{G}[v,r])$ 中不再与更新后的模式图区块匹配的数据图节点；（c）combine，其正确性分析方法与 patEIns 的正确性分析方法相同；和（d）提前返回优化技术（根据引理 4.1）。

计算过程 IdABall 的时间复杂度为 $O(2^h \cdot (|\Delta Q| + h) + ||\text{AffBs}||)$。其中，更新 BF 时间花费为 $O(2^h \cdot |\Delta Q|)$，执行按位与操作时间花费为 $O(2^h \cdot h)$，识别 AffBs 时间花费为 $O(||\text{AffBs}||)$。由于 h 通常很小，一般为 2 到 5 之间的数值。所以 IdABall 的时间复杂度为

表 14　模式图更新增量算法 dynamicP

Algorithm：模式图更新增量算法 dynamicP

Input: 模式图 Q，h-区块 Q_h，数据图 G，正整数 r 和 k，模式图更新 ΔQ，辅助数据结构 $\tilde{M}(Q,G)$，FBM，BF 和 UP。

Output: 更新后的模式图 $Q \oplus \Delta Q$ 在数据图 G 中的 top-k 完美匹配子图。

1. $L_k := \emptyset$;
2. AffBs := IdABall($Q_h, \Delta Q$, FBM, BF);
3. Sort AffBs by $\hat{G}$[den] in non-ascending order;
4. **for each** $\hat{G}[v,r]$ in AffBs **do**　　/* 递减顺序 */
5. **if** $|L_k| \geq k$ **and** $\hat{G}[v,r]$[den] $\leq$ den$_{L_k[k-1]}$ **then**
6. **Output** $L_k[0:k-1]$.　　/* 提前返回优化技术 */
7. IncMatch($M(Q_{fi}, \hat{G}[v,r]), \hat{G}[v,r], \Delta Q_{fi}$) ($i \in [1,h]$);
   /* 后台继续运行 */
8. S_{G_s} := combine($\bigcup_{i\in[1,h]} M(Q_{fi}, \hat{G}[v,r]), \hat{G}[v,r], C \oplus \Delta C$);
9. Insert the set of perfect subgraphs in S_{G_s} into L_k;
10. **return** $L_k[0:k-1]$.

$O(|\Delta Q| + \|\text{AffBs}\|)$；计算过程 patEIns 的时间复杂度为 $O(|M(Q_{fi}, \hat{G}[v,r])| + |V_{Q_{fi}}||E_{\hat{G}[v,r]}|)$。其中，迭代删除 $M(Q_{fi}, \hat{G}[v,r])$ 中无效节点过程的时间花费为 $O(|V_{Q_{fi}}||E_{\hat{G}[v,r]}|)$，因为更新 $M(Q_{fi}, \hat{G}[v,r])$ 的时间花费取决于其变化大小，并且这种变化是单调递减的；计算过程 combine 的时间复杂度为 $O(|\bigcup_{i=1}^{h} rM(Q_{fi} \oplus \Delta Q_{fi}, \hat{G}[v,r])| + r|V_{Q\oplus\Delta Q}||E_{\hat{G}[v,r]}|)$。

由上可知，算法 dynamicP 处理 ΔQ 的时间复杂度为 $O(\bigcup_{\hat{G}\in\text{AffBs}} \bigcup_{i\in[1,h]}(|M(Q_{fi}, \hat{G})| + r|M(Q_{fi} \oplus \Delta Q_{fi}, \hat{G})|) + r|Q \oplus \Delta Q\|\text{AffBs}| + |\Delta Q|)$。其中，$r$ 通常很小，取值一般为 2 或 3。

4.6.3　数据图更新增量算法

根据章节 4.5.2 的算法设计框架，本章将设计算法 dynamicG 来处理数据图更新 ΔG。给定辅助数据结构 $\tilde{M}(Q,G)$ 和 FBM，算法 dynamicG 整合了计算过程 IdABall，IncMatch 和 combine，计算得到模式图 Q 在更新后的数据图 $G \oplus \Delta G$ 中的图匹配查询结果。由于计算过程 IncMatch 和 combine 处理 ΔG 的方法与处理 ΔQ 的方法基本相同，所以本章主

要介绍计算过程 IdABall 如何根据 ΔG 标识 AffBs。

计算过程 IdABall。给定数据图 G，数据图更新 ΔG 和辅助数据结构 FBM，IdABall 根据下述引理标识 AffBs。

引理 4.2: 根据 ΔG 和 FBM，当满足以下条件之一时，数据图 G 中以节点 v 为球心的球结构 $\hat{G}[v,r]$ 为 AffB：

（1）对于 ΔG 中的单元更新 δ，其中，（a）δ 是插入边或删除边操作，即 $(w_1,w_2)^+/(w_1,w_2)^-$，并且节点 v 同时属于球 $\hat{G}[w_1,r]$ 和球 $\hat{G}[w_2,r]$。或（b）δ 是插入节点或删除节点操作，即 $(w,(w,w'))^+/(w)^-$，并且 v 属于球 $\hat{G}[w,r]$；或者

（2）球 $\hat{G}[v,r]$ 在 FBM 中的类型代码为 $(1,\ldots,1)$。 □

本文称满足上述条件（1）的球为受更新影响的结构变化球。由于 ΔG 中的一些更新操作，导致这类球的结构发生了变化。

定理 4.9: 给定模式图 Q，数据图 G 和数据图更新 ΔG，如果更新后数据图 $G \oplus \Delta G$ 中的球 $\widehat{G \oplus \Delta G}[v,r]$ 中存在 Q 的完美匹配子图，则球 $\hat{G}[v,r]$ 肯定是由计算过程 IdABall 标识出的受更新影响球。 □

计算过程 IncMatch 在处理模式图更新维护 $\tilde{M}(Q,G)$ 时可以调用增量计算过程。然而，IncMatch 在处理数据图更新时，对于 $\tilde{M}(Q,G)$ 中每个模式图区块在结构变化球（引理 4.2 条件（1））中的部分匹配关系，IncMatch 需要重新计算这些部分匹配关系。对于满足引理 4.2 条件（2）的 AffBs 则不需要花费任何计算维护 $\tilde{M}(Q,G)$。计算过程 combine 按照与处理 ΔQ 相同的方法处理 ΔG，将所有部分匹配关系合并得到最终查询结果。

更新 FBM。算法 dynamicG 为所有 AffBs 更新辅助数据结构 FBM。除了与处理模式图更新一样的部分之外，即更新 FBM 中 FS 指向 BS 的链接，dynamicG 还需要按照以下方式维护 BS：（a）由于删除节点操作导致一些球结构的球心从 G 中删除，需要删除该球在 BS 中的相应条目；由于插入节点操作导致 G 中新增球结构，需要为该球在 BS 中相应插入新的条目。以及（b）根据 ΔG 为 AffBs 中所有球更新其在 BS 中的den 属性。以上所有更新操作可以在 $O(|\text{AffBs}|)$ 时间内执行完毕。

例 4.12: 给定模式图 Q_1 和数据图 G_1（均不包括虚线边），如图 13 所示，其辅助数据结构 FBM 如图 15 所示。当数据图产生更新 $\Delta G_1 = (\text{SD}_3, \text{ST}_3)^+$ 时，根据引理 4.2，IdABall 将球 $\hat{G}[\text{SD}_3,2]$、$\hat{G}[\text{ST}_3,2]$、$\hat{G}[\text{SA}_3,2]$ 和 $\hat{G}[\text{PM}_2,2]$ 标识为结构变化球，并且通过访问 FBM 得到类型代码为 $tc\ (1,1)$ 的球，即 $\hat{G}[\text{PM}_1,2]$。IdABall 将以上所有球结构标记为 AffBs，同

时过滤掉所有其他球结构。 □

数据图更新增量算法 dynamicG。给定辅助数据结构 $\tilde{M}(Q,G)$ 和 FBM。对于数据图更新 ΔG，算法 dynamicG 通过依次调用计算过程 IdABall，IncMatch 和 combine 计算模式图 Q 在更新后的数据图 $G \oplus \Delta G$ 中的 top-k 完美匹配子图，并同时更新和维护辅助数据结构。

算法正确性和复杂性分析。算法 dynamicG 的正确性由计算过程 IdABall（根据定理 4.9）、计算过程 IncMatch 和计算过程 combine 的正确性保证。其中，计算过程 IncMatch 和 combine 在处理数据图更新时的正确性可以用与处理模式图更新时相同的方法来证明。

计算过程 IdABall 的时间复杂度为 $O(|\text{AffBs}| + |\Delta G|)$；计算过程 IncMatch 为所有模式图区块在 AffBs 的所有球结构中更新部分匹配关系的时间花费为 $O(|Q||\text{AffBs}|)$；计算过程 combine 的时间复杂度为 $O(\bigcup_{i\in[1,h]} r|M(Q_{fi}, \widehat{G \oplus \Delta G})| + r|V_Q||E_{\widehat{G\oplus\Delta G}}|)$。由上可知，算法 dynamicG 处理 ΔG 的时间复杂度为 $O(\bigcup_{\widehat{G\oplus\Delta G}\in\text{AffBs}} \bigcup_{i\in[1,h]} r|M(Q_{fi}, \widehat{G \oplus \Delta G})|+r|Q||\text{AffBs}| + |\Delta G|)$。其中，$r$ 通常很小，取值一般为 2 或 3。

4.6.4 统一增量查询算法

本文在章节 4.5.2 设计了图匹配查询的通用增量计算框架。章节 4.6.2 根据增量计算框架设计了用于处理模式图更新的增量算法 dynamicP; 章节 4.6.3 根据增量计算框架设计了用于处理数据图更新的增量算法 dynamicG。本章通过将增量算法 dynamicP 和 dynamicG 合并，设计统一的增量算法 dynamic，用于处理单独或混合的模式图和数据图的连续更新。

增量算法 dynamic 能够处理混合的 ΔQ 和 ΔG，因为算法 dynamicP 和 dynamicG 在以下几处具有一致性：（1）处理 ΔQ 和 ΔG 的工作流程遵循相同的计算框架（章节 4.5.2）；（2）处理 ΔQ 和 ΔG 使用相同的辅助数据结构；以及（3）算法 dynamicP 和 dynamicG 中三个计算过程分别合并后的计算过程足以支持混合的模式图和数据图更新。

分析可知，因为 dynamic 在整个计算过程中增量维护所有辅助数据结构，所以使得 dynamic 能够处理连续产生的模式图和数据图的混合更新。

备注。可以看出算法 dynamicP 的运行时间取决于参数 $\{Q, \Delta Q, \tilde{M}(Q,G), \text{AffBs}\}$，算法 dynamicG 的运行时间取决于参数 $\{Q, \Delta G, \tilde{M}(Q,G), \text{AffBs}\}$，均不直接取决于 G。所以，定理 4.6 的正确性得证。

4.7　实验评估

本章采用两个真实数据集和一个合成数据集，设计了以下四组实验评估本文设计算法的性能：（1）top-k 图匹配查询算法 batch 的性能；（2）动态 top-k 图匹配查询算法 dynamic 处理单轮（a）模式图更新，（b）数据图更新，和（c）混合的模式图和数据图更新的效率；（3）算法 dynamic 处理多轮连续模式图和数据图更新的效率；以及（4）算法 dynamic 使用的辅助数据结构的物理存储空间测试。

4.7.1　实验设置

本章首先介绍实验设置和数据集等实验相关信息。

数据图。本章采用两个真实数据集和一个合成数据集来评估方法的性能。
（1）论文引用网络图（Citation ）是从 DBLP、ACM 和 MAG 等网站上获取的论文之间的引用关系网络图 [100]。其包含 1.39M 个论文节点，3.02M 条论文之间的引用边。本文使用其无向图版本，并根据论文标题的短语聚类为论文引用网络图生成 200 个标签。
（2）YouTube 视频网络图（YouTube ）是从视频网站 YouTube 上获取的表示视频之间推荐关系的网络图 [76]。其包含 2.03M 个视频节点，12.2M 条视频之间的推荐边。本文使用其无向图版本，并根据视频类别及上传时间为视频网络图生成 398 个标签。
（3）合成数据图（Synthetic ）是基于 LFR 基准图模型生成的具有现实生活中社区结构的合成数据图 [101]。其由三个参数控制：节点数量 n，节点平均度 d 和节点标签的数量 l。

模式图。本章通过实现一个模式图自动生成器来生成模式图。自动生成器由 4 个参数控制，从而生成不同的模式图：$|V_Q|$ 控制模式图的节点个数，$|E_Q|$ 控制模式图边的个数，标签函数 l_Q 从实验用数据图的字表 Σ 中为每个节点选取一个标签。容量函数 f_Q 为每个节点指定一个容量区间。

算法。本章用 C++ 实现了以下算法：（1）基于组合模拟的 top-k 图匹配查询算法 batch，（2）基于组合模拟的动态 top-k 图匹配查询算法 dynamic，（3）真实应用团队构建问题的三个对比算法：（a）通过最小化查询结果直径的 minDia 算法 [89]，该算法是团队构建问题的最初始解决方法；（b）通过最小化查询结果中所有节点对之间距离之和的 minSum 算法 [92]；（c）通过最大化查询结果密度的 denAlk 算法 [90]，该算法与本文提出的算法有相同的优化目标。团队构建问题的大部分算法（包括 minDia 和 denAlk）只计算出最优的一个查询结果，而 minSum 通过使用 Lawler 程序 [102] 从而能够在多项式时间内找到 top-k 查询结果。本文通过使用 Lawler 程序扩展算法 minDia 和 denAlk 使其也可以在

多项式时间内找到 top-k 查询结果。

所有实验都是在一台配置为 Intel Core i5-4570 CPU and 16GB of memory 的机器上测试。本章为所有的实验设定默认参数为 $k = 10$，$r = 2$，$h = 3$，$(|V_Q|, |E_Q|)$ 为 (10,12)，节点容量区间为 [1,10]。当生成合成数据图时，设定参数为 $n = 10^7$，$d = 10$，$l = 200$。每组实验生成 3 组不同输入并重复 5 次，取平均数记录于以下实验结果章节中。

4.7.2 Batch 算法测试

本章设计两组实验评估算法 batch 的性能。

(1) Batch 算法效率测试。本节评估了算法 batch，minDia，minSum 和 denAlk 的效率。本文为算法 batch 生成了模式图，为算法 minDia，minSum 和 denAlk 生成了相应的标签查询。

算法 minDia，minSum 和 denAlk 在大规模数据图上的查询效率极低，事实上，(i) 算法 minDia 和 minSum 运行了超过 8 小时时间来完成预处理，即计算全源最短路径；(ii) 即使当 $k = 1$ 最简单情况，算法 denAlk 在 Citation 上依然运行了超过 24 小时。相比之下，算法 batch 在默认参数设置下在 Citation 上仅运行约 100 秒。因此，本章仅在 Citation 中和 YouTube 中由 10,000 个节点构成的具有代表性的采样数据图上测试这些算法的有效性。

(2) Batch 算法有效性测试。本节通过检查算法 batch，minDia，minSum 和 denAlk 输出的查询结果质量来评估四个算法在实际应用中的有效性。

本文从实用性角度定义了四个衡量查询结果质量的评价标准，用于评估以上四个算法输出查询结果的质量。给定匹配子图 G_S 和模式图 $Q(V_Q, E_Q)$。

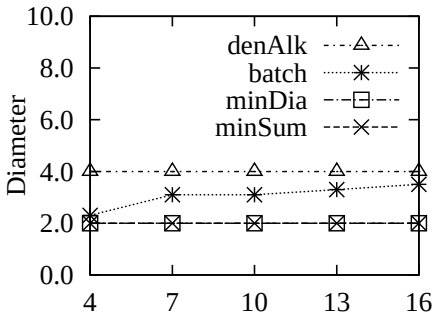
(a) [直径]：G_S 的直径。

(b) [密度]：G_S 的密度。

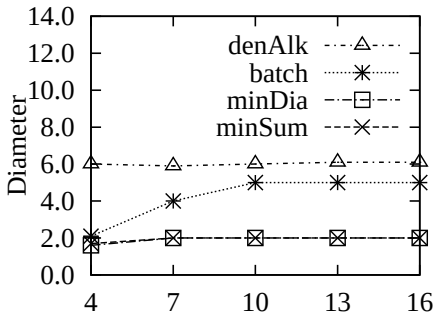
(c) [节点可满足性]：$\eta_V(G_S, Q) = \#\text{sat}_V(G_S, Q)/|V_Q|$。其中，$\#\text{sat}_V(G_S, Q)$ 是 Q 中节点标签和容量要求均被 G_S 中匹配节点满足的节点数目。如果 G_S 中存在一个节点集合 V_u 与 Q 中节点 u 匹配，并且 V_u 满足 u 的容量区间限制，则称 G_S 满足模式图 Q 的节点 u。

(d) [边可满足性]：$\eta_E(G_S, Q) = \#\text{sat}_E(G_S, Q)/|E_Q|$。其中，$\#\text{sat}_E(G_S, Q)$ 是 Q 中边被 G_S 中匹配节点之间的边满足的边的数目。对于 Q 中边 (u_1, u_2)，如果对于 G_S 中每个与 u_1 匹配的节点 v_1，都存在一条边 (v_1, v_1')，使得 v_1' 与 u_2 匹配；并且对于 G_S 中每个与 u_2 匹配的节点 v_2，都存在一条边 (v_2, v_2') 使得 v_2' 与 u_1 匹配。则称 G_S 满足模式图 Q 的边 (u_1, u_2)。

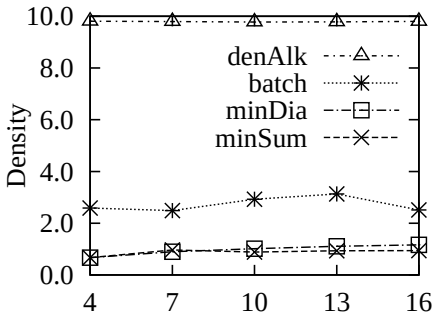
值得注意的是 (a) 和 (b) 是现有团队构建问题算法使用的两种传统质量测量方法 [89–92]。直观理解，$\eta_V(G_S, Q)$ 衡量 G_S 满足 Q 中节点标签和容量要求的比例，$\eta_E(G_S, Q)$



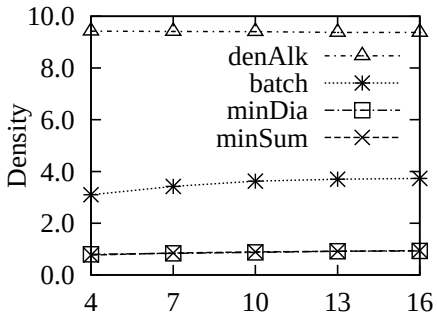
(a) Citation: 变化 $|V_Q|$



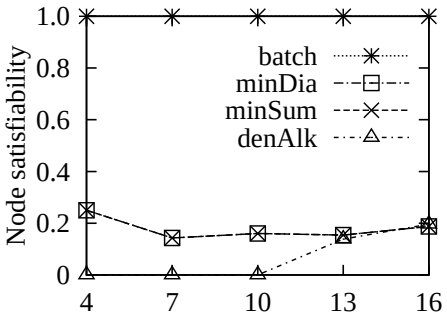
(b) YouTube: 变化 $|V_Q|$



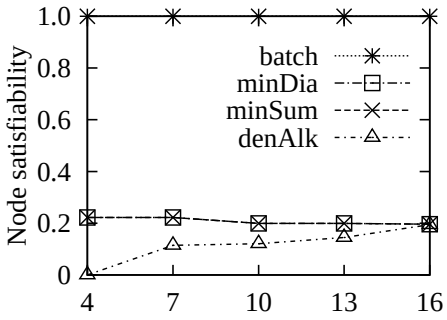
(c) Citation: 变化 $|V_Q|$



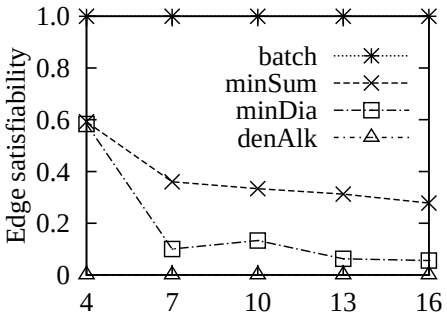
(d) YouTube: 变化 $|V_Q|$



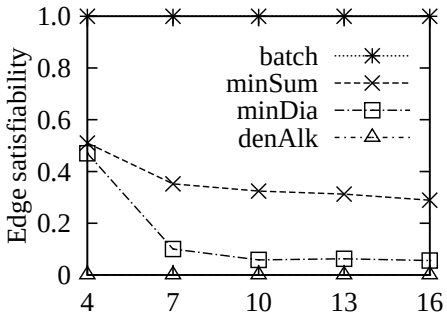
(e) Citation: 变化 $|V_Q|$



(f) YouTube: 变化 $|V_Q|$



(g) Citation: 变化 $|V_Q|$



(h) YouTube: 变化 $|V_Q|$

图 17　Batch 算法有效性测试：变化 $|V_Q|$

衡量 G_S 满足 Q 中结构要求的比例，两个参数的取值范围均为 $[0,1]$。

（a）$|V_Q|$ 对查询结果的影响。通过改变 Q 中节点大小 $|V_Q|$，使其从 4 逐步变化至 16，使用四个质量衡量标准分别在数据图 Citation 和 YouTube 上评估由算法 batch，minDia，minSum 和 denAlk 输出的查询结果的质量。数据图 Citation 上的实验结果如图 17a，17c，17e，17g 所示，数据图 YouTube 上的实验结果如图 17b，17d，17f，17h 所示。

实验发现如下所示：（i）batch 的查询结果直径与 minDia 和 minSum 的查询结果直径相近，而后者是特别设计用于最小化查询结果直径的算法，如图 17a，17b 所示。batch 通过使用球结构来最小化查询结果直径。（ii）batch 找到的查询结果密度小于特别设计用于最大化查询结果密度的 denAlk 算法，但是大于 minDia 和 minSum 找到的查询结果密度，如图 17c，17d 所示。（iii）batch 找到的查询结果节点可满足性远高于 minDia、minSum 和 denAlk，如图 17e，17f 所示。batch 找到的查询结果节点可满足性在所有情况下都为 1.0，而 minDia、minSum 和 denAlk 在所有情况下都不大于 0.2。（vi）batch 找到的查询结果具有更高的边可满足性，如图 17g，17h 所示。batch 找到的查询结果边可满足性在所有情况下都为 1.0，而 minDia、minSum 和 denAlk 在所有情况下都不大于 0.6。

（b）k 对查询结果的影响。通过改变 k 的大小，使其从 1 逐步变化至 20。数据图 Citation 上的实验结果如图 18a，18c，18e，18g 所示，数据图 YouTube 上的实验结果如图 18b，18d，18f，18h 所示。观察可知，四个算法找到的查询结果质量的变化趋势与变化 $|V_Q|$ 时的趋势相同。并且可以观察到查询结果质量对参数 k 的变化不敏感，这是 top-k 语义的理想性质。

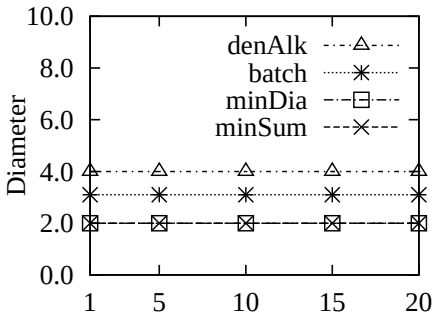
以上实验用四个查询结果实用性评价标准验证了算法 batch 在实际应用中的有效性。其可以找到最优 k 个满足实际需求的查询结果。

4.7.3 Dynamic 算法测试

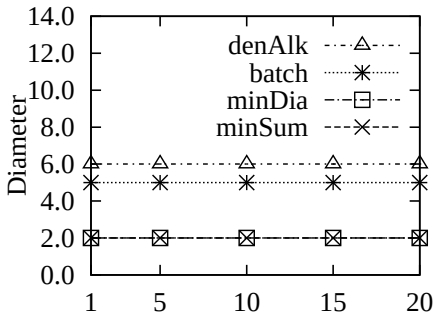
本节设计三组实验评估算法 dynamic 的性能。

（1）Dynamic 算法处理单轮更新效率测试。本节分别在数据图 Citation、YouTube 和 Synthetic 上测试动态 top-k 图匹配查询算法 dynamic 处理单轮次模式图更新，数据图更新和混合的模式图和数据图更新的效率，并与批处理算法 batch 进行对比。

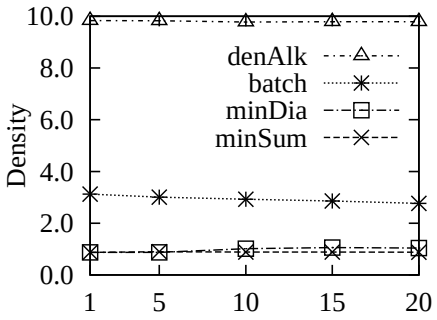
（a）模式图更新。通过将初始模式图 Q 的大小 $(|V_Q|,|E_Q|)$ 固定为 $(10,12)$，并将 ΔQ 中的单位更新个数 $|\Delta Q|$ 由 1 逐步变化至 11，对应于 4.5% 至 49.5% 的模式图更新。本节分别



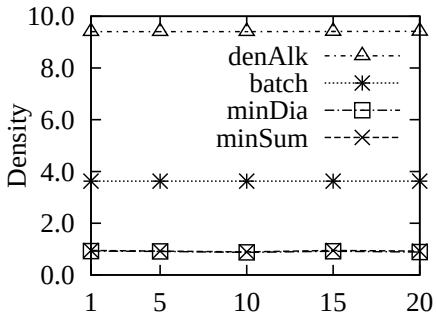
(a) Citation: 变化 k



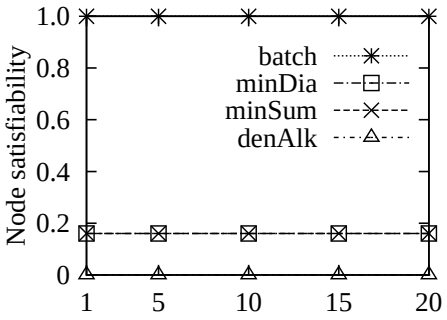
(b) YouTube: 变化 k



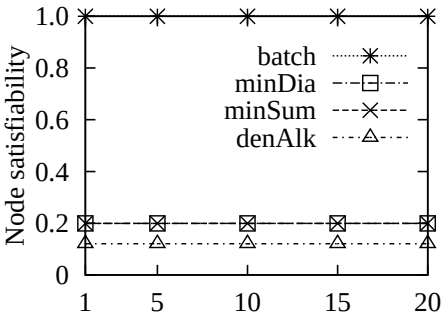
(c) Citation: 变化 k



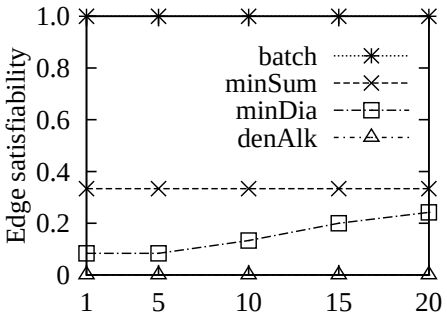
(d) YouTube: 变化 k



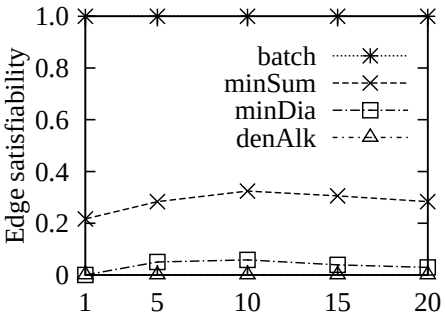
(e) Citation: 变化 k



(f) YouTube: 变化 k

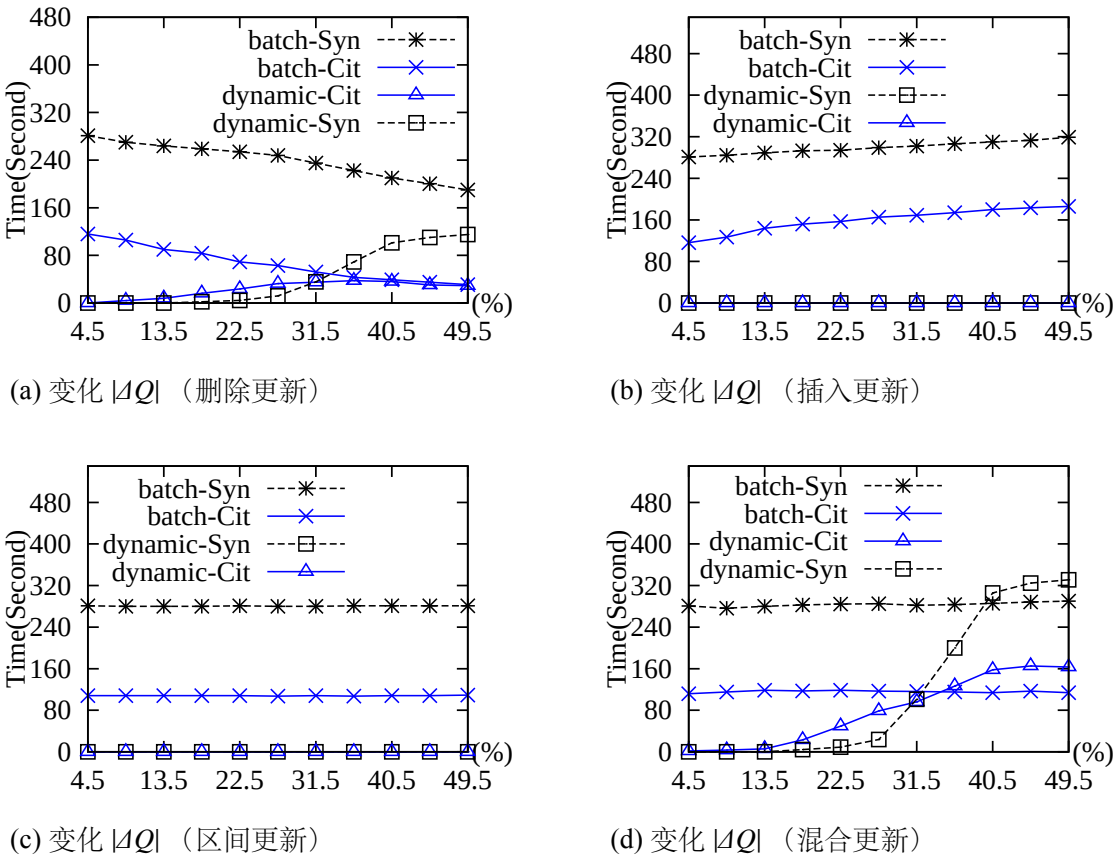


(g) Citation: 变化 k



(h) YouTube: 变化 k

图 18　Batch 算法有效性测试：变化 k



(a) 变化 $|\Delta Q|$（删除更新）　(b) 变化 $|\Delta Q|$（插入更新）

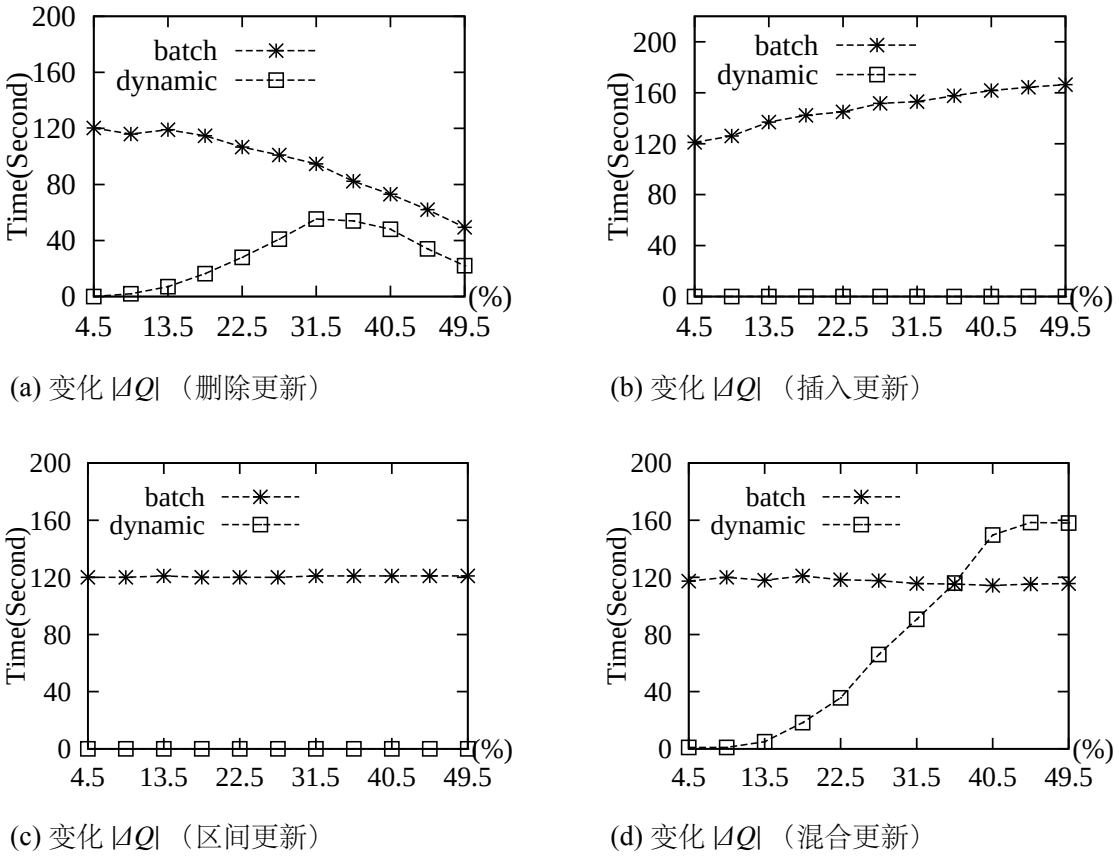
(c) 变化 $|\Delta Q|$（区间更新）　(d) 变化 $|\Delta Q|$（混合更新）

图 19　Dynamic **算法处理单轮模式图更新效率测试：变化** $|\Delta Q|$（Citation，Synthetic）

测试当 ΔQ 仅包含删除（边和节点）更新操作，插入（边和节点）更新操作，区间更新操作，和混合更新操作（5 种更新类型保持比例相等）时，算法 dynamic 的更新效率。数据图 Citation 和 Synthetic 上的实验结果如图 19a，19b，19c，19d 所示，数据图 YouTube 上的实验结果如图 20a，20b，20c，20d 所示。

实验发现如下所示：（i）即使当 Citation 上的删除更新达到 40.5%，Synthetic 上的删除更新达到 49.5%，YouTube 上的删除更新达到 49.5% 时，增量算法相对批处理算法查询效率都有提升。由于提前返回优化技术，dynamic 始终优于 batch。（ii）dynamic 在仅处理插入更新和区间更新操作时效率相对 batch 有大幅度提升。（iii）对于相同的 $|\Delta Q|$，dynamic 处理插入更新的时间少于处理删除更新的时间。（vi）当处理混合更新时，即使当 Citation 上的混合更新达到 31.5%，Synthetic 上的混合更新达到 40.5%，YouTube 上的混合更新达到 36% 时，增量算法相对批处理算法查询效率都有提升。这是因为当模式图更新积累到一定程度时，所有球都被标识为 AffBs。

（*b*）数据图更新。数据图上的删除（边和节点）更新和插入（边和节点）更新的生成方



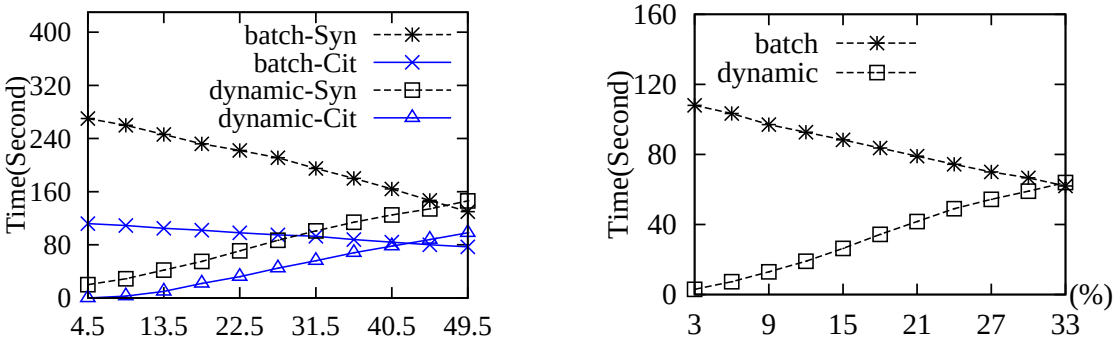
(a) 变化 $|\Delta Q|$（删除更新）　(b) 变化 $|\Delta Q|$（插入更新）

(c) 变化 $|\Delta Q|$（区间更新）　(d) 变化 $|\Delta Q|$（混合更新）

图 20　Dynamic 算法处理单轮模式图更新效率测试：变化 $|\Delta Q|$（YouTube）

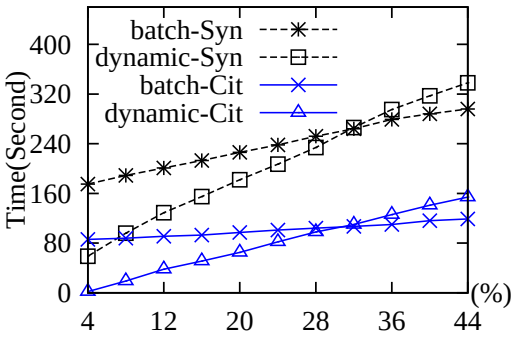
法如下。例如，数据图 Citation 大小为 $|G| = 4.4M$，对于删除更新，通过从 G 中随机选择节点和边的子集并逐步将其从 G 中删除，从而将 $|G|$ 从 $4.4M$ 每次减少 4.5% 的节点和边逐步变化至 $2.22M$。对于插入更新，首先从 G 中随机选择一个节点和边的子集并将其从 G 中删除，得到初始数据图 G。再逐步将这些节点和边插入 G，从而将 $|G|$ 从 $3.05M$ 每次增加 4% 的节点和边逐步变化至 $4.4M$。对于混合数据图更新（4 种类型），首先从 G 中随机抽样一个子图 G_s 并将其从 G 中删除，得到初始数据图 G。再从 G 中随机选择节点和边的子集并将其从 G 中删除，而后从 G_s 中随机选择节点和边的子集并将其加入 G 中，从而生成 G 上 4% 的混合更新，重复此步骤即可得到多次混合更新。数据图 Citation 和 Synthetic 上的实验结果如图 21a，21c，21e 所示，数据图 YouTube 上的实验结果如图 21b，21d，21f 所示。

实验发现如下所示：（i）即使当 Citation 上的删除更新达到 40.5%，Synthetic 上的删除更新达到 45%，YouTube 上的删除更新达到 33% 时，增量算法相对批处理算法查询效率都有提升。（ii）即使当 Citation 上的插入更新达到 28%，Synthetic 上的插入更新

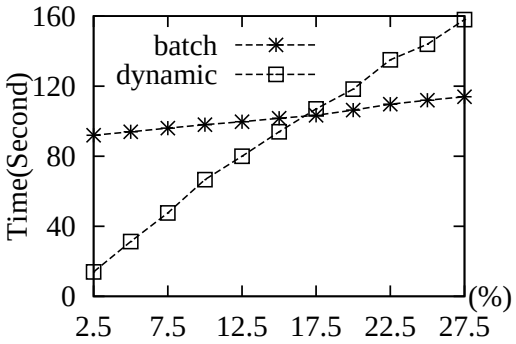


(a) 变化 $|\Delta G|$（删除更新）

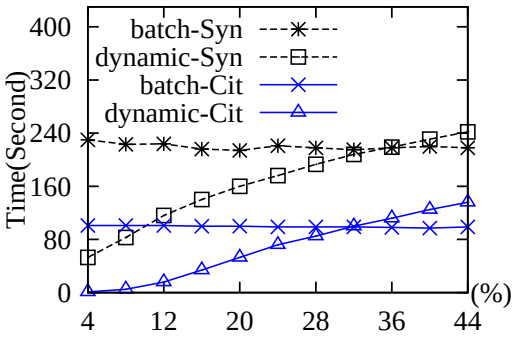
(b) 变化 $|\Delta G|$（删除更新）



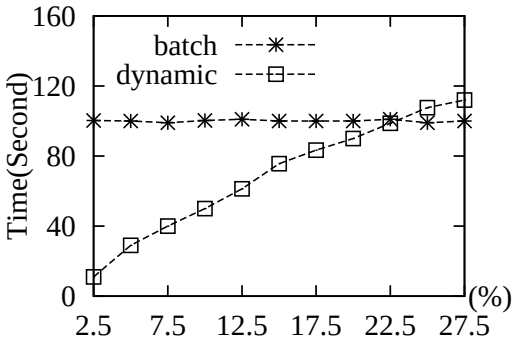
(c) 变化 $|\Delta G|$（插入更新）



(d) 变化 $|\Delta G|$（插入更新）



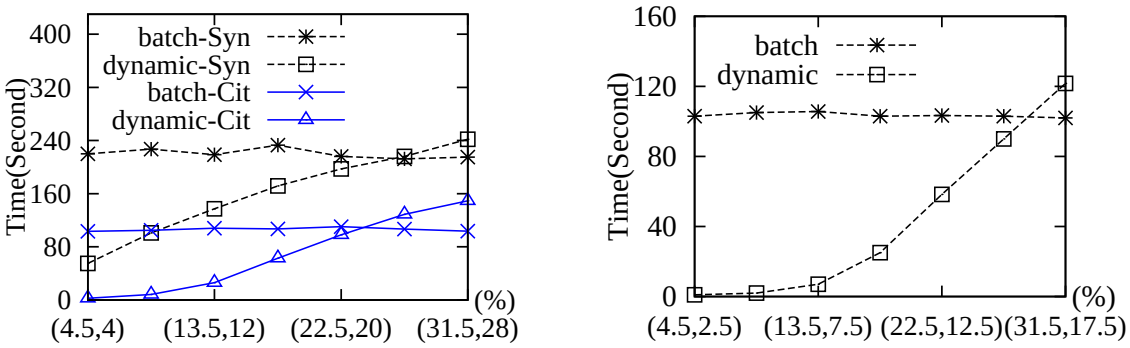
(e) 变化 $|\Delta G|$（混合更新）



(f) 变化 $|\Delta G|$（混合更新）

图 21　Dynamic **算法单轮数据图更新效率：**Citation，Synthetic：(a), (c), (e); YouTube：(b), (d), (f)

达到 32%，YouTube 上的插入更新达到 15% 时，增量算法相对批处理算法查询效率都有提升。（iii）对于相同的 $|\Delta G|$，dynamic 处理删除更新的时间少于处理插入更新的时间。（vi）本文对社交网络应用数据做了调研：Facebook 每日新增用户达到 1.23‰ [2]，Twitter 每日新增用户达到 2.47‰ [103]。因此，dynamic 能够增量高效的处理 Facebook 和 Twitter 上累积数十天的数据。（v）即使当 Citation 上的混合更新达到 32%，Synthetic 上的混合更新达到 36%，YouTube 上的混合更新达到 22.5% 时，增量算法相对批处理算法查询



(a) Citation, Synthetic　　(b) YouTube

图 22　Dynamic 算法处理单轮模式图和数据图混合更新效率测试：变化 $(\Delta Q, \Delta G)$

效率都有提升。

（*c*）模式图和数据图混合更新。使用与单独模式图更新和单独数据图更新相同的设置同时改变模式图和数据图，将 $(\Delta Q, \Delta G)$ 由 (4.5%, 4%) 逐步变化至 (31.5%, 28%)，数据图 Citation 和 Synthetic 上的实验结果如图 22a 所示，数据图 YouTube 上的实验结果如图 22b 所示。实验发现即使当 Citation 上的混合更新达到 (22.5%, 20%)，Synthetic 上的混合更新达到 (27%, 24%)，YouTube 上的混合更新达到 (27%, 15%) 时，增量算法相对批处理算法查询效率都有提升。

（2）Dynamic 算法处理多轮连续更新效率测试。本节分别在数据图 Citation、YouTube 和 Synthetic 上测试动态 top-k 图匹配查询算法 dynamic 处理多轮连续模式图和数据图更新的效率，并与批处理算法 batch 进行对比。

（*a*）模式图更新。本节生成连续 5 个轮次的模式图混合更新，并测试 dynamic 逐个处理完成 5 个轮次更新所花费的平均时间。通过将每个轮次的模式图更新操作个数由 1 逐步变化至 7，对应于 4.5% 至 31.5% 的模式图更新，生成 7 组实验输入。数据图 Citation 和 Synthetic 上的实验结果如图 23a 所示，数据图 YouTube 上的实验结果如图 23b 所示。

回顾算法 dynamic 设计可知其采用延迟更新策略。虽然延迟更新策略会减少当前轮次的更新处理时间，但是必然会影响下轮次的更新处理时间。然而实验结果显示，即使当 Citation 上每轮模式图更新达到 27%，Synthetic 上每轮模式图更新达到 31.5%，YouTube 上每轮模式图更新达到 27% 时，增量算法相对批处理算法查询效率都有提升。这验证了延迟更新策略的有效性。

（*b*）数据图更新。本节采用与模式图多轮连续更新相同的实验设置。通过将每个轮次的

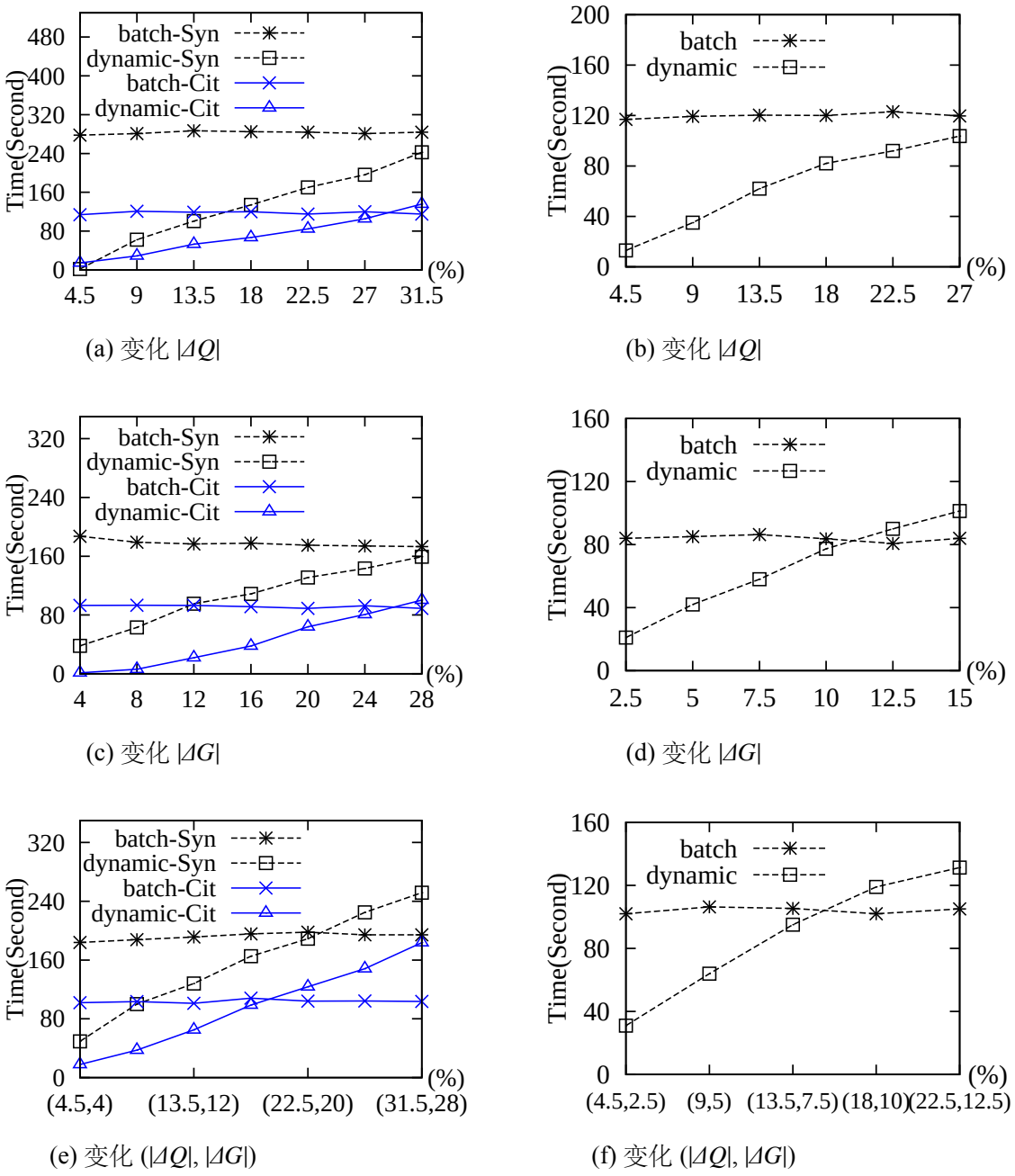


图 23　Dynamic 算法多轮连续更新效率：Citation，Synthetic：(a), (c), (e); YouTube：(b), (d), (f)

数据图更新操作数量由 4% 逐步变化至 28%，生成 7 组实验输入，测试 dynamic 逐个处理完成 5 个轮次更新所花费的平均时间。数据图 Citation 和 Synthetic 上的实验结果如图 23c 所示，数据图 YouTube 上的实验结果如图 23d 所示。实验发现，即使当 Citation 上每轮数据图更新达到 24%，Synthetic 上每轮数据图更新达到 28%，YouTube 上每轮数据图更新达到 10% 时，增量算法相对批处理算法查询效率都有提升。

（*c*）模式图和数据图混合更新。本节采用与单独模式图和单独数据图多轮连续更新相同

的实验设置。通过将每个轮次的混合模式图和数据图更新操作数量 $(\Delta Q, \Delta G)$ 由 (4.5%, 4%) 逐步变化至 (31.5%, 28%)，生成 7 组实验输入，测试 dynamic 逐个处理完成 5 个轮次更新所花费的平均时间。数据图 Citation 和 Synthetic 上的实验结果如图 23e 所示，数据图 YouTube 上的实验结果如图 23f 所示。实验发现，即使当 Citation 上每轮混合更新达到 (18%, 16%)，Synthetic 上每轮混合更新达到 (22.5%, 20%)，YouTube 上每轮混合更新达到 (13.5%, 7.5%) 时，增量算法相对批处理算法查询效率都有提升。

(3) 辅助数据结构物理存储空间测试。数据图 Citation 和 Synthetic 上的实验结果如图 24a 所示，数据图 YouTube 上的实验结果如图 24b 所示。数据图 Citation、Synthetic 和 YouTube 所占存储空间分别为 36MB、482MB 和 114MB，dynamic 分别需要 25MB、130MB 和 58MB 额外空间来存储所有辅助数据结构。也就是说，dynamic 设计的辅助数据结构是轻量级的数据结构，仅占原始数据图 69.4%、26.9% 和 50.8% 的存储空间。

4.7.4 实验总结

实验结果表明，（1）本文提出的基于组合模拟的 top-k 图匹配查询语义可以有效的表达实际应用的查询需求。（2）本文提出的基于组合模拟的 top-k 图匹配查询批处理算法效率较高，例如当数据图规模为 $|V_G| = 1.39M$，模式图规模为 $|V_Q| = 10$ 时，批处理算法只需 116s 即可完成查询。（3）本文提出的基于组合模拟的动态 top-k 图匹配查询增量算法能够处理连续的单独或混合的模式图和数据图更新，查询效率相对批处理算法有显著提升。即使（a）当模式图更新平均达到 36%，数据图更新平均达到 0.2%，混合模式图和数据图更新平均达到 (25.5%, 20%) 时，以及（b）当连续模式图更新平均达到 28.5%，连续数据图更新平均达到 20.7%，连续混合模式图和数据图更新平均达到 (18%, 14.5%) 时，增量算法相对批处理算法查询效率都有提升。

4.8 相关工作分析

本课题的相关工作有以下几方面。

图匹配查询语义。本文在章节 2.1 中详细介绍了图匹配查询三种主要匹配语义，包括子图同构 [11, 12]，图模拟 [13, 28] 及其扩展强模拟 [14, 15] 匹配查询语义。另外，图模拟语义还存在很多扩展语义。例如 [6] 提出基于图模拟的有界模拟图匹配语义，通过为模式图的边增加正则表达式从而使得匹配图中包括种类不同的匹配边；[93] 通过为模式图的节点增加匹配节点容量要求从而使得匹配图中包括指定个数的匹配节点。其中，[21] 将有界

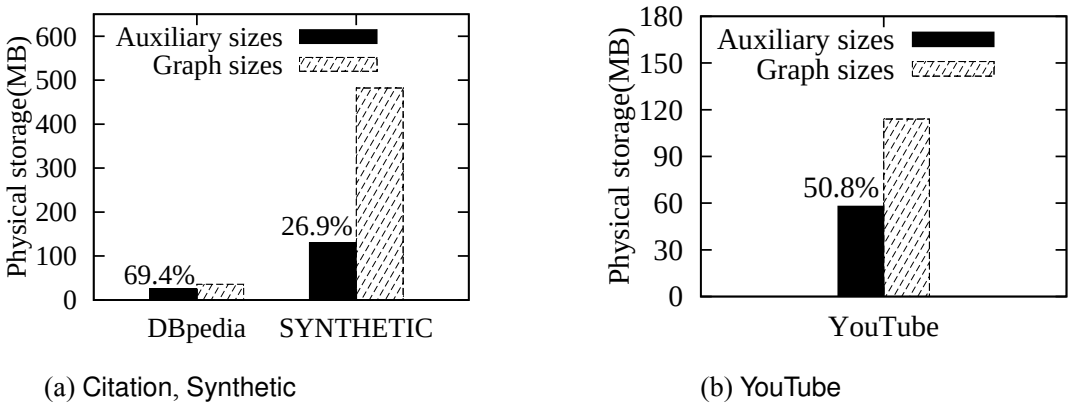


图 24　Dynamic 算法使用辅助数据结构物理存储空间测试

模拟匹配语义应用于专家推荐应用中，用于从模式图的匹配图中找到 top-k 个匹配节点。然而，以上所述图匹配查询语义在表达实际应用需求方面都存在不足之处。本课题提出的基于组合模拟的 top-k 图匹配查询语义与以上工作的不同在于：（1）该语义可以从数据图中找到与模式图匹配的 top-k 个匹配子图，并且（2）该语义可以有效表达实际应用中的查询需求，包括匹配节点个数、匹配结果半径、匹配结果密度等实际需求。

团队构建算法。团队构建问题是社交网络上典型的实际应用问题，相关研究为团队构建问题提出了很多解决方法 [89–92]。这些方法通过优化查询结果的直径、密度、最小生成树以及所有成员之间距离之和来最小化团队成员之间的沟通成本，查找目标团队。然而，这些解决方法存在以下问题：（1）上述所有团队构建方法其本质是图数据上的关键词搜索方法。基于关键词搜索的团队构建方法只能表达出团队的职能需求，无法表达出团队的结构需求，而满足团队结构需求是团队可以高效合作的关键和保证。（2）[92] 通过使用 Lawler 程序 [102] 从而找到 top-k 查询结果。然而经过实验验证，基于 Lawler 程序的 top-k 查询方法在大规模数据图上的查询效率较低。（3）团队构建算法中，[89] 通过最小化查询结果直径查找目标团队，[90] 通过最大化查询结果密度查找目标团队，这两个优化目标在所有团队构建算法中显示出了良好的性能。本课题提出具有实用价值的 top-k 图匹配查询语义，用于解决包括团队构建问题等实际应用问题。该语义可以有效表达实际应用中的查询需求，包括匹配结果的结构需求、匹配节点个数、匹配结果半径、匹配结果密度以及高效输出 top-k 匹配结果等需求。

图匹配查询增量查询算法。与本课题最相关的工作是由 [50, 51] 提出的图匹配查询的动态数据图增量查询算法。本文在章节 2.3.2 中详细介绍了这项工作。本课题与这项工作有以下几处不同：（1）现有工作只考虑数据图更新的增量算法，本文考虑连续混合的模

式图更新和数据图更新。（2）本课题提出了统一的图匹配查询增量计算框架，该框架可以处理连续的单独或混合的模式图和数据图更新。这是对模式图更新的第一次研究，也是迄今为止所考虑的最广泛和最实用的动态设置。（3）本课题提出的图匹配查询增量计算框架是通用方法，可以适用于 top-k 图匹配查询方法和一般图匹配查询方法以及任意一种图匹配查询语义。

Top-k 查询技术。Top-k 查询技术被广泛应用于关系数据查询 [99, 104] 和 XML 数据查询 [105, 106] 等不同数据类型的应用中，使得查询引擎可以从数据集中找出最优 top-k 个查询结果。Fagin 算法 [99] 是具有提前终止性质的 top-k 查询算法。该算法通过从不同的排序列表中读取属性并构造元组，直至找到 top-k 元组，同时提前终止查询程序。[21, 107] 提出将 top-k 查询语义应用于基于图模拟 [13] 及其扩展有界模拟 [6] 的图匹配查询过程中，用于从模式图的匹配图中找到 top-k 个匹配节点。本课题与这项工作有以下两处不同：（1）本课题提出的基于组合模拟的 top-k 图匹配查询语义可以从数据图中找到与模式图匹配的 top-k 个匹配子图。（2）本课题考虑模式图和数据图动态变化环境下的 top-k 图匹配查询问题。

4.9　小结

本文首先提出了组合模拟图匹配语义和基于组合模拟的 top-k 图匹配查询语义。相较于传统图匹配查询语义，该语义更具实用价值，能够表达社交推荐和团队搜索等实际应用的查询需求。本文为基于组合模拟的 top-k 图匹配查询语义设计了批处理算法。本文提出了动态图匹配查询问题，考虑连续混合的模式图更新和数据图更新。本文首先对动态图匹配查询问题的计算复杂性进行了理论分析。尽管由理论分析得知问题的计算复杂度较高，本文设计了基于模式图划分和标识影响区域的增量计算策略，从而将模式图和数据图的更新影响范围最小化和局域化。基于增量计算策略，本文提出统一的图匹配查询增量计算框架，可以处理连续的单独或混合的模式图和数据图的连续更新。该计算框架是通用计算方法，适用于 top-k 图匹配查询和一般图匹配查询方法以及任意一种图匹配查询语义。基于图匹配查询的增量计算框架，本文设计了统一的增量算法来处理单独或混合的模式图和数据图更新。本文通过实验验证了 top-k 图匹配查询批处理算法的有效性，并且图匹配查询增量计算方法相对批处理算法效率有显著提升。

第五章　图匹配查询松弛重构技术

传统的基于子图同构的图匹配查询语义，由于其严格的匹配约束条件经常导致模式图在数据图中的图匹配查询结果较少甚至为空集。针对这一问题，基于图模拟及其扩展强模拟等减弱匹配条件约束的图匹配查询语义被相继提出。另外，相关研究提出通过引入语义分类信息来减弱图匹配查询中标签完全相同的匹配约束，从而得到更多合理的图匹配查询结果。然而，通过实验分析发现，即使采用上述方法图匹配查询结果依然极其稀少，数据图中仍然存在大量合理结果因无法满足图匹配查询的匹配条件而无法被查询得到。而且，对于规模较大并且结构复杂的数据图，非专业用户由于无法全面掌握数据图基本信息，很容易构造出不存在任何查询结果的模式图。本文旨在解决图匹配查询方法在真实数据集难以查询得到合理匹配结果的问题，使得任何用户都能够清晰简单的构造出模式图，并且图匹配查询引擎能够根据用户的查询意图高效的返回用户期望的图匹配查询结果。本文通过将查询松弛重构技术应用于图匹配查询过程中，提出通用的图匹配查询松弛重构计算框架，适用于任意一种图匹配查询语义。本文在论述其工作原理时以图模拟为图匹配查询语义。在这项研究中，本文首先将语义分类信息引入图匹配查询，提出基于语义分类信息的泛化图匹配查询。而后，基于泛化图匹配查询语义，本文提出利用松弛重构技术为模式图生成 top-k 松弛模式图。本文进一步提出松弛图匹配查询多查询处理优化方法通过最大化共享计算量，从而同时、快速的计算得出模式图和 top-k 松弛模式图在数据图中的图匹配查询结果。最后，本文提出松弛图匹配查询结果溯源解释方法为用户提供新结果如何产生的溯源解释。最后，在带有语义信息的真实数据集知识图谱 DBpedia 和知识图谱 YAGO 上，本文通过实验分析证明了图匹配查询松弛重构计算框架能够有效的和高效的识别出数据图中更多有意义的图匹配查询结果。

5.1　引言

传统的基于子图同构的图匹配查询语义，由于其严格的匹配约束条件经常导致模式图在数据图中的图匹配查询结果较少甚至为空集。针对这一问题，相关研究提出通过将语义分类信息引入子图同构匹配语义来减弱图匹配查询中标签完全相同的匹配约束，从而得到更多合理的图匹配查询结果 [108]。其中，语义分类信息指数据图和模式图中节点标签的类别信息。模式图中的一个标签为 l' 的节点可以与数据图中标签为 l 的节点匹配，当且仅当根据语义分类信息，标签 l 是标签 l' 的后代标签。然而，上述方法仍然无法有

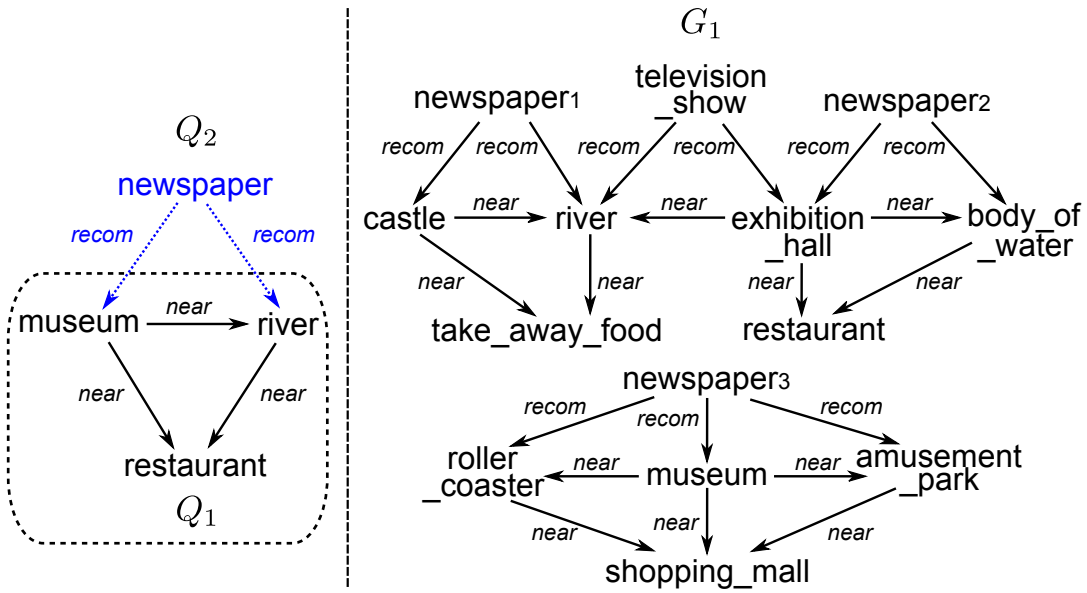


图 25　查询旅游规划网络图

效的识别出数据图中有意义的图匹配查询结果。

例 5.1: 以一个旅游规划网络图 G_1 为例 [109]，如图 25 所示。G_1 中每个节点表示一个实体，节点上的标签表示实体类型，例如河流景点（river）、餐厅（restaurant）。G_1 中的边表示两个实体之间的关系，例如报刊 newspaper$_1$ 推荐过河流景点 river（recom），展览馆 exhibition_hall 与河流景点 river 距离很近（near）。一位游客希望通过使用旅游规划网络图规划他的旅游行程。游客期望他的旅游行程可以（1）参观一个博物馆 museum，（2）游览一个靠近博物馆的河流景点 river，并且（3）能够在靠近这两个景点的餐厅 restaurant 用餐。他将旅游行程需求用一个模式图 Q_1 来表示，如图 25 所示（虚线方格内）。可以验证，Q_1 在 G_1 中不存在基于子图同构语义的图匹配查询结果。

即使采用引入语义分类标签匹配的子图同构语义，仍然无法在 G_1 中找到任何 Q_1 的图匹配查询结果。以知识图谱 DBpedia [75] 中的（部分）语义分类图 T_1 为例，如图 26 所示。由 T_1 可知，（1）展览馆 exhibition_hall 是博物馆 museum 的一种，然而戏剧院 theater 不是 museum 的一种；（2）快餐厅 take_away_food 是餐厅 restaurant 的一种。

尽管如此，由于子图同构严格的结构匹配约束条件，即使引入 T_1 中的语义分类信息，Q_1 在 G_1 中的图匹配查询结果仍然为空。但是，观察 T_1 中的语义分类信息可知，G_1 中由节点 river，exhibition_hall，take_away_food 和 restaurant 构成的子图是能够与 Q_1 匹配的有意义的查询结果。 □

针对这一问题，一个自然的想法是在放松子图同构标签匹配的基础上，进一步放松

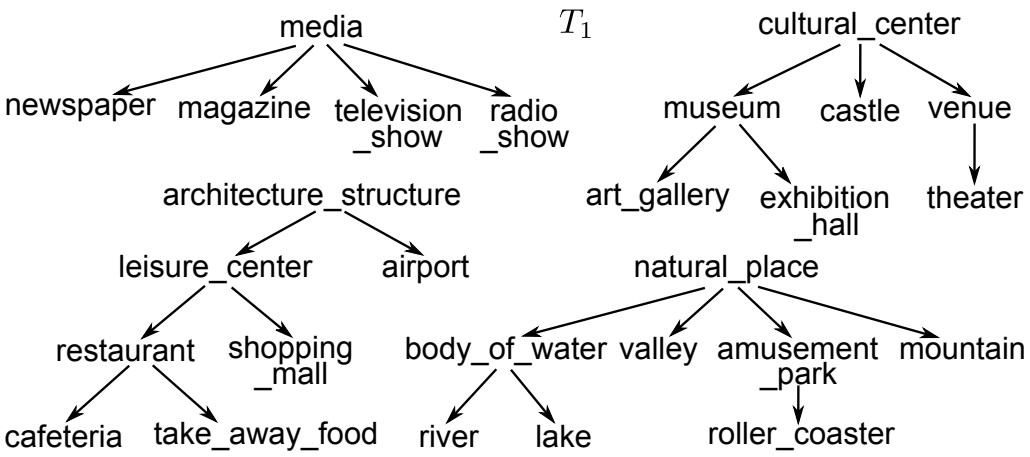


图 26　知识图谱语义分类图

其结构匹配约束条件。回顾章节 2.1 可知，图模拟匹配语义 [13, 28] 将子图同构语义定义中要求模式图与数据图结构对应的双射函数减弱为二元关系对应，将子图同构中要求拓补结构一模一样的约束条件，转变为只保持节点父子关系匹配。所以，本文将语义分类知识信息引入图模拟语义，放松图模拟的标签匹配约束条件。结合语义分类图 T_1 可知，放松标签匹配约束条件实质上是将标签完全相同的匹配要求放松为与“向下方向”标签相同的匹配要求。可以验证，采用基于语义分类标签匹配的图模拟语义可以查询得到例 5.1 最后所述的子图。并且可以验证在任何数据图上，基于语义分类标签匹配图模拟语义的图匹配查询结果一定包括基于图模拟语义、语义分类标签匹配的子图同构语义和子图同构语义的图匹配查询结果。

然而,在真实数据集中,即使采用标签匹配和结构匹配均放松后的基于语义分类标签匹配的图模拟语义，仍然无法有效的识别出有意义的图匹配查询结果，如以下实验所示。

本文利用知识图谱 DBpedia [75] 和 YAGO [110] 的数据图和语义分类图执行下述实验。该实验为两个数据图分别随机生成节点大小在 2 到 10 之间的模式图，并使用数据图中的节点标签集合随机为模式图生成节点标签。采用基于语义分类标签匹配的图模拟语义，测试图匹配查询结果非空的模式图占总体数量的百分比，实验结果如表 15 所示。

由实验可知，即使规模较小的模式图仍然难以在数据图中找到图匹配查询结果。在 DBpedia 中，4 个节点的模式图中有 18% 的模式图可以得到非空图匹配查询结果；在 YAGO 中，4 个节点的模式图中只有 2% 的模式图可以得到非空图匹配查询结果；而对于规模中等及较大的模式图，其图匹配查询结果基本为空集。在 DBpedia 和 YAGO 中，6 个节点及以上的模式图无法在数据图中找到任何图匹配查询结果。对于匹配约束条件

表 15　非空图匹配查询结果模式图所占百分比

$\|V_Q\|$	**2**	**4**	**6**	**8**	**10**
DBpedia	90%	18%	0%	0%	0%
YAGO	54%	2%	0%	0%	0%

更为严格的图模拟语义、基于语义分类标签匹配的子图同构语义和子图同构语义，这些百分比数值更低。然而，通过对实验数据图和模式图分析发现，即使对于图匹配查询结果为空的模式图，数据图中都存在大量未被查询得出的合理匹配结果。

例 5.2: 继续以例 5.1 中 Q_1 和 G_1 为例。如果这位游客期望其旅游行程中将要参观的两个景点 museum 和 river 是由报刊推荐过的优质景点。他通过扩展 Q_1 构建模式图 Q_2 来实现这个查询需求。如图 5.1 所示，Q_2 在 Q_1 的基础上增加了一个 newspaper 节点以及由 newspaper 分别指向 museum 和 river 表示推荐关系的两条边（蓝色虚线边）。采用基于语义分类标签匹配的图模拟语义，Q_2 在 G_1 中不存在任何图匹配查询结果。然而，由图 26 中所示语义分类图 T_1 可知，报刊 newspaper 和电视节目 television_show 都是媒体 media 的一种。根据以上分析可知，由例 5.1 中所述节点集合以及节点 television_show 构成的子图是 Q_2 在 G_1 中的一个合理图匹配查询结果。 □

由 DBpedia 和 YAGO 上的实验以及例 5.2 所示问题可知，即使采用匹配条件最不严格的图匹配查询语义，模式图在数据图中的图匹配查询结果依然大部分为空集。另外，对于规模较大并且结构复杂的数据图，非专业用户由于无法全面掌握数据图基本信息，很容易构造出不存在任何查询结果的模式图。一个用户友好的图匹配查询引擎需要具备易用性。即查询引擎应该使任何用户都能够清晰简单的构造出模式图，并且查询引擎能够根据用户的查询意图高效的返回用户期望的图匹配查询结果。

回顾章节 2.3.4 对查询重构技术的介绍。其中，查询松弛重构技术是用于解决查询过程中查询结果过少问题的技术。该技术通过分析用户输入的查询为用户生成一组查询条件松弛后的查询，供用户选择。另外，根据例 5.2 可知，在语义分类图中通过“向上方向”利用语义分类信息可以为放松模式图匹配条件提供有力支持。例如，根据语义分类图 T_1，模式图 Q_2 中的节点标签 newspaper 可以被“向上方向”放松为 media 从而与数据图 G_1 中标签为 television_show 的节点匹配。本文通过利用语义分类信息，将查询松弛重构技术应用于图匹配查询过程中，对模式图进行松弛重构生成 top-k 松弛模式

图，从而利用松弛模式图在数据图中查询出更多有意义的图匹配查询结果。不同于已有工作通过松弛图匹配查询约束条件提出新的图匹配查询语义来解决图匹配查询结果为空集的问题。本文提出新的图匹配查询“模式”来提高图匹配查询的易用性，使得用户能够查询得到期望的图匹配查询结果。这种查询“模式”是通用的查询方法，可以为任何一种图匹配查询语义生成松弛模式图，从而查询得到更多有意义的图匹配查询结果。

本章主要内容提纲。本文将提出通用的图匹配查询松弛重构计算框架。该计算框架使得图匹配查询引擎能够根据用户输入的模式图自动生成 top-*k* 松弛模式图，并同时计算得出模式图和 top-*k* 松弛模式图在数据图中的图匹配查询结果以及新结果如何产生的溯源解释。本章节研究内容提纲如下所示。

（1） 本文首先将语义分类信息引入图匹配查询过程提出泛化图匹配查询（章节 5.2.1）。通过“向下方向”使用语义分类信息增加图匹配查询结果。而后，基于泛化图匹配查询语义，本文提出通用的图匹配查询松弛重构计算框架，通过“向上方向”使用语义分类信息进一步增加图匹配查询结果（章节 5.2.2）。本文设计了拓补排序函数和多元化拓补排序函数用于生成模式图的 top-*k* 查询松弛。这两个函数综合考虑了泛化图匹配查询特征、语义分类图特征、以及模式图和数据图之间的相关关系特征（章节 5.3）。

（2） 本文对根据拓补排序函数和多元化拓补排序函数生成模式图 top-*k* 查询松弛问题进行了理论分析（章节 5.4）。本文发现（a）基于拓补排序函数的 top-*k* 查询松弛问题是 PTIME 问题；（b）基于多元化拓补排序函数的 top-*k* 查询松弛问题是 NP-完全问题，并且该问题不属于 APX 类可近似问题（APX 类包含具有常数近似度保证的多项式时间近似算法的问题 [73]）。根据理论分析结果，本文通过使用 Lawler 程序 [102] 为问题（a）设计了 PTIME 精确算法。Lawler 程序是用于解决 top-*k* 组合优化问题的重要方法；另外，本文将问题（b）归约至已经被深入研究的 *maximum dispersion*（maxDP）问题，从而可以使用 maxDP 问题的高效算法解决 top-*k* 查询松弛问题。

（3） 本文提出松弛图匹配查询多查询处理优化方法通过最大化共享计算量，从而同时、快速的计算得出 top-*k* 松弛模式图在数据图中的图匹配查询结果（章节 5.5）。本文根据松弛模式图之间的关系构建由 top-*k* 松弛模式图构成的层次结构，并基于层次结构设计泛化图匹配查询有界增量查询方法，从而最大化计算 top-*k* 松弛模式图在数据图上图匹配查询结果过程的共享计算量。

（4） 本文提出松弛图匹配查询结果的最小溯源解释问题，通过捕捉松弛模式图中的关键松弛部分来解释图匹配查询结果为何能够由松弛模式图查询得出（章节 5.6）。通过理论分析发现，（a）基于拓补排序函数的溯源解释问题是 PTIME 问题；（b）基于多元化拓补排序函数的最小溯源解释问题是 NP-完全问题。根据理论分析结果，本文为问题（a）设计了线性时间复杂度最优精确算法；另外，本文为问题（b）设计了精确度由时间复杂度决定的参数化算法，可以在算法输出结果精确度和算法时间复杂度之间做任意平衡。

（5） 本文采用知识图谱 DBpedia 和知识图谱 YAGO 两个真实数据集来评估方法性能，知识图谱的数据图包含百万级节点数据和千万级边数据（章节 5.7）。经实验验证得知（a）模式图的 top-k 查询松弛是有效的：松弛图匹配查询方法能够查询得到 11.9 倍于普通图匹配查询方法的查询结果，并且其中 74% 的结果被证明是有意义的查询结果；（b）当 k=15 时，松弛图匹配查询多查询处理优化方法比传统图匹配查询处理方法快 1.8 倍，并且效率提升幅度随 k 增大而增加；（c）松弛图匹配查询结果溯源解释方法的准确率在不访问数据图的前提下可以达到 85%，在允许 2 次访问数据图的前提下可以达到 99%。

为了论述简洁，本文只将部分定理证明列于正文中，其他定理证明详见附录 C。

5.2 图匹配查询松弛重构

本章首先将语义分类信息引入图匹配查询过程提出泛化图匹配查询（章节 5.2.1）。基于泛化图匹配查询语义，本章提出通用的图匹配查询松弛重构计算框架（章节 5.2.2）。

本文在介绍图匹配查询松弛重构计算框架时以图模拟语义为图匹配查询语义。

5.2.1 基于语义分类的泛化图匹配查询

本文在章节 2.1 对模式图、数据图、图匹配查询进行了详细介绍。本章首先对这些基本概念进行回顾，而后将语义分类信息引入图模拟匹配语义，提出泛化图模拟匹配语义。

定义 5.1 (模式图和数据图): 模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$ 是分别由节点集和边集构成的有向图。节点上有表示实体有关信息的标签，边上有表示实体间关系信息的标签。例如，数据图 G 的节点集合为 V_G；边集为 E_G，并且 $E_G \subseteq V_G \times V_G$；$l_G$ 为标签函数将 V_G 中任意一个节点 v 映射至节点标签集合 Σ_V 中的一个标签，并且将 E_G 中任意一条边 (v, v') 映射至边标签集合 Σ_E 中的一个标签。另外，在上下文已知的情况下通

常将数据图 $G(V_G, E_G, l_G)$ 简写为 $G(V_G, E_G)$ 或 G，模式图的表示和含义同理可知。 □

定义 5.2 (图模拟 [13]): 回顾章节 2.1.2 可知，给定模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$，Q 与 G 通过图模拟语义匹配，记为 $Q \prec G$，当且仅当存在一个节点集合之间的二元关系 $R \subseteq V_Q \times V_G$，使得

（1）对于每对节点 $(u, v) \in R$, 节点 u 和节点 v 有相同的标签，即 $l_Q(u) = l_G(v)$；并且

（2）对于 Q 中每个节点 u，在 G 中都存在一个节点 v，使得（a）$(u, v) \in R$，并且（b）对于 Q 中每条边 $e = (u, u')$，在 G 中都存在一条边 $e' = (v, v')$，使得 $(u', v') \in R$，并且 $l_Q(e) = l_G(e')$, 即边 e 和边 e' 有相同的标签。 □

如例 5.1 所示，节点标签之间存在“是一种”关系，而这种“是一种”关系可以由知识图谱中的语义分类图详细的表示出来。

定义 5.3 (语义分类图): 语义分类图是节点带有标签信息的有根森林状数据图，记为 $T(V_T, E_T, l_T)$。其中，（1）一条由节点 u 指向节点 v 的边代表“是一种”关系；并且（2）l_T 是一个标签单射函数将 V_T 中所有节点映射至节点标签集合 Σ_V 中不同的标签。 □

直观理解，T 定义了数据图中节点标签之间存在的特化-细化层次结构。在 T 中节点 u 到节点 u' 的距离记为 $\mathsf{dist}_T(u, u')$，其计算方法如下：如果在 T 中节点 u 是节点 u' 的祖先节点，那么 $\mathsf{dist}_T(u, u')$ 为 T 中节点 u 到节点 u' 的最短路径中包含边的个数；否则，记为 $+\infty$。本文用 $\mathsf{desc}_T(u)$ 表示 T 中 u 的后代节点个数（包括 u）。在只关注 T 中节点标签的情况下，本文也将 $\mathsf{dist}_T(u, u')$ 记为 $\mathsf{dist}_T(l_T(u), l_T(u'))$，将 $\mathsf{desc}_T(u)$ 记为 $\mathsf{desc}_T(l_T(u))$。

定义 5.4 (泛化图模拟): 给定模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$，Q 与 G 通过泛化图模拟语义匹配，记为 $Q \prec_T G$，当且仅当存在一个节点集合之间的二元关系 $R^T \subseteq V_Q \times V_G$，使得

（1）对于每对节点 $(u, v) \in R^T$，$l_G(v) \in \mathsf{desc}_T(l_Q(u))$ 成立；并且

（2）对于 Q 中每个节点 u，在 G 中都存在一个节点 v，使得（a）$(u, v) \in R^T$，并且（b）对于 Q 中每条边 $e = (u, u')$，在 G 中都存在一条边 $e' = (v, v')$，使得 $(u', v') \in R^T$，并且 $l_Q(e) = l_G(e')$, 即边 e 和边 e' 有相同的标签。

其中，G 中会存在多个与 Q 匹配的二元关系，但只存在一个最大二元匹配关系 R_M^T，使得对于 G 中所有与 Q 匹配的二元关系 P，P 都是 R_M^T 的子集，即 $P \subseteq R_M^T$。最大二元匹配关系 R_M^T 中所有数据图 G 中的节点和邻接边构成的子图即为 Q 在 G 中的匹配图。 □

分析可知，泛化图模拟语义通过利用语义知识图的信息，将图模拟语义中要求节点

表 16　基于泛化图模拟的图匹配查询算法 TSim

Algorithm：基于泛化图模拟的图匹配查询算法 TSim

Input: 模式图 $Q(V_Q, E_Q, l_Q)$，数据图 $G(V_G, E_G, l_G)$，语义分类图 T。
Output: 模式图 Q 在数据图 G 中的匹配图 G_s。

1. **for each** $u \in V_Q$ **do**
2. 　　$\mathsf{sim}(u) := \{ v \in V \mid l_G(v) \in \mathsf{desc}_T(l_Q(u))\}$;
3. **while** there are changes **do**
4. 　　**for each** edge (u, u') in E_Q and **each** node $v \in \mathsf{sim}(u)$ **do**
5. 　　**if** there exists no edge (v, v') in G such that
$v' \in \mathsf{sim}(u')$ and $l_G(v, v') = l_Q(u, u')$ **then**
6. 　　$\mathsf{sim}(u) = \mathsf{sim}(u) \setminus \{v\}$;
7. 　　**if** $\mathsf{sim}(u) = \emptyset$ **then return** $\emptyset$;
8. R_M^T:=$\{(u, v) \mid u \in V_Q, v \in \mathsf{sim}(u)\}$;
9. Construct the match graph G_s *w.r.t.* R_M^T;
10. **return** G_s;

标签完全相同的匹配关系放松为“是一种”的匹配关系。

例 5.3: 给定模式图 Q_1 和数据图 G_1，如图 25 所示，给定语义分类图 T_1，如图 26 所示。可以验证 $Q_1 \not\prec G_1$，即 Q_1 与 G_1 不能基于图模拟语义匹配。然而，根据 T_1，可以验证 $Q_1 \prec_T G_1$ 成立；并且，Q_1 在 G_1 中的最大二元匹配关系将 Q_1 中节点 museum，river 和 restaurant 分别匹配至 G_1 中节点 {exhibition_hall}, {river} 和 {take_away_food，restaurant}。 □

本章为泛化图模拟语义设计图匹配查询算法并且证明以下结论成立。

定理 5.1: 给定模式图 Q、数据图 G 和语义分类图 T，判定 $Q \prec_T G$ 是否成立并且计算 Q 在 G 中基于泛化图模拟的最大二元匹配关系可以在三次方多项式时间内计算完毕。 □

本文通过设计基于泛化图模拟的图匹配查询算法 TSim 来证明定理 5.1。

基于泛化图模拟的图匹配查询算法 TSim。算法伪代码如表 16 所示，该算法输入为模式图 Q、数据图 G 和语义分类图 T，输出为数据图 G 中模式图 Q 的匹配图 G_s。算法 TSim 扩展了基于图模拟的图匹配查询 gSim 算法 [13]（章节 2.2 中表 2 所示），其时间复杂度

为 $O(|Q||G|)$。TSim 将 gSim 中判断两个节点标签是否完全相同的 $O(1)$ 时间检测步骤，转变为通过遍历 T 判断两个节点标签之间是否满足“是一种”关系的 $O(T)$ 时间检测步骤（定义 5.4 条件（1）），如表 16 中第二行下划线部分所示。所以，TSim 的时间复杂度为 $O(|V_Q||V_G||T|+|Q||G|)$。其中，$O(|V_Q||V_G||T|)$ 是为 Q 和 G 中所有节点对执行标签匹配检测步骤的总时间。

本文通过设计预处理过程使得 TSim 能够在 $O(1)$ 时间内判断节点标签是否满足“是一种”匹配关系，从而将 TSim 的时间复杂度降为 $O(|Q||G|)$。具体而言，在预处理过程中，TSim 将 T 中每个节点用一个长度为 $|V_T|$ 的比特串 ℓ^T 编码表示。比特串 ℓ^T 中每个比特位置对应于 T 中一个固定的节点，初始值全部为 0。对于 T 中任意一个节点 w，其编码比特串为 $\ell^T(w)$。TSim 通过遍历 T 将 T 中节点 w 的所有后代节点 $\mathsf{desc}_T(w)$ 找出，再将 $\mathsf{desc}_T(w)$ 中每个后代节点在比特串 $\ell^T(w)$ 中的相应比特位置置 1，其他位置为 0。预处理过程时间复杂度为 $O(|V_T||T|)$。

基于预处理过程，算法 TSim 的第二行步骤可以在 $O(1)$ 时间内计算完毕。对于 Q 中一个节点 u，TSim 在比特串集合中找出其节点标签 $l_Q(u)$ 对应的比特串 $\ell^T(l_Q(u))$。对于 G 中一个节点 v，TSim 判断其节点标签 $l_G(v)$ 在比特串 $\ell^T(l_Q(u))$ 中对应的比特位置是否为 1。如果是 1，则将节点 v 并入节点 u 的候选匹配节点集合 $\mathsf{sim}(u)$ 中; 如果是 0，则节点 v 与节点 u 不可能匹配。所以，算法 TSim 的时间复杂度为 $O(|Q||G|)$。

算法正确性和复杂性分析。算法 TSim 的正确性由以下性质保证。对于模式图中每个节点 u 和数据图中每个节点 v，如果 $l_G(v) \in \mathsf{desc}_T(l_Q(u))$，则标签 $l_G(v)$ 在标签 $l_Q(u)$ 的比特串中的对应比特位置必然为 1。如上所述，预处理过程时间复杂度为 $O(|V_T||T|)$。算法 TSim 的时间复杂度为 $O(|Q||G|)$，与算法 gSim [13] 相同。

5.2.2 图匹配查询松弛重构计算框架

本章首先提出模式图的基于语义分类信息的查询松弛，而后提出通用的图匹配查询松弛重构计算框架。

定义 5.5 (标签松弛): 基于语义分类图 T 的一个标签松弛 δ 为 $l \to l'$。其中，标签 l' 为标签 l 在 T 中的祖先标签。 □

定义 5.6 (查询松弛): 给定模式图 Q，语义分类图 T 和正整数 μ。模式图 Q 的一个基于 T 的 μ-有界查询松弛 Δ 是由一组标签松弛构成的集合，使得（1）对于 Δ 中任意标签松弛 $l \to l'$，l 为 Q 中的节点标签并且 $\mathsf{dist}_T(l', l) \le \mu$；并且（2）对于 Δ 中任意两个标签松弛

$l_1 \to l_1'$ 和 $l_2 \to l_2'$，$l_1 \neq l_2$ 成立。在上下文已知情况下，本文称 Δ 为 Q 的一个查询松弛。□

对于查询松弛 Δ 中的每个标签松弛 $l \to l'$，将 Q 中标签为 l 的所有节点的标签由 l 替换为 l'。本文用 $Q \oplus \Delta$ 表示根据 Δ 中所有标签松弛将 Q 中相应标签替换后的模式图，称为松弛模式图。

直观理解，μ 是用于限制 Δ 中标签松弛的松弛距离的参数，使得松弛模式图 $Q \oplus \Delta$ 表达的查询意图与原始模式图 Q 表达的查询意图的差异在可接受范围内。因此，可以通过调整参数 μ 控制 Q 松弛变化的程度。

例 5.4: 给定模式图 Q_2 和数据图 G_1，如图 25 所示，给定语义分类图 T_1，如图 26 所示。当 $\mu = 2$，$\Delta = \{\delta = \text{newspaper} \to \text{media}\}$ 为 Q_2 的一个基于 T_1 的 2-有界查询松弛。$Q_2 \oplus \Delta$ 是将 Q_2 中节点标签 newspaper 替换为标签 media 的松弛模式图。 □

图匹配查询松弛重构计算框架。本文提出通用的图匹配查询松弛重构计算框架来松弛模式图从而查询得到更多有意义的图匹配结果，使得图匹配查询引擎能够根据用户的查询意图高效的返回用户期望的图匹配查询结果，并为用户提供结果如何产生的溯源解释。给定模式图 Q，数据图 G，语义分类图 T，正整数 μ 和 k, 计算框架包括以下四个主要步骤。

（1）首先将语义分类信息引入图匹配查询，构建基于 T 的泛化图匹配查询语义。

（2）而后，根据图匹配查询松弛排序函数，生成模式图的 top-k 松弛模式图。

（3）高效计算得出模式图和 top-k 松弛模式图在数据图中的图匹配查询结果。

（4）为用户提供为何新的图匹配查询结果能够由松弛模式图查询得出的溯源解释。

本文在论述该计算框架工作原理时以图模拟为图匹配查询语义。然而该计算框架是通用方法，适用于任意一种图匹配查询语义。本文在章节 5.2.1 中已经介绍完毕步骤（1）相关内容。本文将在章节 5.3 和章节 5.4 介绍步骤（2）相关内容；在章节 5.5 介绍步骤（3）相关内容；在章节 5.6 介绍步骤（4）相关内容。

5.3　图匹配查询松弛排序函数

对于模式图 Q，数据图 G，语义分类图 T 和正整数 μ，存在多达指数个 Q 的基于 T 的 μ-有界查询松弛。本文通过定义图匹配查询松弛排序函数生成 top-k 最优查询松弛。

本章将设计两个查询松弛排序函数：拓补排序函数利用松弛模式图和原始模式图 Q 之间的松弛距离对查询松弛排序，并综合考虑 G 和 T 的相关特征信息（章节 5.3.1）；多元化拓补排序函数将拓补排序函数与多元化函数结合对查询松弛综合排序（章节 5.3.2）。

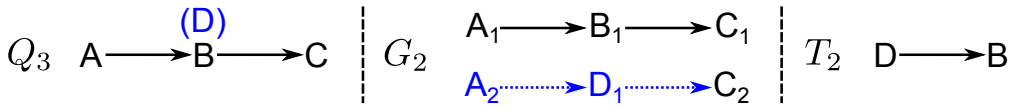


图 27　松弛率示例

基于两个查询松弛排序函数，本文分别提出两个 top-k 查询松弛问题。

5.3.1　拓补排序函数

拓补排序函数是双目标优化函数，由两部分组成：（a）松弛率，用于衡量松弛模式图与原始模式图之间的松弛距离，从而评估松弛模式图与原始模式图的差异，进而评估松弛模式图相对原始模式图的查询准确度；（b）信息率，用于估算松弛模式图在数据图中图匹配查询结果的数目，从而评估松弛模式图能够改善原始模式图查询结果的程度。本节将详细介绍拓补排序函数相关定义。

定义 5.7 (松弛率): 模式图 $Q(V_Q, E_Q, l_Q)$ 的基于 T 的一个查询松弛 $\delta = l \rightarrow l'$ 的松弛率 $\gamma_Q(\delta)$ 定义为：

$$\sum_{u \in V_Q,\, l_Q(u)=l} \mathsf{rank}_Q(u) \cdot \rho(\mathsf{dist}_T(l', l)) \tag{5.1}$$

其中，$\mathsf{rank}_Q(u)$ 表示在 Q 中能够通过一条有向路径到达节点 u 的所有节点 u' 的个数。$\rho(x)$ 是用于正规化 $\mathsf{dist}_T(l', l)$ 权重的单调递增函数。$\rho(x)$ 的常见选择是 $\rho(x) = x$ 和 $\rho(x) = e^x$，经常被应用于衡量社交距离等应用 [111]。本文默认使用 $\rho(x) = e^x$。 □

直观理解，越大的 $\mathsf{dist}_T(l', l)$ 值表示节点 u 更容易在数据图中查询得到基于泛化图模拟语义的匹配节点。这个效果被 $\mathsf{rank}_Q(u)$ 进一步增强，这是由泛化图模拟语义的迭代特征决定的。根据泛化图模拟语义，松弛 Q 中节点 u 的标签将使 Q 中 u 的祖先节点匹配更多数据图中节点，然而不会改变 Q 中 u 的后代节点在数据图中的匹配状态。

例 5.5: 给定模式图 Q_3，数据图 G_2，和语义模式图 T_2，如图 27 所示。Q_3 在 G_2 中基于泛化图模拟语义的图匹配查询结果包括节点 A_1、B_1、C_1 和 C_2。一个查询松弛 $\Delta = \{\mathsf{B} \rightarrow \mathsf{D}\}$ 将 Q_3 中的节点 B 的标签由 B 松弛为 D，从而使得 G_2 中节点 D_1 与松弛模式图 $Q_3 \oplus \Delta$ 中节点 D 匹配，进而使得 G_2 中节点 A_2 与松弛模式图 $Q_3 \oplus \Delta$ 中节点 A 匹配。然而，松弛 Q_3 中节点 B 的标签并不会改变节点 C 在 G_2 中的匹配状态。观察可知，$\mathsf{rank}_{Q_3}(\mathsf{C}) = 3 > \mathsf{rank}_{Q_3}(\mathsf{B}) = 2 > \mathsf{rank}_{Q_3}(\mathsf{A}) = 1$ 成立。 □

观察可知，越小的 $\gamma_Q(\delta)$ 值表示 Q 根据 δ 松弛后的模式图与原始模式图 Q 差异越小。

定义 5.8 (信息率): 给定模式图 $Q(V_Q, E_Q, l_Q)$ 和数据图 $G(V_G, E_G, l_G)$。Q 的标签松弛 $\delta = l \to l'$ 在 G 中的信息率 $\mathcal{I}_{(Q,G)}(\delta)$ 定义为：

$$\frac{|\mathsf{cand}_{(G,T)}(l)|}{|\mathsf{cand}_{(G,T)}(l')|} \tag{5.2}$$

其中，$\mathsf{cand}_{(G,T)}(l)$ 是 G 中满足 $l_G(v) \in \mathsf{desc}_T(l)$ 的节点 v 的集合。 □

直观理解，$\mathcal{I}_{(Q,G)}(\delta)$ 衡量了 Q 根据 δ 松弛后的模式图在 G 中存在候选匹配节点的数量。越大的 $\mathcal{I}_{(Q,G)}(\delta)$ 值表示松弛模式图在数据图中有查询出更多匹配节点的能力。

定义 5.9 (拓补排序函数): 给定模式图 Q，数据图 G，语义分类图 T，和一个 Q 的基于 T 的查询松弛 Δ。Δ 的拓补排序函数 $\Gamma(Q, \Delta)$ 定义为：

$$\sum_{\delta \in \Delta} \gamma_Q(\delta) \cdot \mathcal{I}_{(Q,G)}(\delta) \tag{5.3}$$

其中，查询松弛 Δ 是由标签松弛 δ 构成的集合。 □

直观理解，$\Gamma(Q, \Delta)$ 是一个双目标优化函数，通过减少松弛模式图 $Q \oplus \Delta$ 与原始模式图 Q 的差异并且提高松弛模式图在数据图中匹配节点的数量来优化松弛模式图的质量。通过最小化 $\Gamma(Q, \Delta)$ 值，可以一方面通过利用松弛率来提高 $Q \oplus \Delta$ 在 G 中匹配结果的质量；另一方面可以通过信息率来最大化 $Q \oplus \Delta$ 在 G 中匹配结果的数量

这促使本文研究 top-k 查询松弛问题，问题定义如下所示。

基于拓补排序函数的 top-k 查询松弛问题（kPR）。本文用 $\mathcal{U}_\mu(Q, T)$ 表示由 Q 的基于 T 的所有 μ-有界查询松弛构成的集合。给定 Q、G、T，正整数 μ 和 k，kPR 即找到一个 k-集合 $S \subseteq \mathcal{U}_\mu(Q, T)$，使得以下等式成立。

$$S = \underset{S' \subseteq \mathcal{U}_\mu(Q,T), |S'|=k}{\arg\min} \sum_{\Delta \in S'} \Gamma(Q, \Delta) \tag{5.4}$$

也就是说，kPR 问题是找到一个由 k 个 μ-有界查询松弛构成的集合，并且这个集合中所有查询松弛对应的拓补排序函数值之和最小。

例 5.6: 给定模式图 Q_2 和数据图 G_1，如图 25 所示，给定语义分类图 T_1，如图 26 所示。假设 G_1 中还存在 7 个孤立节点，其标签分别为 magazine、radio_show, venue、theater、valley、mountain 和 airport（未在图 25 中显示）。假设 $k = 2$，$\mu = 2$。模式图 Q_2 的所有基于 T 的标签松弛和部分查询松弛如表 17 所示。

表 17　模式图标签松弛和查询松弛示例

模式图查询松弛	$\Gamma(Q_2, \Delta_i)$
Δ_1 = {δ_1 = newspaper → media }	1.359
Δ_2 = {δ_2 = museum → cultural_center }	2.175
Δ_3 = {δ_3 = river → natural_place }	3.695
Δ_4 = {δ_4 = river → body_of_water }	4.077
Δ_5 = {δ_5 = restaurant → leisure_center }	7.249
Δ_6 = {δ_6 = restaurant → architecture.. }	14.778
...	...

观察可知以下结论成立。(a)S={Δ_1, Δ_2} 为 kPR 问题的 top-2 查询松弛。(b)$\Gamma(Q_2, \Delta_3)$ 值小于 $\Gamma(Q_2, \Delta_4)$ 值，尽管 $\text{dist}_T(\text{natural\_place}, \text{river}) = 2$ 大于 $\text{dist}_T(\text{body\_of\_water}, \text{river}) = 1$。这是因为尽管松弛率将松弛距离小的查询松弛排序更高，然而信息率会将能够在 G_1 中查询得到更多图匹配结果的查询松弛排序更高，拓补排序函数是综合考虑松弛率和信息率排序的函数。 □

5.3.2　多元化拓补排序函数

理想的 top-k 松弛模式图集合不仅应该在提供更多图匹配查询结果的基础上保持与原始模式图较小的差异，还应该具有多元化性质使得 k 个松弛模式图能够在数据图中查询得到更多不同的结果从而获取更多信息。因此，本章将设计多元化拓补排序函数将拓补排序函数与查询松弛集合的多元化性质相结合。

定义 5.10 (相似距离): 为了衡量模式图的一个查询松弛集合的多元化程度，需要首先衡量每对查询松弛之间的差异性。对于模式图 Q 的任意两个查询松弛 Δ_1 和 Δ_2，Δ_1 和 Δ_2 间的相似距离 $\theta_Q(\Delta_1, \Delta_2)$ 定义为：

$$\frac{|L(Q \oplus \Delta_1) \cap L(Q \oplus \Delta_2)|}{|L(Q \oplus \Delta_1) \cup L(Q \oplus \Delta_2)|} \tag{5.5}$$

其中，$L(Q)$ 是由模式图 Q 中出现的所有节点标签构成的集合。也就是说，相似距离衡量松弛模式图之间节点标签的重叠程度。 □

例 5.7: 回顾例 5.6 中内容。可以验证以下结论成立：（a）$\theta_{Q_2}(\Delta_1, \Delta_2) = \frac{2}{6}$；并且（b）$\theta_{Q_2}(\Delta_7, \Delta_8) = 0$，其中，$\Delta_7$={$\delta_1, \delta_2$}，$\Delta_8$={$\delta_3, \delta_5$}；也就是说，$Q_2 \oplus \Delta_7$ 和 $Q_2 \oplus \Delta_8$ 不存在任

何相同的节点标签。所以，Δ_7 和 Δ_8 是模式图 Q_2 的所有查询松弛中差异最大的一对查询松弛。可以验证松弛模式图 $Q_2 \oplus \Delta_8$ 在 G_1 中的图匹配查询结果包括 G_1 底部的整个连通分量。然而，该查询结果偏离了原始模式图 Q_2 的查询目标，查询松弛 Δ_8 将 Q_2 的观光行程规划需求变成了 $Q_2 \oplus \Delta_8$ 的娱乐行程规划需求。这是因为此时的排序只考虑了查询松弛间的差异程度这一个因素，而未考虑拓补排序函数中涉及的因素。 □

本文将拓补排序函数与查询松弛集合的多元化性质相结合提出多元化拓补排序函数，问题定义如下所示。

定义 5.11 (多元化拓补排序函数): 给定 Q 的基于 T 的一个由 k 个查询松弛 $\Delta_1, \ldots \Delta_k$ 构成的集合 S。查询松弛集合 S 的多元化拓补排序函数 $F(Q,S)$ 定义为：

$$\lambda \cdot (k-1) \sum_{\Delta_i \in S} \widehat{\Gamma}(Q, \Delta_i) + 2 \cdot (1-\lambda) \sum_{\Delta_i \in S, \Delta_j \in S, i<j} \theta_Q(\Delta_i, \Delta_j) \tag{5.6}$$

其中，$\lambda \in [0,1]$ 是一个由用户指定的参数；$\widehat{\Gamma}(Q,\Delta_i) = \frac{\Gamma(Q,\Delta_i)}{|V_Q| \cdot |L(Q)| \cdot e^{\mu}}$ 是用于正规化拓补排序函数的函数；$L(Q)$ 是由模式图 Q 中出现的所有节点标签构成的集合。因为多元化拓补排序函数的拓补排序函数部分有 k 个数字，多元化排序函数部分有 $\frac{k \cdot (k-1)}{2}$ 个数字，所以将拓补排序函数部分乘以因子 $(k-1)$ 从而保持两部分一致。 □

基于多元化拓补排序函数的 top-k 查询松弛问题（$\mathsf{kPR}_{\mathsf{DF}}$）。同 kPR 问题类似，本文用 $\mathcal{U}_{\mu}(Q,T)$ 表示由 Q 的基于 T 的所有 μ-有界查询松弛构成的集合。给定 Q、G、T，正整数 μ 和 k，kPR 即找到一个 k-集合 $S \subseteq \mathcal{U}_{\mu}(Q,T)$，使得以下等式成立。

$$S = \underset{S' \subseteq \mathcal{U}_{\mu}(Q,T), |S'|=k}{\arg\min} F(Q,S') \tag{5.7}$$

也就是说，$\mathsf{kPR}_{\mathsf{DF}}$ 问题是找到一个由 k 个 μ-有界查询松弛构成的集合，并且这个集合对应的多元化拓补排序函数值最小。

例 5.8: 回顾例 5.6 中内容。可以验证以下结论成立：（a）当 $\lambda = 1$ 时，即只使用拓补排序函数时，top-2 查询松弛为 $\{\Delta_1, \Delta_2\}$；（b）当 $\lambda = 0$ 时，即只使用多元化排序函数时，top-2 查询松弛为 $\{\Delta_7=\{\delta_1, \delta_2\}, \Delta_8=\{\delta_3, \delta_5\}\}$。另外，（c）当 $0.823 < \lambda < 0.924$ 时，top-2 查询松弛为 $\{\Delta_3, \Delta_7\}$；当 $\lambda \leq 0.823$ 时，top-2 查询松弛为 $\{\Delta_7, \Delta_8\}$；当 $\lambda \geq 0.924$ 时，top-2 查询松弛为 $\{\Delta_1, \Delta_2\}$。 □

5.4 图匹配查询松弛排序算法

本章首先对基于拓补排序函数和多元化拓补排序函数的 top-k 查询松弛问题分别进行理论分析。而后，基于理论分析结果，本章将分别为两个问题设计算法。章节 5.3 和本章内容共同构成图匹配查询松弛重构计算框架中步骤（2）的基础。本章首先在章节 5.3.1 对 kPR 问题开展研究，而后在章节 5.3.2 对 kPR_{DF} 问题开展研究，

5.4.1 拓补排序算法

本节通过理论分析和算法设计两个方面对 kPR 问题开展研究。本文假设已执行预处理过程将 G 中每个节点标签 l 的 $|\text{cand}_{(G,T)}(l)|$ 值计算得出。该预处理过程可以在 $O(|V_T||V|)$ 时间内计算完毕。

定理 5.2: kPR 问题存在时间复杂度为 $O(\mu k|V_Q||Q|)$ 的算法可以根据 T 和 G 的信息计算得出 Q 的 top-k 个 μ-有界查询松弛。算法执行时间只取决于 $|Q|$，与 $|G|$ 和 $|T|$ 均无关。□

本文通过设计具体算法证明定理 5.2。本文通过使用 Lawler 程序 [102] 为 kPR 问题设计了 PTIME 精确算法。Lawler 程序具有如下性质：对于可以用 n 个 0-1 变量表示成整数规划的优化问题，如果该问题的最优（top-1）结果可以在 $c(n)$ 时间内计算得出，那么该问题的 top-k 结果可以在 $O(k \cdot c(n)) + B$ 时间内计算得出。其中，B 是划分出可行解子空间的总时间。

基于拓补排序函数的查询松弛排序算法 relTF。算法 relTF 伪代码如表 18 和表 19 所示。其中，算法 relTF 包括两个计算过程 topRel 和 LawlerBranch，如表 19 所示。relTF 使用的数据结构包括：（a）列表 L_{TR} 用于存储当前已经得出的 top-k 查询松弛；（b）优先队列 Q 用于缓存生成 top-k 查询松弛的候选查询松弛，同时这些候选查询松弛也被用于划分搜索空间。在初始化步骤后（第 1 行），relTF 为 Q 中出现的每个节点标签生成候选标签松弛列表，所有列表构成候选标签松弛组合 $\mathcal{L}_1$，并且这些标签松弛的松弛距离不大于 μ（第 2-3 行）。具体而言，对于 Q 中出现的每个节点标签 ℓ_i，relTF 为其生成候选标签松弛列表 $\ell_i \rightarrow \ell'$ $(0 \leq \text{dist}_T(\ell', \ell_i) \leq \mu)$，$\mathcal{L}_1$ 是由 Q 中所有节点标签的候选标签松弛列表构成的组合。而后，relTF 调用计算过程 topRel 从 $\mathcal{L}_1$ 中找出最优（top-1）查询松弛 Δ_1，并将 Δ_1 与产生 Δ_1 的搜索空间 $\mathcal{L}_1$ 一同压入优先队列 Q 中（第 4 行）。

之后，算法 relTF 将搜索剩余的最优 $k-1$ 个查询松弛问题转换为迭代搜索 $k-1$ 个 top-1 查询松弛问题。relTF 将 $\mathcal{L}_1$ 迭代划分为多个子搜索空间，并在子搜索空间内查找最优（top-1）查询松弛，直至 Q 为空或者 top-k 查询松弛都已经被找到。relTF 每次从 Q

表 18　基于拓补排序函数的查询松弛排序算法 relTF (I)

Algorithm：基于拓补排序函数的查询松弛排序算法 relTF

Input: 模式图 Q，数据图 G，语义分类图 T，正整数 μ 和 k。
Output: Top-k 查询松弛。

1. $L_{\mathrm{TR}} := [\,]$; $\mathcal{Q} := nil$; $K := 1$;
2. **for each** label ℓ_i in Q **do**　　/* 假设 Q 包括标签 $\ell_1, \ldots, \ell_m$ */
3. 　　generate a list L_i of label relaxations for ℓ_i bounded by μ in T;
4. $\mathcal{L}_1 := (L_1, \ldots, L_m)$; $\Delta_1 := \mathrm{topRel}(Q, G, \mathcal{L}_1)$; $\mathcal{Q}.\mathrm{push}(\langle \Delta_1, \mathcal{L}_1 \rangle)$;
5. **while** $\mathcal{Q} \neq \emptyset$ **do**
6. 　　**if** $K = k + 1$ **then break**;
7. 　　$\langle \Delta_K, \mathcal{L}_K \rangle := \mathcal{Q}.\mathrm{pop}()$;　　/* 在 $\mathcal{Q}$ 中 Δ_K 的 $\Gamma(Q, \Delta_K)$ 值最小 */
8. 　　append Δ_K to L_{TR};
9. 　　sub-collections $\mathcal{L}^{s_1}, \ldots, \mathcal{L}^{s_m} := \mathrm{LawlerBranch}(\Delta_K, \mathcal{L}_K)$;
10. 　　**for** i in $[1, m]$ **do**
11. 　　　　$\Delta_i := \mathrm{topRel}(Q, G, \mathcal{L}^{s_i})$; $\mathcal{Q}.\mathrm{push}(\langle \Delta_i, \mathcal{L}^{s_i} \rangle)$;
12. 　　$K := K + 1$;
13. **return** L_{TR};

中弹出拓补排序函数值最小的查询松弛 Δ_K 以及产生 Δ_K 的搜索空间，即候选标签松弛组合 $\mathcal{L}_K$（第 7 行）。relTF 将 Δ_K 放入结果列表 L_{TR} 中作为当前第 K 优的查询松弛（第 8 行）。而后，算法 relTF 调用 Lawler 计算过程 LawlerBranch（参见 [102]），基于 $\mathcal{L}_K$ 和 Δ_K 生成 $\mathcal{L}_K$ 的子搜索空间 $\mathcal{L}^{s_1}, \ldots, \mathcal{L}^{s_m}$。其中，$\mathcal{L}^{s_i}$ $(1 \leq i \leq m)$ 是 $\mathcal{L}_K$ 的子候选标签松弛组合（第 9 行）。算法 relTF 调用计算过程 topRel 在每个子搜索空间 $\mathcal{L}^{s_i}$ 中找出最优查询松弛 Δ_i（第 10-11 行）。当 top-k 查询松弛都已经被找到时，relTF 返回最终 top-k 查询松弛结果列表 L_{TR}（第 13 行）。

最优查询松弛计算过程 topRel。计算过程 topRel 伪代码如表 19 所示。给定模式图 Q，数据图 G，和候选标签松弛组合 $\mathcal{L}$，对于 Q 中每个出现的节点标签，topRel 通过在其候选标签松弛列表 L_i 中选择 $\gamma_Q(\delta) \cdot \mathcal{I}_{(Q,G)}(\delta)$ 值最小的标签松弛 δ，再将所有选出的标签松弛 δ^i_{min} 合并即得到 $\mathcal{L}$ 中最优（top-1）查询松弛。

表 19　基于拓补排序函数的查询松弛排序算法 relTF (II)

Procedure：**最优查询松弛计算过程** topRel($Q, G, \mathcal{L}$)

Input: 模式图 Q，数据图 G，候选标签松弛组合 $\mathcal{L}$。
Output: 基于拓补排序函数 $\Gamma(Q, \Delta)$，$\mathcal{L}$ 中最优的查询松弛 Δ。

1. **for each** i in $[1, m]$ **do**
2. $\quad \delta^i_{\min} := \arg\min_{\delta \in L_i} \gamma_Q(\delta) \cdot \mathcal{I}_{(Q,G)}(\delta)$;
3. $\Delta := \{\delta^1_{min}, \delta^2_{min}, \ldots, \delta^m_{min}\}$;
4. **return** Δ;

Procedure：**Lawler 过程** LawlerBranch($\Delta, \mathcal{L}$)

Input: 查询松弛 $\Delta = \{\delta^1, \delta^2, \ldots, \delta^m\}$，候选标签松弛组合 $\mathcal{L} = \{\mathcal{L}'_1, \ldots, \mathcal{L}'_m\}$。
Output: m 个子候选标签松弛组合。

1. **for** i **from** 1 **to** m **do**
2. $\quad$ **for** j **from** 1 **to** $i-1$ **do**
3. $\quad\quad \mathcal{L}^i_j := \{\delta^j\}$;
4. $\quad \mathcal{L}^i_i := \mathcal{L}'_i \setminus \{\delta^i\}$;
5. $\quad$ **for** j **from** $i+1$ **to** m **do**
6. $\quad\quad \mathcal{L}^i_j := \mathcal{L}'_j$;
7. $\mathcal{L}^i := (\mathcal{L}^i_1, \ldots, \mathcal{L}^i_m)$;
8. **return** $\langle \mathcal{L}^1, \ldots, \mathcal{L}^m \rangle$;

Lawler 计算过程 LawlerBranch [102]。计算过程 LawlerBranch 伪代码如表 19 所示。给定查询松弛 Δ，候选标签松弛组合 $\mathcal{L}$。其中，Δ 是上个迭代轮次找到的全局最优查询松弛，$\mathcal{L}$ 是产生 Δ 的搜索空间。LawlerBranch 根据 Δ 将 $\mathcal{L}$ 划分为 m 个新的子候选标签松弛组合 $\mathcal{L}^i$ $(1 \le i \le m)$。每个子候选标签松弛组合中产生的最优查询松弛中的最优查询松弛即为当前轮次的全局最优查询松弛。

算法正确性和复杂性分析。算法 relTF 的正确性由 Lawler 过程的正确性保证（参见 [102]）。算法 relTF 的时间复杂度为 $O(\mu k|V_Q||E_Q|)$。其中，计算过程 topRel 的时间复杂度为 $O(\mu|V_Q||E_Q|)$，计算过程 LawlerBranch 迭代划分子搜索空间的总时间花

费为 $O(k|V_Q|^2)$。根据前文中提到的 Lawler 过程的性质，算法 relTF 的时间复杂度为 $O(k\cdot\mu|V_Q||E_Q|)+O(k|V_Q|^2)=O(\mu k|V_Q||Q|)$。

5.4.2　多元化拓补排序算法

本节通过理论分析和算法设计两个方面对 kPR_{DF} 问题开展研究。通过上节分析可知 kPR 问题是 PTIME 可解问题，然而 kPR_{DF} 问题的固有难度较高，如下所示。

定理 5.3: （1）kPR_{DF} 的判定问题是 NP-完全问题。
（2）kPR_{DF} 的优化问题不属于 APX 类可近似问题（APX 类包含具有常数近似度保证的多项式时间近似算法的问题 [73]）　□

尽管 kPR_{DF} 问题的计算复杂度和近似复杂度都很高，本文在仔细分析 kPR_{DF} 问题特征的基础上设计了算法 relDF。算法 relDF 通过将 kPR_{DF} 问题归约至已经被深入研究的 *maximum dispersion*（maxDP）问题 [112]，从而可以使用 maxDP 问题的高效精确，近似或启发算法解决 kPR_{DF} 问题。maxDP 问题即从一个边上带（正）权重的完全图 G_c 中找到一个由 k-节点集合 V_k 生成的诱导子图 G_c'，使得 G_c' 中所有边上权重之和最大。maxDP 问题是一个经典的最大化优化问题，存在很多高效精确，近似和启发算法 [112]。

基于多元化拓补排序函数的查询松弛排序算法 relDF。算法 relDF 的关键是将 kPR_{DF} 问题归约至 maxDP 问题，其归约过程如下。需要注意的是尽管 kPR_{DF} 问题是最小化优化问题，而 maxDP 问题是最大化优化问题，算法 relDF 的归约过程仍然保证 maxDP 问题的最优解即为 kPR_{DF} 问题的最优解。

给定模式图 Q，数据图 G，语义分类图 T，正整数 μ 和 k，算法 relDF 按如下方法构建 maxDP 问题中的带（正）权重完全图 G_c。（1）relDF 将 Q 的每个 μ-有界查询松弛 Δ 用一个 G_c 中节点 u_Δ 表示。（2）对于任意两个 G_c 中节点 u_{Δ_1} 和 u_{Δ_2}，边 $e=(u_{\Delta_1},u_{\Delta_2})$ 上的权重 $w(e)$ 定义为：

$$M-\lambda\sum_{i\in\{1,2\}}\widehat{\Gamma}(Q,\Delta_i)-2(1-\lambda)\theta_Q(\Delta_1,\Delta_2) \tag{5.8}$$

其中，$M=2\lambda\cdot\max_{\Delta\in U}\widehat{\Gamma}(Q,\Delta)+2(1-\lambda)$，$U$ 是由 Q 的基于 T 的所有 μ-有界查询松弛构成的集合。验证可知因为 $w(e)>0$，所以 G_c 是 maxDP 问题的一个合法实例。

显然，G_c 中一个 k-节点集合 V_k 编码了一个由 Q 的 k 个查询松弛构成的集合。所以，算法 relDF 通过调用 maxDP 问题的算法为 maxDP 问题返回一个 k-节点集合 V_k，则对应于为 kPR_{DF} 问题返回一个由 k 个查询松弛构成的集合。

定理 5.4: 如果 G_c 中一个 k-节点集合 V_k 是 maxDP 问题的最优解，那么由 V_k 编码对应的一个由 k 个查询松弛构成的集合则是 kPR_{DF} 问题的最优解。 □

定理 5.4 保证了以下性质：只要归约问题 maxDP 能够返回准确结果，那么被归约问题 kPR_{DF} 也能返回 top-k 查询松弛。下面证明该定理成立。用 S_k 表示由 V_k 编码对应的一个由 k 个查询松弛构成的集合。G_c 中由 V_k 生成的诱导子图中所有边权重之和 W_k 为：

$$\sum_{u,v\in S_k, u\neq v} w(u,v) = \frac{k(k-1)}{2}\cdot M - \lambda(k-1)\sum_{\Delta\in S_k}\widehat{\Gamma}(Q,\Delta) - 2(1-\lambda)\sum_{\Delta_i,\Delta_j\in S_k, i<j}\theta_Q(\Delta_i,\Delta_j) \tag{5.9}$$

根据上式，可知以下等式成立：

$$W_k = \frac{k(k-1)}{2}\cdot M - F(Q,S_k) \tag{5.10}$$

因为 V_k 是 maxDP 问题的最优解，可知 W_k 是 maxDP 问题所有 k-节点集合可行解中的最大值。根据公式 5.10 可知，$F(Q,S_k)$ 是 kPR_{DF} 问题的所有 k-查询松弛集合可行解中的最小值。

备注。本文可以通过上述归约方法设计高效算法的原因是 μ 和 $L(Q)$ 的值通常很小，从而能够以较低的计算开销事先计算得出 Q 的所有 μ-有界查询松弛。事实的确如此，知识图谱 DBpedia 的语义分类图中，树状分类图的最大高度仅为 6，也就是 μ 在 DBpedia 中的最大取值仅为 6；树状分类图的平均高度仅为 2.29，也就是 μ 在 DBpedia 中的平均取值仅为 2.29（详见章节 5.7）。

5.5 松弛图匹配查询多查询处理优化

章节 5.4 通过设计查询松弛排序算法生成了 top-k 松弛模式图。本章提出松弛图匹配查询多查询处理优化方法计算 top-k 松弛模式图在数据图中的图匹配查询结果。本章内容构成图匹配查询松弛重构计算框架（章节 5.2.2）中步骤（3）的基础。

给定模式图 Q，数据图 G，语义分类图 T，和 k 个查询松弛 $\Delta_1, \ldots, \Delta_k$，下一步的目标是计算松弛模式图 $Q\oplus\Delta_1, \ldots, Q\oplus\Delta_k$ 在数据图 G 中的图匹配查询结果。最直接的解决方案是逐个依次计算松弛模式图在数据图中的图匹配查询结果。然而，分析可知所有松弛模式图具有完全相同的拓补结构，并且具有相互关联的节点标签。受此启发，本文提出松弛图匹配查询多查询处理优化方法通过最大化共享计算量，从而同时、快速的计算得出 top-k 松弛模式图在数据图中的图匹配查询结果。

表 20　松弛图匹配查询多查询处理优化算法 evalPR

Algorithm：松弛图匹配查询多查询处理优化算法 evalPR

Input: 模式图 Q，数据图 G，语义分类图 T，k 个查询松弛 $\Delta_1, \ldots, \Delta_k$。
Output: 松弛模式图 $Q \oplus \Delta_1, \ldots, Q \oplus \Delta_k$ 分别在数据图 G 中的图匹配查询结果。

```
1. compute the minimum pairing tree 𝒯 of Δ1, …, Δk for Q;
2. set u to the root of 𝒯;
3. Q:=∅; Q.push(u);
4. while Q is not empty do
5.     u := Q.pop();
6.     u.ans := BDeval(pre(u), u);   /* 有界增量计算 */
7.     Q.push(post(u));
8. return {u.ans | u is a leaf of 𝒯};
```

松弛图匹配查询多查询处理优化算法 evalPR。算法 evalPR 伪代码如表 20 所示。算法 evalPR 的工作流程分为两步：（1）构建最小匹配树 $\mathcal{T}$，一个组织所有松弛模式图的查询计算顺序从而最大化共享计算量的层次结构；（2）根据 $\mathcal{T}$ 中顺序为所有松弛模式图进行基于泛化图模拟的有界增量计算。

（1）构建最小匹配树。模式图 Q 的 k 个查询松弛 $\Delta_1, \ldots, \Delta_k$ 的最小匹配树 $\mathcal{T}$ 是一个按如下方法构建的查询松弛的层次结构。（a）$\mathcal{T}$ 中的每个节点表示一个查询松弛。（b）$\mathcal{T}$ 有 k 个叶子节点（第 0 层），对应于 Q 的 k 个查询松弛 $\Delta_1, \ldots, \Delta_k$。（c）第 $i+1$ 层节点表示的查询松弛为第 i ($i \in [0, \lceil \log k \rceil - 1]$) 层节点表示的查询松弛的最小匹配。其中，查询松弛 $\Delta_1, \ldots, \Delta_n$ 的一个匹配是一个由 $\lceil \frac{n}{2} \rceil$ 个查询松弛 $\Delta'_1, \ldots, \Delta'_{\lceil \frac{n}{2} \rceil}$ 构成的集合 P，使得（i）第 $i+1$ 层的每个查询松弛 Δ'_j ($j \in [1, \lceil \frac{n}{2} \rceil]$) 是第 i 层的两个查询松弛 Δ_p 和 Δ_q ($p, q \in [1, n]$) 的合并。Δ'_j 将 Δ_p 和 Δ_q 中的所有标签松弛合并。其中，如果 Δ_p 和 Δ_q 中存在两个标签松弛 $l \to l'$ 和 $l \to l''$ 同时作用于同一个节点标签 l，则将 l 松弛至松弛距离更远的一个标签 l''。也就是说，Δ'_j 只包含松弛距离较大的一个标签松弛 $l \to l''$（如果 $\mathsf{dist}_T(l, l') \le \mathsf{dist}_T(l, l'')$）；（ii）如果 $j \neq j'$，则 Δ'_j 和 $\Delta'_{j'}$ 对应于不同的查询松弛对。也就是说，Δ'_j 和 $\Delta'_{j'}$ 对应的第 i 层中的查询松弛是互不相交的。

如果一个 n-查询松弛集合 S 的匹配集合 P 是最小匹配当且仅当在 S 的所有匹配集

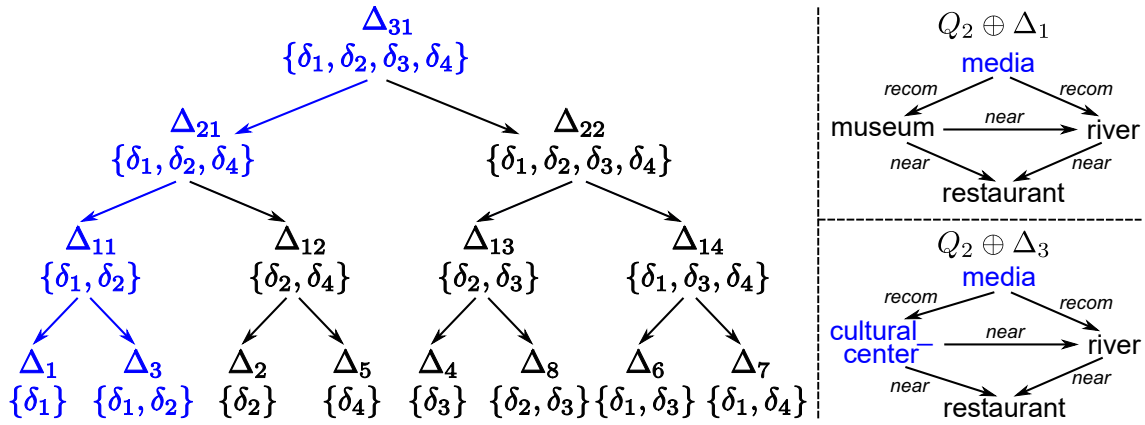


图 28　最小匹配树示例

合中，P 使得以下公式取值最小。

$$\sum_{\Delta' \in P} \sum_{\Delta_i \in \Delta', i=1,2} \sum_{u \in V_Q} |\mathsf{cand}_{(G,T)}(f_{Q \oplus \Delta'}(u)) \setminus \mathsf{cand}_{(G,T)}(f_{Q \oplus \Delta_i}(u)| \tag{5.11}$$

其中，$\Delta_i \in \Delta'$ $(i = 1, 2)$ 表示查询松弛 Δ' 合并查询松弛 Δ_1 和 Δ_2；$\mathsf{cand}_{(G,T)}(l)$ 表示 G 中满足 $l_G(v) \in \mathsf{desc}_T(l)$ 的节点 v 的集合。

直观理解，S 的最小匹配 P 将 S 中的查询松弛两两配对，使得在图匹配查询处理过程中，可以先计算 P 中松弛模式图在数据图中的查询结果，作为共享计算结果。再从 P 中松弛模式图的查询结果中“恢复”计算出 S 中配对的两个松弛模式图的查询结果。最小匹配方法使得松弛模式图两两之间共享计算量。最小匹配树递归的组织这种共享计算量，从而最小化全局计算量。

本文提出通过调用 $\lceil \log k \rceil$ 次已经被深入研究的 *maximum weighted matching* 问题的算法 [113] 来构建最小匹配树 $\mathcal{T}$。相关研究针对该问题提出了时间复杂度为 $O(k^{2.5})$ 的精确算法 [113]。所以，构建最小匹配树 $\mathcal{T}$ 的时间复杂度为 $O(k^{2.5} \log k)$。

例 5.9: 回顾例 5.6 中表 17 中的标签松弛 $\delta_1, \delta_2, \ldots, \delta_6$。当 $k = 8$，$\mu = 2$ 时，验证可知如图 28 所示叶子节点的查询松弛 $\Delta_1, \Delta_2, \ldots, \Delta_8$ 是 Q_2 的基于拓补排序函数的 top-8 查询松弛。算法 evalPR 首先构建最小匹配树，如图 28 所示。其中，第 $i+1$ 层节点的查询松弛是第 i $(i \in [0, 2])$ 层节点的查询松弛的最小匹配。例如，第 1 层的查询松弛 $\Delta_{11}, \Delta_{12}, \ldots, \Delta_{14}$ 是第 0 层查询松弛 $\Delta_1, \Delta_2, \ldots, \Delta_8$ 的最小匹配。其中，Δ_{11} 是 Δ_1 和 Δ_3 的合并。 □

（2）有界增量计算。当构建完毕 $\Delta_1, \ldots, \Delta_k$ 的最小匹配树 $\mathcal{T}$，算法 evalPR 根据 $\mathcal{T}$ 中顺序由树根到叶子节点为 $\mathcal{T}$ 中所有查询松弛对应的 Q 的松弛模式图计算其在数据图中的图

匹配查询结果。对于 $\mathcal{T}$ 中每个节点 u，evalPR 通过调用基于泛化图模拟的有界增量算法 BDeval（如下所示），从 u 的父节点 pre(u) 对应的松弛模式图的图匹配查询结果中有界增量的计算得出 u 对应的松弛模式图的图匹配查询结果。evalPR 最终返回 $\mathcal{T}$ 的所有叶子节点对应的松弛模式图的图匹配查询结果。

基于泛化图模拟的有界增量算法 BDeval 计算方式如下。如果在 $\mathcal{T}$ 中 Δ' 合并 Δ_l 和 Δ_r，则 BDeval 从 $(Q\oplus\Delta')(G)$ 中分别增量计算得出 $(Q\oplus\Delta_l)(G)$ 和 $(Q\oplus\Delta_r)(G)$。BDeval 是图匹配查询增量算法的扩展 [50]，其通过从 $(Q\oplus\Delta')(G)$ 中迭代去除经验证与 $Q\oplus\Delta_{l(r)}$ 不匹配的数据节点，从而得到 $(Q\oplus\Delta_{l(r)})(G)$。与 [50] 类似，算法 BDeval 的计算开销是有界的，因为 BDeval 的计算开销只由算法输入和输出的变化量 $|(Q\oplus\Delta')(G)\setminus(Q\oplus\Delta_{l(r)})(G)|$、以及算法必须维护的辅助数据结构的变化量决定，而不直接取决于 $|G|$。

例 5.10: 继续以例 5.9 中内容为例。算法 evalPR 根据如图 28 所示最小匹配树 $\mathcal{T}$ 中查询松弛的排列顺序执行增量图匹配查询计算过程。以 $Q_2\oplus\Delta_1$ 和 $Q_2\oplus\Delta_3$ 为例，算法 evalPR 从根节点出发首先计算得出 $Q_2\oplus\Delta_{31}$ 在 G_1 中的图匹配查询结果，再根据 $Q_2\oplus\Delta_{31}$ 的查询结果依次增量计算得出 $Q_2\oplus\Delta_{21}$，$Q_2\oplus\Delta_{11}$ 和 $Q_2\oplus\Delta_1$（$Q_2\oplus\Delta_3$）的图匹配查询结果。具体而言，（a）evalPR 计算得出 $Q_2\oplus\Delta_{31}$ 在 G_1 中的图匹配查询结果 C_1 为包含节点 newspaper$_1$ 的连通分量。而后，（b）evalPR 增量计算得知 C_1 也是 $Q_2\oplus\Delta_{21}$ 在 G_1 中的图匹配查询结果。之后，（c）evalPR 将节点 body_of_water 和节点 newspaper$_2$ 从 C_1 中去除，得到 $Q_2\oplus\Delta_{11}$ 在 G_1 中的图匹配查询结果 C_2。最后，（d）evalPR 将节点 castle 和 newspaper$_1$ 从 C_2 中去除，得到 $Q_2\oplus\Delta_1$ 在 G_1 中的图匹配查询结果，并且（e）evalPR 增量计算得知 C_2 也是 $Q_2\oplus\Delta_3$ 在 G_1 中的图匹配查询结果。 □

5.6　松弛图匹配查询结果溯源解释

章节 5.5 计算得出 top-k 松弛模式图在数据图中的图匹配查询结果。此时，一个自然的问题是为什么数据图中的某些节点可以被松弛模式图查询得出。本文提出松弛图匹配查询结果的溯源解释问题，通过捕捉松弛模式图中的关键松弛部分为用户解释图匹配查询结果为何能够由松弛模式图查询得出。本章内容构成图匹配查询松弛重构计算框架（章节 5.2.2）中步骤（4）的基础。

定义 5.12 (最小溯源解释): 给定模式图 Q，数据图 G，语义分类图 T，查询松弛 Δ，松弛模式图 $Q\oplus\Delta$ 在 G 中图匹配查询结果 $(Q\oplus\Delta)(G)$ 中的一个匹配节点 v ($v\in V_G$)。匹配节点 v 基于 Δ 的一个溯源解释是 Δ 的一个子集，记为 $\mathcal{E}_\Delta(v)$，使得 v 是 $(Q\oplus\mathcal{E}_\Delta(v))(G)$ 中的

匹配节点。直观理解，$\mathcal{E}_{\Delta}(v)$ 通过利用 Δ 中标签松弛解释为什么节点 v 是 $Q \oplus \Delta$ 在 G 中的匹配节点。存在特殊情况，当 v 已经是 Q 在 G 中的匹配节点，则 v 基于 Δ 的一个溯源解释为空集 $\emptyset$。 □

匹配节点 v 的基于 Δ 的溯源解释 $\mathcal{E}^m_{\Delta}(v)$ 是最小溯源解释当且仅当 $\mathcal{E}^m_{\Delta}(v)$ 在 v 的基于 Δ 的所有溯源解释中包含的标签松弛个数最少。直观理解，$\mathcal{E}^m_{\Delta}(v)$ 是 Δ 中能够使节点 v 由不匹配状态变为匹配状态的最少标签松弛集合。

例 5.11: 继续以例 5.10 中模式图 Q_2, 数据图 G_1，查询松弛 $\Delta_3 = \{\delta_1, \delta_2\}$ 为例。松弛模式图 $Q_2 \oplus \Delta_3$ 在 G_1 中的图匹配查询结果包括节点 {newspaper$_1$, television_show, castle, river, exhibition_hall, take_away_food, restaurant}。验证可知，（1）匹配节点 exhibition_hall 的基于 Δ_3 的最小溯源解释 $\mathcal{E}^m_{\Delta_3}$(exhibition_hall) 为 $\{\delta_1\}$，即 Δ_3 的一个子集。（2）匹配节点 castle 的基于 Δ_3 的最小溯源解释也是 Δ_3 的一个子集，即 $\mathcal{E}^m_{\Delta_3}$(castle) = $\{\delta_2\}$。 □

松弛图匹配查询结果最小溯源解释问题（MRE）。给定模式图 Q，数据图 G，语义分类图 T，Q 的基于 T 的 k 个查询松弛 $\Delta_1, \ldots, \Delta_k$ 以及相应松弛模式图在数据图中的图匹配查询结果 $(Q \oplus \Delta_1)(G), \ldots, (Q \oplus \Delta_k)(G)$，正整数 $i \in [1, k]$，和 $(Q \oplus \Delta_i)(G)$ 中的一个匹配节点 v。MRE 问题即找到匹配节点 v 的基于查询松弛 Δ_i 的最小解释 $\mathcal{E}^m_{\Delta_i}(v)$。

在生成模式图的 top-k 查询松弛（章节 5.4）并计算完毕其对应的松弛模式图在数据图中的图匹配查询结果后（章节 5.5），MRE 问题通过捕捉松弛模式图中的关键松弛部分使得用户知道数据图中的匹配节点为何能由松弛模式图查询得出。然而，这个问题的固有复杂度很高。

定理 5.5: MRE 的判定问题是 NP-完全问题。 □

尽管 MRE 问题计算复杂度很高，本文在仔细分析问题特征的基础上为 MRE 设计了实用算法。本文对 MRE 问题的两个实例 MRE$_{\text{TF}}$ 和 MRE$_{\text{DF}}$ 分别进行了研究，分别用于解释由章节 5.4 中算法 relTF 和算法 relDF 生成的 top-k 查询松弛对应的松弛模式图的图匹配查询结果。

（I）基于拓补排序函数的最小溯源解释问题（MRE$_{\text{TF}}$）。当解释基于拓补排序函数生成的松弛查询图在数据图中的图匹配查询结果时，MRE 问题变的易于处理。基于以下性质，可以在线性时间内计算得出 MRE$_{\text{TF}}$ 问题的最小溯源解释。

定理 5.6: 对于任意模式图 Q，数据图 G，语义分类图 T，算法 relTF 计算得出的 top-k 查

询松弛 $\Delta_1, \ldots, \Delta_k$，任意正整数 $i \in [1,k]$ 和任意 $(Q \oplus \Delta_i)(G)$ 中节点 v，必然存在一个正整数 $j \in [1,k]$，使得 Δ_j 是 v 的基于 Δ_i 的最小溯源解释。 □

定理 5.6 说明基于拓补排序函数的 top-k 查询松弛生成的图匹配查询结果的最小溯源解释肯定是 top-k 查询松弛其中之一。定理 5.6 同时提供了解决 $\mathsf{MRE}_{\mathsf{TF}}$ 问题的算法，一个不需要访问 G 和 T 的线性时间复杂度算法，记为 expTF。算法 expTF 工作流程如下：线性扫描 $(Q \oplus \Delta_1)(G), \ldots, (Q \oplus \Delta_k)(G)$，返回包含标签松弛个数最少的 Δ_j，并且满足 v 是 $(Q \oplus \Delta_j)(G)$ 中的匹配节点。值得注意的是算法 expTF 是最优算法，因为其只需扫描一遍输入即可输出结果。

（II）基于多元化拓补排序函数的最小溯源解释问题（$\mathsf{MRE}_{\mathsf{DF}}$）。下面研究如何为基于多元化拓补排序函数生成的松弛查询图在数据图中的图匹配查询结果计算溯源解释。由于 $\mathsf{kPR}_{\mathsf{DF}}$ 问题中的多元化排序函数，定理 5.6 中的性质在 $\mathsf{MRE}_{\mathsf{DF}}$ 问题中不再成立。

尽管 MRE 问题的计算复杂度较高，本文为其设计了实用的算法，记为 expDF，适用于包括 $\mathsf{MRE}_{\mathsf{DF}}$ 在内的 MRE 问题的所有实例。算法 expDF 是精确度由时间复杂度决定的参数化算法。算法 expDF 的输入参数 M 表示在计算 v 的基于 Δ_i 的最小溯源解释 $\mathcal{E}_{\Delta_i}(v)$ 的过程中，最多可以调用 M 次图匹配查询算法。通过调整 M 可以在算法输出结果精确度（$\mathcal{E}_{\Delta_i}(v)$ 的大小）和算法时间复杂度之间做任意平衡：较大的 M 可以计算得出较小的 $\mathcal{E}_{\Delta_i}(v)$，越小的 $\mathcal{E}_{\Delta_i}(v)$ 越准确。

算法 expDF(M) 的工作流程分为以下两步。

（1）在 top-k 个查询松弛中查找个数最小的查询松弛 Δ_j ($j \in [1,k]$)，并且满足（a）$\Delta_j \subseteq \Delta_i$ 和（b）$v \in (Q \oplus \Delta_j)(G)$ 两个条件。

（2）按照 $\mathsf{cand}_{(G,T)}(l')$ 值递减顺序对查询松弛 Δ_j 中每个标签松弛 $\delta = l \to l'$ 执行下述检测。如果在章节 5.5 中多查询处理优化算法 evalPR 没有为 $Q \oplus (\Delta_j \setminus \{\delta\})$ 计算其在 G 中的图匹配查询结果 $(Q \oplus (\Delta_j \setminus \{\delta\}))(G)$，则调用图匹配查询算法（例如基于泛化图模拟的图匹配查询算法 TSim，章节 5.2.1）计算该图匹配查询结果，并判断节点 v 是否是 $(Q \oplus (\Delta_j \setminus \{\delta\}))(G)$ 中匹配节点。如果是其中匹配节点，则从 Δ_j 中去除 δ。算法返回 Δ_j 作为 v 的基于 Δ_i 最小溯源解释，直至已经检查完毕 Δ_j 中所有剩余标签松弛或者已经调用了 M 次图匹配查询算法。

5.7 实验评估

本章采用两个真实数据集，设计有效性测试和效率测试实验来评估图匹配查询松弛重构方法的性能。

5.7.1 实验设置

本章首先介绍实验设置和数据集等实验相关信息。

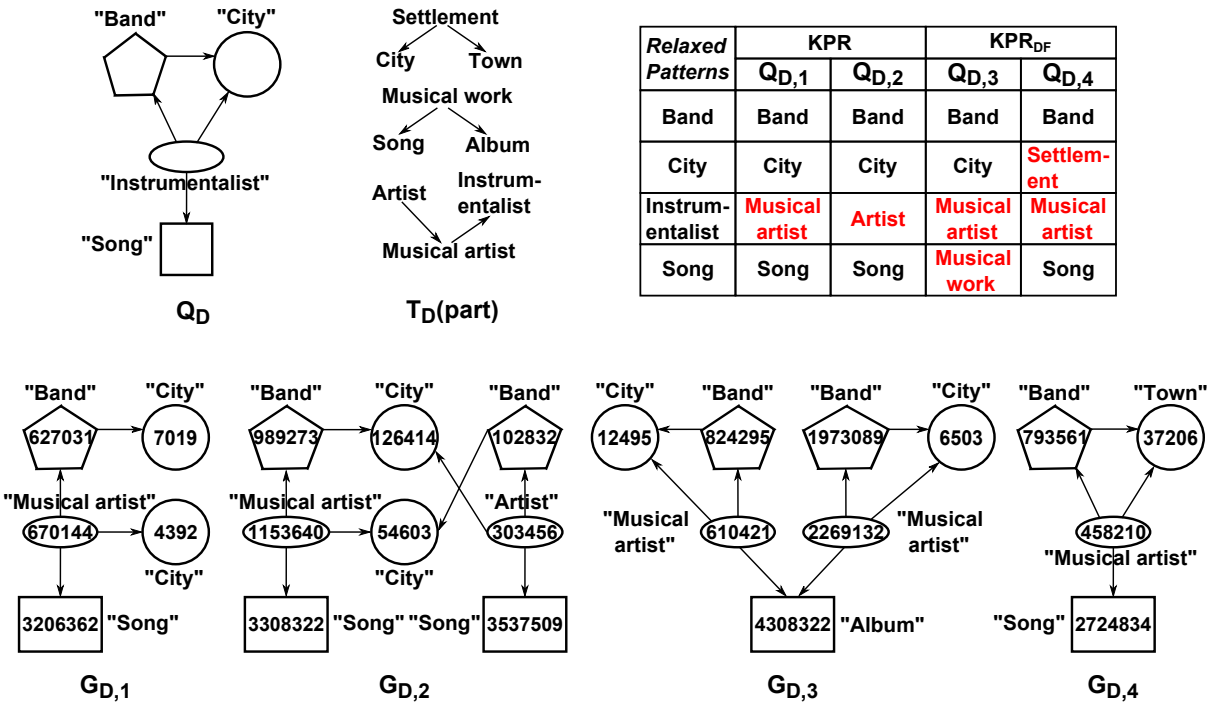
数据图。本章采用两个真实数据集和一个合成数据集来评估方法的性能。

（1）知识图谱 DBpedia 版本号为 DBpedia 201504 [75]。其包括（i）数据图，由 4.43M 个节点和 8.43M 条边组成，和（ii）语义分类图，由 735 个标签节点组成，边表示标签之间的“是一种”关系。语义分类图是有根森林状数据图，平均高度为 2.29，最大高度为 6。（2）知识图谱 YAGO [110] 包括（i）数据图，由 5.13M 个节点和 5.39M 条边组成，和（ii）语义分类图，由 6488 个标签节点组成。语义分类图平均高度为 3.27，最大高度为 13。

模式图。本章通过实现一个模式图自动生成器来生成模式图。自动生成器由 3 个参数控制，从而生成不同的模式图：$|V_Q|$ 控制模式图节点个数，由 2 逐步变化至 10；$|E_Q| = \alpha|V_Q|$ 控制模式图边的个数，标签函数 l_Q 从实验用数据图字表 Σ 中为每个节点选取一个标签。

算法。本章用 C++ 实现了以下算法：（1）本文设计的算法 TSim，relTF，relDF，evalPR，expTF 和 expDF；（2）算法 evalTF 通过利用算法 evalPR 计算 kPR 问题生成的 top-k 松弛模式图的图匹配查询结果；算法 evalDF 通过利用算法 evalPR 计算 kPR_{DF} 问题生成的 top-k 松弛模式图的图匹配查询结果；（3）算法 TSimTF 通过调用 k 次算法 TSim 计算 kPR 问题生成的 top-k 松弛模式图的图匹配查询结果；算法 TSimDF 通过调用 k 次算法 TSim 计算 kPR_{DF} 问题生成的 top-k 松弛模式图的图匹配查询结果；（4）对比算法 gSim [13]，基于图模拟的图匹配查询算法；（5）对比算法 relC，根据松弛模式图和原始模式图之间所有标签的松弛距离之和对查询松弛进行排序，生成 top-k 查询松弛 [114]；（6）对比算法 TSimC 通过调用 k 次算法 TSim 计算由算法 relC 生成的 top-k 松弛模式图的图匹配查询结果。

所有实验都是在一台配置为 Intel Core(TM) Duo 3.00GHz CPU and 16GB of memory 的机器上测试。本章为所有的实验设定固定参数 $\lambda = 0.5$，设定默认参数 $|V_Q| = 6$，$k = 15$，$\mu = 3$。当生成模式图时，设定固定参数 $\alpha = 1.2$。每组实验重复 10 次，并取平均数记录于以下实验结果章节中。



Relaxed Patterns	KPR		KPR_{DF}	
	$Q_{D,1}$	$Q_{D,2}$	$Q_{D,3}$	$Q_{D,4}$
Band	Band	Band	Band	Band
City	City	City	City	Settlem-ent
Instrum-entalist	Musical artist	Artist	Musical artist	Musical artist
Song	Song	Song	Musical work	Song

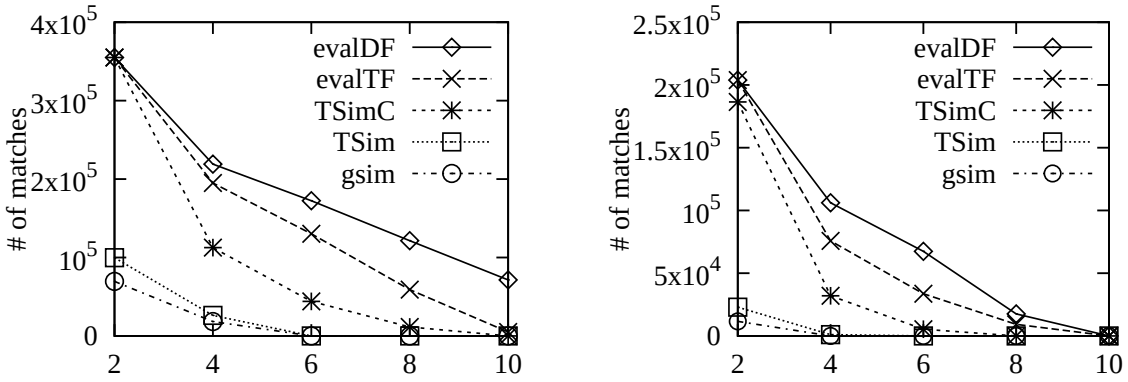
图 29　知识图谱 DBpedia 中基于查询松弛的图匹配查询结果案例分析

5.7.2　有效性测试

（1）案例分析。本章首先通过实验案例分析图匹配查询松弛重构方法查询结果的质量。

以一个真实应用模式图 Q_D 为例，如图 29 所示。模式图 Q_D 期望从 DBpedia 的数据图中找到满足以下条件的所有“城市”节点：（a）该“城市”是某些“乐器演奏家”的出生地，（b）“乐器演奏家”在该“城市”建立过“乐队”，并且（c）“乐器演奏家”发行过“歌曲”。DBpedia 的模式图和数据图中的节点是具有唯一 ID 和标签的实体，标签用于表示实体的类别。图 29 中的图节点根据不同分类标签用不同的形状表示，包括圆形、椭圆形、方形和五边形。模式图 Q_D 和数据图中只有形状相同的节点才存在匹配可能。

通过为 Q_D 在 DBpedia 上执行图匹配查询计算，实验发现 DBpedia 中不存在 Q_D 的基于泛化图模拟的图匹配查询结果。更不用说匹配要求条件更为严格的泛化子图同构语义、子图同构语义和图模拟语义，这些查询语义均不存在 Q_D 的图匹配结果。下面分析 Q_D 的查询松弛。通过使用 DBpedia 的语义分类图 T_D，如图 29 所示，实验计算出基于拓补排序函数的 top-2 查询松弛 $Q_{D,1}$ 和 $Q_{D,2}$，以及基于多元化拓补排序函数的 top-2 查询松弛 $Q_{D,3}$ 和 $Q_{D,4}$。四个松弛模式图的部分图匹配查询结果分别为 $G_{D,1}$、$G_{D,2}$、$G_{D,3}$ 和 $G_{D,4}$。通过验证得知，查询结果中的所有匹配节点都能够满足 Q_D 的查询意图，是有意



(a) 变化 $|V_Q|$：DBpedia　　(b) 变化 $|V_Q|$：YAGO

图 30　松弛图匹配查询结果数量测试：变化 $|V_Q|$

义的图匹配查询结果。然而，在不使用松弛重构方法时这些节点无法被 Q_D 查询得出。

本文认为造成这种状况的原因是用户无法完全熟悉知识图谱中数目庞大的分类标签信息，例如 DBpedia 有 735 个内置分类标签，YAGO 有 6488 个内置分类标签，导致用户无法准确的用标签描述查询意图。

（2）泛化图模拟有效性测试。本节通过对基于泛化图模拟语义的图匹配查询结果进行分析，说明需要进一步使用松弛重构方法提高图匹配查询结果数量的必要性。

（*I*）查询结果数量测试。为了证明需要使用松弛重构方法的必要性，本节首先测试普通图匹配查询方法查找匹配结果的能力。通过改变模式图 Q 的节点大小 $|V_Q|$，使其从 2 逐步变化至 10，分别在知识图谱 DBpedia 和 YAGO 上评估由算法 TSim 和算法 gSim 输出匹配节点的平均数目。知识图谱 DBpedia 上的实验结果如图 30a 所示，知识图谱 YAGO 上的实验结果如图 30b 所示。实验发现即使对于中等规模的模式图，TSim 和 gSim 只能找到极少匹配结果。例如，当 $|V_Q| \geq 6$ 时，两个算法在两个知识图谱中均不能找到任何图匹配结果。对于较小规模的模式图，算法 TSim 比算法 gSim 找到更多图匹配结果。例如，当 $|V_Q| = 4$ 时，TSim 在 DBpedia 中找到的图匹配结果是 gSim 找到的图匹配结果的 1.4 倍。这些实验数据证明了需要使用松弛重构方法提高图匹配查询结果数量的必要性。

（*II*）查询结果质量测试。为了证明以泛化图匹配查询语义为基础进行松弛重构的可行性，本节评估基于泛化图模拟的图匹配查询方法找到图匹配结果的质量。通过从两个知识图谱的数据图中随机采样出 30 个子图作为模式图。对于每个模式图 Q，本文只保留 Q 中具有清晰查询意图的节点，例如满足某些条件的地点和书籍等。本文将这些查询意图作为判定模式图查询出的匹配节点是否为合理有效匹配节点的黄金定律。本文将 Q

在数据图 G 中的图匹配查询结果 S 的准确率（有效性）定义如下：

$$\text{acc}(S,Q,G) = \sum_{(u,v)\in S} \text{valid}(u,v)/|S| \tag{5.12}$$

其中，如果 G 中一个节点 v 表示的实体满足 Q 中节点 u 表示的查询意图（使用 WordNet 和 Wiki 分辨同义实体），则 $\text{valid}(u,v) = 1$；反之，$\text{valid}(u,v) = 0$。分析可知 $\text{acc}(S,Q,G)$ 的值属于 $[0,1]$，如果 S 中结果全部是准确结果，则 $\text{acc}(S,Q,G)$ 为 1。为了能够人工检查图匹配查询结果的准确率，本文限制 $|S| \leq 50$：如果 Q 在 G 中的图匹配查询结果 $Q(G)$ 中的匹配数据节点数目大于 50，则从 $Q(G)$ 中采样出一个子集 $S \subseteq Q(G)$ 使得 $|S| = 50$，只需检测 S 的准确率即可；否则，令 $S = Q(G)$ 即可。

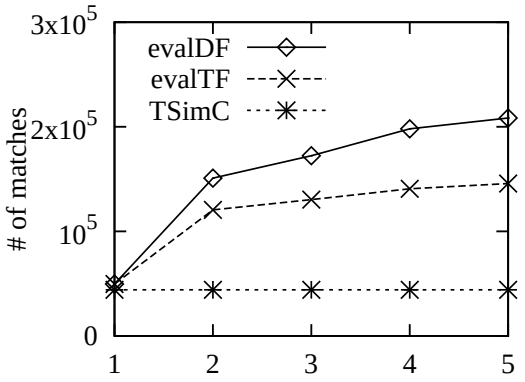
通过使用准确率衡量标准检测由算法 TSim 输出的 30 个模式图的图匹配查询结果。实验发现图匹配查询结果在 DBpedia 上的平均准确率达到 0.98；图匹配查询结果在 YAGO 上的平均准确率达到 0.94。这些数据证明了泛化图匹配查询语义可以查询得到比较准确的图匹配查询结果，从而可以作为图匹配查询松弛重构方法的基础查询语义。

（3）图匹配查询松弛重构有效性测试。本节对图匹配查询松弛重构方法有效性进行分析。

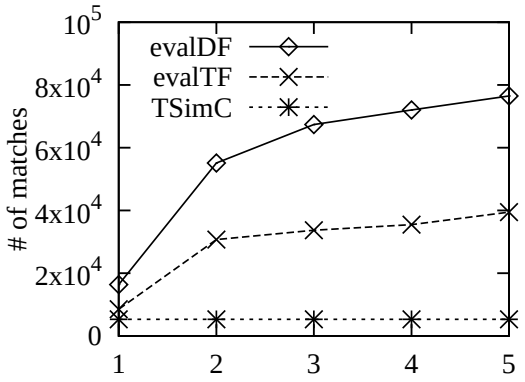
（*I*）查询结果质量测试。本节评估由松弛模式图查询得到的匹配结果的质量。通过使用与实验 (2)-(II) 相同的实验设置检测 30 个模式图的 top-5 松弛模式图的图匹配查询结果。实验发现基于拓补排序函数，图匹配查询结果在 DBpedia 上的平均准确率达到 0.81，在 YAGO 上的平均准确率到 0.75；基于多元化拓补排序函数，图匹配查询结果在 DBpedia 上的平均准确率达到 0.72，在 YAGO 上的平均准确率达到 0.68。

（*II*）查询结果数量测试。算法 relTF、relDF 和 relC 分别根据不同排序函数生成不同松弛模式图，本节通过比较由算法 evalTF、evalDF 和 TSimC 计算出的上述不同松弛模式图在数据图中的图匹配查询结果数目，评估不同松弛重构方法查找图匹配结果的能力。

（*a*）$|V_Q|$ 对查询结果数量的影响。通过改变模式图 Q 的节点大小 $|V_Q|$，使其从 2 逐步变化至 10，测量查询结果的平均数目，评估 $|V_Q|$ 对查询结果数量的影响。知识图谱 DBpedia 上的实验结果如图 30a 所示，知识图谱 YAGO 上的实验结果如图 30b 所示。实验发现如下所示：（i）算法 evalTF、evalDF 和 TSimC 找到的图匹配查询结果数量相对算法 gSim 找到的图匹配查询结果数量在所有情况下都有显著提升。例如当 $|V_Q| = 4$ 时，evalTF、evalDF 和 TSimC 分别是 gSim 找到图匹配查询结果数量的 10.7、11.9 和 6.1 倍。（ii）在所有情况下，算法 evalTF 和 evalDF 始终比算法 TSimC 找到更多图匹配查询结果。这验证了拓补排序函数和多元化拓补排序函数中信息率的有效性，信息率能够充分利用数据

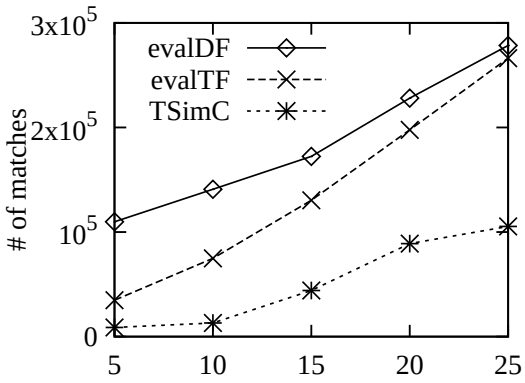


(a) 变化 μ：DBpedia

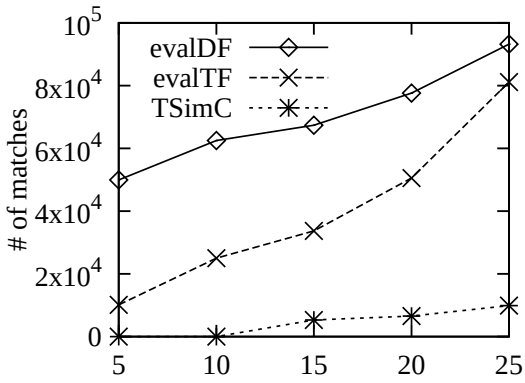


(b) 变化 μ：YAGO

图 31　松弛图匹配查询结果数量测试：变化 μ



(a) 变化 k：DBpedia



(b) 变化 k：YAGO

图 32　松弛图匹配查询结果数量测试：变化 k

图中信息对查询松弛进行排序。（iii）算法 evalDF 比算法 evalTF 找到更多图匹配查询结果。这是因为多元化拓补排序函数倾向于生成松弛距离较大的标签松弛，从而同时优化拓补排序函数值和查询松弛集合的多样性。

（b）μ 对查询结果数量的影响。通过改变 μ 的大小，使其从 1 逐步变化至 5，测量查询结果的平均数目，评估 μ 对查询结果数量的影响。知识图谱 DBpedia 上的实验结果如图 31a 所示，知识图谱 YAGO 上的实验结果如图 31b 所示。实验发现如下所示：（i）即使当 μ 很小时，算法 evalTF 和 evalDF 都能够在两个数据集中找到合理有效的图匹配查询结果，并且结果数量多于算法 TSimC 找到的图匹配查询结果数量。例如，当 $\mu = 1$ 时，evalTF 和 evalDF 分别在 YAGO 中找到 8525 和 16351 个图匹配节点，多于算法 TSimC 找到的 5320 个图匹配节点。（ii）算法 evalTF 和 evalDF 能够受益于较大的 μ 值，因为较大的 μ 值使得 evalTF 和 evalDF 能够更充分的利用数据图和语义分类图中的信息。例

表 21　准确溯源解释所占百分比

M	**0**	**1**	**2**	**3**	**4**
DBpedia	85%	93%	99%	100%	100%
YAGO	88%	97%	100%	100%	100%

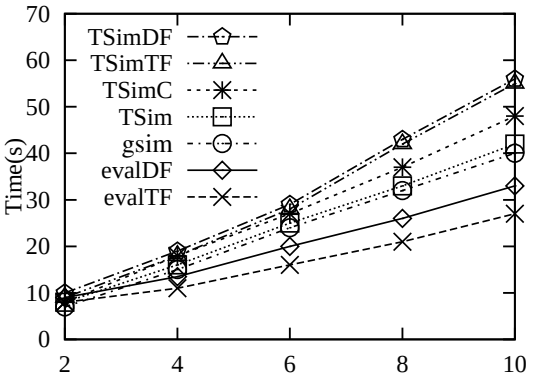
如，当当 $\mu = 5$ 时，evalTF 和 evalDF 分别在 YAGO 中找到 39478 和 76502 个图匹配节点，而算法 TSimC 仍然只找到 5320 个图匹配节点。

（c）k 对查询结果数量的影响。通过改变 k 的大小，使其从 5 逐步变化至 25，测量查询结果的平均数目，评估 k 对查询结果数量的影响。知识图谱 DBpedia 上的实验结果如图 32a 所示，知识图谱 YAGO 上的实验结果如图 32b 所示。实验发现即使当 k 很小时，算法 evalTF 和 evalDF 都能够在两个数据集中找到合理有效的图匹配查询结果。例如，当 $k = 5$ 时，evalTF 在 YAGO 中找到 10100 个图匹配节点，算法 evalDF 找到了更多图匹配节点，而算法 TSimC 无法找到任何图匹配节点。

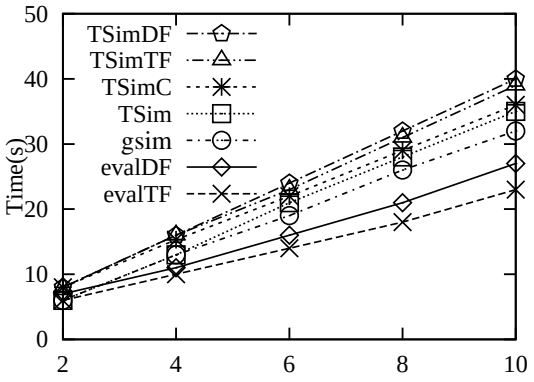
（4）松弛图匹配查询结果溯源解释有效性测试。为了评估松弛图匹配查询结果溯源解释方法的有效性，本文分别从算法 relTF 和 relDF 输出的松弛模式图的图匹配结果中随机选择出 100 个匹配节点，并分别使用算法 expTF 和 expDF（参数 M 由 0 变化至 4）计算每个匹配节点的最小溯源解释。本文通过定义溯源解释的准确率来评估溯源解释方法的有效性。溯源解释的准确率定义为算法 expTF 或 expDF 输出的所有最小溯源解释中真正最小溯源解释所占百分比。由于算法 expTF 的准确率始终为 1，本节只测试算法 expDF 的准确率，其在知识图谱 DBpedia 和 YAGO 上的实验结果如表 21 所示。实验发现算法 expDF 可以很好的解释松弛图匹配查询结果。即使当 $M = 0$ 时，算法 expDF 在 DBpedia 上的准确率始终高于 85%，在 YAGO 上的准确率始终高于 88%；当 $M \geq 2$ 时，算法 expDF 在两个数据集上的准确率均高于 99%。

5.7.3　效率测试

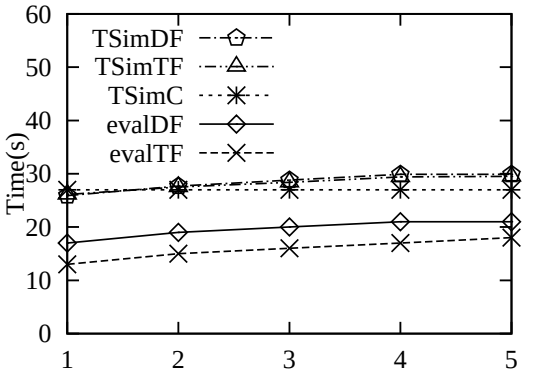
（1）查询松弛排序算法和多查询处理优化算法效率测试。本节测试了图匹配查询松弛排序算法和松弛图匹配查询多查询处理优化算法的效率。由于所有图匹配查询松弛排序算法 relTF、relDF 和 relC 在所有情况下都会在 2s 内终止运行，所以本文只测试松弛图匹配查询多查询处理优化算法的效率。在预处理过程中，将知识图谱 DBpedia 和 YAGO 的语义分类图中所有分类标签编码为比特串的运行时间分别为 0.02s 和 2.97s；为语义分类



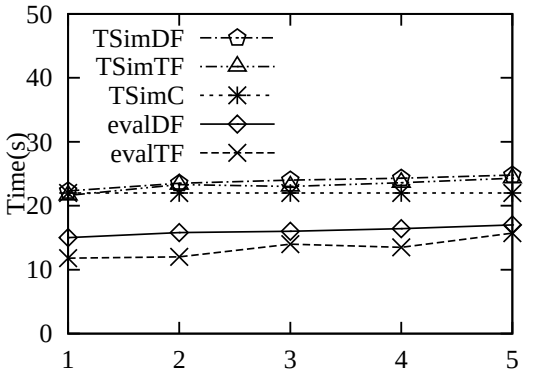
(a) 变化 $|V_Q|$：DBpedia



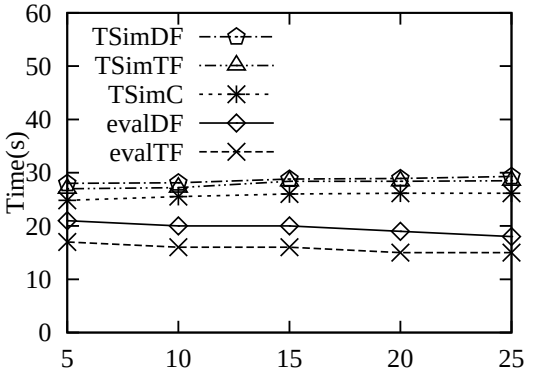
(b) 变化 $|V_Q|$：YAGO



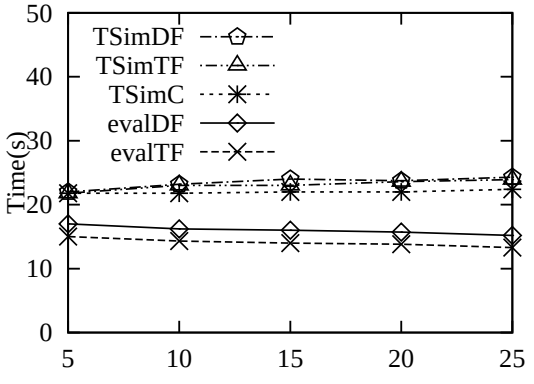
(c) 变化 μ：DBpedia



(d) 变化 μ：YAGO



(e) 变化 k：DBpedia

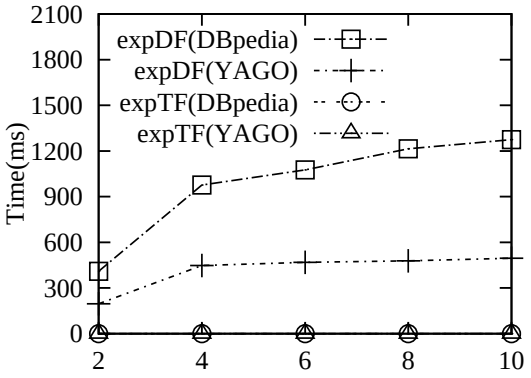


(f) 变化 k：YAGO

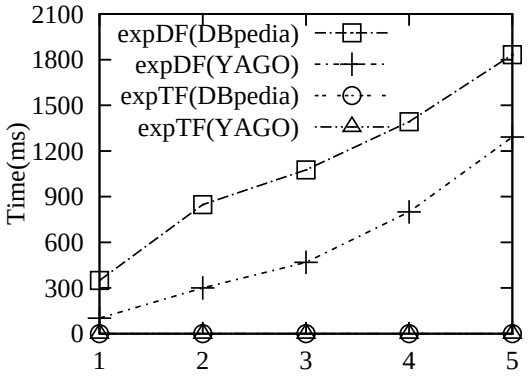
图 33　松弛图匹配查询多查询处理优化算法效率测试

图中每个分类标签 l 统计其在 DBpedia 和 YAGO 的数据图中对应节点个数 $|\mathsf{cand}_{(G,T)}(l)|$ 的运行时间分别为 7.38s 和 6.21s。

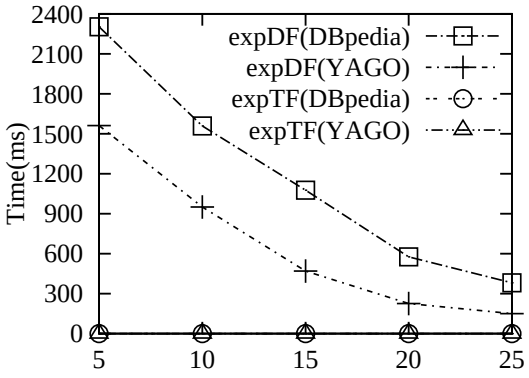
本节使用与效果测试实验 (3)-(II) 相同的实验设置，通过比较算法 evalTF、evalDF、TSimTF、TSimDF 和 TSimC 的运行时间，评估图匹配查询多查询处理优化算法的效率。



(a) 变化 $|V_Q|$



(b) 变化 μ



(c) 变化 k

图 34　松弛图匹配查询结果溯源解释算法效率测试

知识图谱 DBpedia 上的实验结果如图 33a，33c，33e 所示，知识图谱 YAGO 上的实验结果如图 33b，33d，33f 所示。实验发现如下所示：（i）算法 evalTF 和 evalDF 显著提升了算法 TSimTF 和 TSimDF 的效率。例如，当 $k = 15$，$|V_Q| = 6$，$\mu = 3$ 时，算法 evalTF 在知识图谱 DBpedia 上的运行时间比算法 TSimTF 的运行时间快 1.8 倍。当 $|V_Q|$ 或 k 变大时，上述效率提升幅度显著增加。（ii）算法 TSimTF 和 TSimDF 的运行时间多于算法 TSimC 的运行时间。这是因为算法 TSimTF 和 TSimDF 的排序函数考虑了数据图信息，所以相较于算法 TSimC，算法 TSimTF 和 TSimDF 能够查询得到更多有效的图匹配结果。即使中等规模的模式图，算法 TSimC 也经常无法找到任何图匹配查询结果。例如，当 $|V_Q| = 6$ 时，算法 TSimC 在 YAGO 上的图匹配查询结果为空集。

（2）松弛图匹配查询结果溯源解释算法效率测试。本节使用与效果测试实验 (4) 相同的实验设置并设置默认值 $M = 4$，通过比较算法 expTF 和 expDF 在 DBpedia 和 YAGO 上的运行时间，评估松弛图匹配查询结果溯源解释算法的效率。知识图谱 DBpedia 和 YAGO

上的实验结果如图 34a，34b，34c 所示。实验发现如下所示：（i）算法 expTF 和 expDF 都能够高效的计算出松弛图匹配查询结果的最小溯源解释。算法 expTF 在所有情况下的运行时间少于 1ms，算法 expDF 在所有情况下的运行时间少于 2.4s。（ii）算法 expDF 的运行时间随着 $|V_Q|$ 或 μ 值的增大而增大，与预期一致。expDF 的运行时间随着 k 值的增大而减小。这是因为当 k 取较大值时会生成更多松弛模式图，所以 expDF 调用算法 TSim 的次数会降低。

本文还发现较小的 λ 值会生成能够找到更多图匹配查询结果的查询松弛。例如，在所有情况下，算法 evalDF（$\lambda = 0.5$）比算法 evalTF（$\lambda = 1$）找到数目更多的图匹配查询结果。这是因为为了增加查询松弛集合的多元性，算法 evalDF 倾向于生成松弛距离较大即具有更广义松弛标签的查询松弛。相应的，evalDF 的运行时间长于 evalTF 的运行时间。

5.7.4 实验总结

实验结果表明，图匹配查询松弛重构计算方法可以有效的和高效的查询得到更多合理的图匹配查询结果并为松弛图匹配查询结果提供溯源解释。（1）基于泛化图模拟语义的图匹配查询方法在知识图谱 DBpedia 上能够查询得到 1.4 倍于基于图模拟语义的图匹配查询方法的查询结果，并且两种方法的运行时间基本相等。（2）当模式图 Q 的节点个数为 4 时，基于泛化图模拟语义的图匹配查询松弛重构方法在 DBpedia 上能够查询得到 11.9 倍于普通图匹配查询方法的查询结果，并且其中 74% 的结果被证明是合理有效的查询结果。（3）当 k=15 时，松弛图匹配查询多查询处理优化方法比传统图匹配查询处理方法快 1.8 倍，并且效率提升幅度随 k 增大而增加。（4）松弛图匹配查询结果溯源解释方法可以准确的和高效的计算得出图匹配查询结果的最小溯源解释。在 DBpedia 和 YAGO 上，溯源解释方法的准确率在不访问数据图的前提下均可以达到 85%，在允许 2 次访问数据图的前提下均可以达到 99%，并且在所有情况下运行时间均少于 2.4s。

5.8 相关工作分析

图匹配查询语义。本文在章节 2.1 中详细介绍了图匹配查询的主要匹配语义。其中，最基础的图匹配查询语义是子图同构语义。然而，基于子图同构的图匹配查询问题是NP-完全问题，严格的查询条件经常导致模式图在数据图中的查询结果较少甚至为空集。为了使图匹配查询方法能够找到更多有意义的图匹配结果，一个方法是通过采用图模拟及其扩展作为图匹配查询语义来放松子图同构严格的结构匹配约束条件 [13–15, 28]; 另一个

方法是通过引入本体特征信息或标签分类信息来放松子图同构严格的标签匹配约束条件 [108, 114, 115]。然而，这两种方法均不能从根本上解决图匹配查询方法的空集问题。

查询松弛重构。与本课题最相关的工作是由 [72] 提出的基于结构松弛的图匹配查询松弛重构技术。本文在章节 2.3.4 中详细介绍了这项工作。本课题与这项工作有以下几处不同：（1）现有工作通过改变模式图结构，即减少模式图中的节点和边来查询得到更多匹配结果。然而，图匹配查询的目标是查询与模式图结构匹配的子图，修改模式图结构意味着从根本改变了用户查询意图。本文提出了能够保持原有模式图结构的图匹配查询松弛重构方法，通过引入语义分类信息来松弛模式图节点标签从而查询得到更多有意义的匹配结果。（2）本文提出了通用的图匹配查询松弛重构计算框架，包括（a）松弛图匹配查询多查询处理优化方法通过最大化共享计算量，从而同时、快速的计算得出 top-k 松弛模式图的图匹配查询结果；以及（b）松弛图匹配查询结果溯源解释方法为用户提供新结果如何产生的溯源解释。以上技术在现有查询松弛重构研究工作中均未被考虑。

多查询处理优化。多查询处理优化方法在图匹配查询 [116] 及 SPARQL 查询 [117] 等领域已有相关研究工作。现有工作通过分解多个输入查询并共享公共查询部分的计算量来提高多查询处理的效率。本课题采用现有多查询处理优化工作的一般思路来同时计算 top-k 松弛模式图的图匹配查询结果。本课题与现有工作的不同之处在于，现有图匹配查询和 SPARQL 查询的多查询处理优化问题均为 NP-完全问题，而本课题提出的松弛图匹配查询多查询处理优化问题是 PTIME 问题。此外，本课题通过利用有界增量算法实现共享计算过程，进一步提高多查询处理效率。

查询解释。与松弛图匹配查询结果溯源解释方法相关的研究工作包括弹性解释 [118]、现象解释 [119] 和溯源解释 [120] 等。本课题与上述工作的不同之处在于，上述研究工作是针对关系数据查询的解释，而本课题研究松弛图匹配查询的溯源解释。

5.9　小结

本文提出了通用的图匹配查询松弛重构计算框架，适用于任意一种图匹配查询语义。本文将语义分类信息引入图匹配查询过程提出泛化图匹配查询，并基于泛化图匹配查询提出图匹配查询松弛重构概念。本文设计了松弛排序函数用于对模式图的查询松弛进行排序。本文还设计了高效实用的算法用于生成 top-k 查询松弛，计算 top-k 松弛模式图的图匹配查询结果，以及溯源解释松弛图匹配查询结果。最后，本文通过实验证明了图匹配查询松弛重构计算方法的有效性和易用性。

第六章　总结与展望

6.1　本文工作总结

互联网时代，社交网络与电子商务等网络和应用的升级推动了社会计算的发展，产生了以 PB 为数量级的非结构化数据信息。自关系数据模型理论被提出后，由于其具有简单明确的模型特征和坚实的理论基础，建立于关系数据模型上的关系数据库很快被广泛应用。然而，随着互联网的飞速发展，关系数据库逐渐暴露出其固有的缺陷和问题。现如今，网络及商业应用生成了大量信息互联的非结构化数据，每个实体都与周围实体存在广泛的联系关系，这些联系关系里存在大量的潜在信息。然而关系数据模型更加注重刻画实体内部的属性，实体间的关系主要通过外键来实现。这种存储模式使得关系数据库已经越来越难以承载查询海量数据深层次关系的需求。

与关系数据模型相比，图在描述和刻画复杂的现实世界方面具有更强的表达能力。图可以将现实世界中的实体抽象为节点，将实体间的关系抽象为节点之间的边，从而将现实世界直观建模。相较于结构化关系数据模型，非结构化图数据模型具有更强的动态性和可扩展性，可以承载海量数据存储、数据量迅猛增长，数据来源种类不断增多等带来的挑战。

在“大数据”时代，如何从海量数据中获取信息，从而进行快速、准确的判断和决策已经成为各类应用中最迫切的需求。图数据在各个领域日益增长的应用自然而然使得图查询（从图中查询信息）成为学术界和工业界的共同研究热点。图查询主要包括点查询、路径查询和图匹配查询。其中，图匹配查询尚处于研究的初始阶段，存在准确性低、效率低和易用性差等问题。本文致力于提高图匹配查询实用能力的研究。本文以现有图匹配查询语义和算法为基础，设计了新的与语义无关的“通用图匹配查询计算方法”，能够提供准确的、高效的和用户友好的查询方法，并且能够支持现有的所有图匹配查询语义。

本文主要贡献如下：

（1）本文提出了基于视图的近似图匹配查询方法，该方法适用于任意一种图匹配查询语义。本文利用该方法为模式图生成基于视图的上界近似查询和下界近似查询，并分析了基于视图的上界和下界近似查询的性质。本文对方法中的八个基本问题，即基于视图的（完全）上界/下界近似查询的存在性/最优化问题，从计算复杂性和可近似性两个方面进行了理论分析。根据理论分析结论，本文设计了高效的精确算法、具有近似度保

证的近似算法和实用的启发算法计算基于视图的最优（完全）上界和下界近似查询。本文在论述该方法工作原理时以图模拟为图匹配查询语义，并将该方法扩展至子图同构图匹配查询语义以验证方法的通用性。基于上述研究，本文抽象出基于视图的近似图匹配查询方法的通用计算框架。本文通过实验验证了基于视图的近似图匹配查询方法的查询效率优于基于视图的图匹配查询方法以及传统的图匹配查询方法。总而言之，本文提出的方法将基于视图的精确图匹配查询扩展为近似图匹配查询，提高了视图的命中率和利用率，从而提高图匹配查询引擎的效率。

（2）本文首先提出了组合模拟图匹配语义和基于组合模拟的 top-*k* 图匹配查询语义。该语义具有更高的实用价值，能够表达社交推荐和团队搜索等实际应用的查询需求。本文为基于组合模拟的 top-*k* 图匹配查询语义设计了批处理算法。本文提出了动态图匹配查询问题，考虑连续混合的模式图更新和数据图更新。本文首先对动态图匹配查询问题的计算复杂性进行了理论分析。尽管由理论分析得知问题的计算复杂度较高，本文设计了基于模式图划分和标识影响区域的增量计算策略，从而将模式图和数据图的更新影响范围最小化和局域化。基于增量计算策略，本文提出统一的图匹配查询增量计算框架，可以处理连续的单独或混合的模式图和数据图的连续更新。该计算框架是通用计算方法，适用于 top-*k* 图匹配查询和一般图匹配查询方法以及任意一种图匹配查询语义。基于图匹配查询的增量计算框架，本文设计了统一的增量算法来处理单独或混合的模式图和数据图更新。本文通过实验验证了 top-*k* 图匹配查询批处理算法的有效性，并且图匹配查询增量计算方法相对批处理算法效率有显著提升。

（3）本文提出了通用的图匹配查询松弛重构计算方法。本文将语义分类信息引入图匹配查询过程提出了泛化图匹配查询。基于泛化图匹配查询，本文提出通用的图匹配查询松弛重构计算框架。本文设计了拓补排序函数和多元化拓补排序函数用于生成模式图的 top-*k* 查询松弛。这两个函数综合考虑了泛化图匹配查询特征、语义分类图特征、以及模式图和数据图之间的相关关系特征。本文对根据拓补排序函数和多元化拓补排序函数生成模式图 top-*k* 查询松弛问题进行了理论分析。基于理论分析结论，本文设计了用于生成 top-*k* 查询松弛的高效实用算法。本文提出松弛图匹配查询多查询处理优化方法通过最大化共享计算量，从而同时、快速的计算得出 top-*k* 松弛模式图的图匹配查询结果。本文提出松弛图匹配查询结果的最小溯源解释问题，通过捕捉松弛模式图中的关键松弛部分来解释图匹配结果为何能够由松弛模式图查询得出。本文通过实验验证了图匹配查询松弛重构计算方法能够有效的从数据图中查询出更多合理的图匹配查询结果。

6.2 未来研究展望

为了扩展图匹配查询的应用领域，构建大规模图数据上能够满足各种查询需求的图匹配查询引擎，需要图匹配查询具有实用能力，即能够提供准确的、高效的和用户友好的查询方法。对于提高图匹配查询实用能力的研究，本文做出了一点探索。然而，这项研究工作尚处于初始阶段。总的来说，其未来的研究方向可能包括以下几个方面。

分布式图匹配查询。针对类似社交网络等领域产生的大规模图数据，使用分布式技术来提高图匹配查询效率往往是一种不可避免选择。图匹配查询的分布式计算一般存在迭代轮次多和通信开销大等问题，因此一个可能并且比较重要的方向是如何设计通用的优化方法使得能够利用迭代轮次之间的关系来优化和减少迭代轮次。

通用图压缩方法。当不需要计算精确的图查询结果时，对大规模复杂数据图进行压缩是提高查询效率的一种有效方法。目前相关研究已经提出针对特定语义的能保持查询结果正确性的无损压缩方法。但是该方法需要针对不同的查询语义分别采用不同的方法对数据图进行预处理，因而该方法存在计算开销大和不实用等问题。因此，如何设计通用的分层压缩模式来提供不同粒度的并且支持所有查询语义的图压缩方法是一个值得研究的问题。这可能需要利用有精确度控制的有损压缩方法，并且需要结合索引技术来提高压缩方法的通用性和有效性。

图数据模型上的约束。类似于传统关系数据库上的函数依赖等约束关系，目前在图数据上也存在几种形式的约束。这些约束表达了模式图不同节点或者边之间的匹配依赖关系。这些依赖关系往往可以用来简化图匹配查询计算过程，从而减少查询的计算开销。研究在图数据上新型依赖关系下图匹配查询的优化是一个很重要的课题，存在深入的理论问题。

图数据库系统整合。已有的图匹配查询算法和本文提出的通用计算框架和相关技术都是从算法和理论层面展开的。如何能将这些概念、理论计算框架和算法整合到现有的图数据库系统中是一项很重要的研究课题。这需要将已有的算法和技术用图数据库中的代数操作表达出来，并且还需要同时保持高效性和有效性。

参 考 文 献

[1] Internet World Stats[EB/OL]. https://www.internetworldstats.com/stats.htm.

[2] Facebook[EB/OL]. https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/.

[3] Facebook[EB/OL]. https://zephoria.com/top-15-valuable-facebook-statistics/.

[4] Facebook[EB/OL]. https://research.fb.com/facebook-s-top-open-data-problems/.

[5] Libkin L. Elements of Finite Model Theory[M]. Texts in Theoretical Computer Science. An EATCS Series, 2004.

[6] Fan W., Li J., Ma S., et al. Adding regular expressions to graph reachability and pattern queries[A]. Proceedings of the 27th International Conference on Data Engineering[C]., 2011:39–50.

[7] Dey S. K., Jamil H. M. A hierarchical approach to reachability query answering in very large graph databases[A]. Proceedings of the 19th ACM Conference on Information and Knowledge Management[C]., 2010:1377–1380.

[8] Maserrat H., Pei J. Neighbor query friendly compression of social networks[A]. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]., 2010:533–542.

[9] Wei F. TEDI: efficient shortest path query answering on graphs[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data[C]., 2010:99–110.

[10] Cormen T. H., Leiserson C. E., Rivest R. L., et al. Introduction to Algorithms, Third Edition[M], 2009.

[11] Ullmann J. R. An Algorithm for Subgraph Isomorphism[J]. J. ACM, 1976, 23(1):31–42.

[12] Cordella L. P., Foggia P., Sansone C., et al. A (sub) graph isomorphism algorithm for matching large graphs[J]. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(10):1367–1372.

[13] Henzinger M. R., Henzinger T. A., Kopke P. W. Computing Simulations on Finite and Infinite Graphs[A]. 36th Annual Symposium on Foundations of Computer Science[C]., 1995:453–462.

[14] Ma S., Cao Y., Fan W., et al. Capturing Topology in Graph Pattern Matching[J]. PVLDB,

2011, 5(4):310–321.

[15] Ma S., Cao Y., Fan W., et al. Strong simulation: Capturing topology in graph pattern matching[J]. ACM Trans. Database Syst., 2014, 39(1):4:1–4:46.

[16] Barceló P., Hurtado C. A., Libkin L., et al. Expressive languages for path queries over graph-structured data[A]. Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems[C]., 2010:3–14.

[17] Tian Y., Patel J. M. TALE: A Tool for Approximate Large Graph Matching[A]. Proceedings of the 24th International Conference on Data Engineering[C]., 2008:963–972.

[18] Bader D. A., Madduri K. A graph-theoretic analysis of the human protein-interaction network using multicore parallel algorithms[J]. Parallel Computing, 2008, 34(11):627–639.

[19] Durand P., Labarre L., Meil A., et al. GenoLink: a graph-based querying and browsing system for investigating the function of genes and proteins[J]. BMC Bioinformatics, 2006, 7:21.

[20] Terveen L. G., McDonald D. W. Social matching: A framework and research agenda[J]. ACM Trans. Comput.-Hum. Interact., 2005, 12(3):401–434.

[21] Fan W., Wang X., Wu Y. ExpFinder: Finding experts by graph pattern matching[A]. 29th IEEE International Conference on Data Engineering[C]., 2013:1316–1319.

[22] Eckerson W. W. Data quality and the bottom line: Achieving business success through a commitment to high quality data[R], 2002.

[23] Fan W., Li J., Ma S., et al. Interaction between record matching and data repairing[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data[C]., 2011:469–480.

[24] Liu C., Chen C., Han J., et al. GPLAG: detection of software plagiarism by program dependence graph analysis[A]. Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]., 2006:872–881.

[25] Ma S., Li J., Hu C., et al. Big graph search: challenges and techniques[J]. Frontiers Comput. Sci., 2016, 10(3):387–398.

[26] 马帅, 李佳, 刘旭东, et al. 大数据时代的图搜索技术 [J]. 信息通信技术, 2013, 7(6):44–51.

[27] Gallagher B. Matching structure and semantics: A survey on graph-based pattern matching[J]. AAAI FS, 2006, 6:45–53.

[28] Fan W., Li J., Ma S., et al. Graph Pattern Matching: From Intractable to Polynomial Time[J]. PVLDB, 2010, 3(1):264–275.

[29] Cook S. A. The Complexity of Theorem-Proving Procedures[A]. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing[C]., 1971:151–158.

[30] Samsung Newsroom[EB/OL]. https://news.samsung.com/global/samsung-electronics-begins-mass-production-of-industrys-largest-capacity-ssd-30-72tb-for-next-generation-enterprise-systems.

[31] Newman M., Barabasi A.-L., Watts D. J. The structure and dynamics of networks[M]. Vol. 19, 2011.

[32] Habibi M., Popescu-Belis A. Query Refinement Using Conversational Context: A Method and an Evaluation Resource[A]. Natural Language Processing and Information Systems - 20th International Conference on Applications of Natural Language to Information Systems[C]., 2015:89–102.

[33] Sajjad H., Pantel P., Gamon M. Underspecified Query Refinement via Natural Language Question Generation[A]. 24th International Conference on Computational Linguistics[C]., 2012:2341–2356.

[34] Milner R. Communication and concurrency[M]. PHI Series in computer science, 1989.

[35] Khan A., Wu Y., Aggarwal C. C., et al. NeMa: Fast Graph Search with Label Similarity[J]. PVLDB, 2013, 6(3):181–192.

[36] Song C., Ge T., Chen C. X., et al. Event Pattern Matching over Graph Streams[J]. PVLDB, 2014, 8(4):413–424.

[37] Lee J., Han W., Kasperovics R., et al. An In-depth Comparison of Subgraph Isomorphism Algorithms in Graph Databases[J]. PVLDB, 2012, 6(2):133–144.

[38] igraph-The network analysis package[EB/OL]. http://igraph.org.

[39] Boost C++ library[EB/OL]. https://www.boost.org.

[40] Abiteboul S., Hull R., Vianu V. Foundations of Databases[M], 1995.

[41] Halevy A. Y. Theory of Answering Queries Using Views[J]. SIGMOD Record, 2000, 29(4):40–47.

[42] Halevy A. Y. Answering queries using views: A survey[J]. VLDB J., 2001, 10(4):270–294.

[43] Lakshmanan L. V. S., Wang W. H., Zhao Z. J. Answering Tree Pattern Queries Using Views[A]. Proceedings of the 32nd International Conference on Very Large Data Bases[C]., 2006:571–582.

[44] Wang J., Li J., Yu J. X. Answering tree pattern queries using views: a revisit[A]. EDBT 2011, Proceedings of the 14th International Conference on Extending Database Technology[C]., 2011:153–164.

[45] Wu X., Theodoratos D., Wang W. H. Answering XML queries using materialized views revisited[A]. Proceedings of the 18th ACM Conference on Information and Knowledge Management[C]., 2009:475–484.

[46] Fan W., Wang X., Wu Y. Answering graph pattern queries using views[A]. IEEE 30th International Conference on Data Engineering[C]., 2014:184–195.

[47] Fan W., Wang X., Wu Y. Answering Pattern Queries Using Views[J]. IEEE Trans. Knowl. Data Eng., 2016, 28(2):326–341.

[48] Ramalingam G., Reps T. W. A Categorized Bibliography on Incremental Computation[A]. Conference Record of the Twentieth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages[C]., 1993:502–510.

[49] Ramalingam G., Reps T. W. On the Computational Complexity of Dynamic Graph Problems[J]. Theor. Comput. Sci., 1996, 158(1&2):233–277.

[50] Fan W., Wang X., Wu Y. Incremental graph pattern matching[J]. ACM Trans. Database Syst., 2013, 38(3):18:1–18:47.

[51] Fan W., Li J., Luo J., et al. Incremental graph pattern matching[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011[C]., 2011:925–936.

[52] Fan W., Hu C., Tian C. Incremental Graph Computations: Doable and Undoable[A]. Proceedings of the 2017 ACM International Conference on Management of Data[C]., 2017:155–169.

[53] Aggarwal C. C., Wang H. Managing and Mining Graph Data[M]. Advances in Database Systems, vol. 40, 2010.

[54] Klein K., Kriege N., Mutzel P. CT-index: Fingerprint-based graph indexing combining cycles and trees[A]. Proceedings of the 27th International Conference on Data Engineering[C]., 2011:1115–1126.

[55] Williams D. W., Huan J., Wang W. Graph Database Indexing Using Structured Graph Decomposition[A]. Proceedings of the 23rd International Conference on Data Engineering[C]., 2007:976–985.

[56] Giugno R., Shasha D. E. GraphGrep: A Fast and Universal Method for Querying Graphs[A]. 16th International Conference on Pattern Recognition[C]., 2002:112–115.

[57] Fan W., Li J., Wang X., et al. Query preserving graph compression[A]. Proceedings of the ACM SIGMOD International Conference on Management of Data[C]., 2012:157–168.

[58] Dovier A., Piazza C., Policriti A. A Fast Bisimulation Algorithm[A]. Computer Aided Verification, 13th International Conference[C]., 2001:79–90.

[59] Leskovec J., Faloutsos C. Sampling from large graphs[A]. Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]., 2006:631–636.

[60] Maiya A. S., Berger-Wolf T. Y. Benefits of bias: towards better characterization of network sampling[A]. Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]., 2011:105–113.

[61] Li R., Yu J. X., Qin L., et al. On random walk based graph sampling[A]. 31st IEEE International Conference on Data Engineering[C]., 2015:927–938.

[62] Bar-Yossef Z., Berg A. C., Chien S., et al. Approximating Aggregate Queries about Web Pages via Random Walks[A]. Proceedings of 26th International Conference on Very Large Data Bases[C]., 2000:535–544.

[63] Mottin D., Bonchi F., Gullo F. Graph Query Reformulation with Diversity[A]. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]., 2015:825–834.

[64] Mishra C., Koudas N. Interactive query refinement[A]. EDBT 2009, 12th International Conference on Extending Database Technology[C]., 2009:862–873.

[65] Yi P., Choi B., Bhowmick S. S., et al. AutoG: a visual query autocompletion framework for graph databases[J]. VLDB J., 2017, 26(3):347–372.

[66] Martinenghi D., Torlone R. Taxonomy-based relaxation of query answering in relational databases[J]. VLDB J., 2014, 23(5):747–769.

[67] Mottin D., Marascu A., Roy S. B., et al. A Probabilistic Optimization Framework for the Empty-Answer Problem[J]. PVLDB, 2013, 6(14):1762–1773.

[68] Elbassuoni S., Ramanath M., Weikum G. Query Relaxation for Entity-Relationship Search[A]. The Semanic Web: Research and Applications - 8th Extended Semantic Web Conference[C]., 2011:62–76.

[69] Huang H., Liu C., Zhou X. Computing Relaxed Answers on RDF Databases[A]. Web Information Systems Engineering - WISE 2008, 9th International Conference[C]., 2008.

[70] Carpineto C., Romano G. A Survey of Automatic Query Expansion in Information Retrieval[J]. ACM Comput. Surv., 2012, 44(1):1:1–1:50.

[71] Schenkel R., Theobald M. Feedback-Driven Structural Query Expansion for Ranked Retrieval of XML Data[A]. Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology[C]., 2006:331–348.

[72] Vasilyeva E., Thiele M., Mocan A., et al. Relaxation of subgraph queries delivering empty results[A]. Proceedings of the 27th International Conference on Scientific and Statistical Database Management[C]., 2015:28:1–28:12.

[73] Ausiello G. Complexity and approximation: combinatorial optimization problems and their approximability properties[M], 1999.

[74] Young N. E. Greedy Set-Cover Algorithms[M]. , Encyclopedia of Algorithms[C]., 2008.

[75] DBpedia[EB/OL]. http://wiki.dbpedia.org/Downloads2015-04.

[76] YouTube[EB/OL]. http://netsg.cs.sfu.ca/youtubedata/.

[77] Jagadish H. V., Koudas N., Muthukrishnan S., et al. Optimal histograms with quality guarantees[A]. VLDB[C]., 1998.

[78] Cormode G., Garofalakis M. N., Haas P. J., et al. Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches[J]. FTDB, 2012.

[79] Barceló P., Libkin L., Romero M. Efficient Approximations of Conjunctive Queries[J]. SICOMP, 2014, 43(3):1085–1130.

[80] Fan W., Geerts F., Cao Y., et al. Querying Big Data by Accessing Small Data[A]. PODS[C]., 2015.

[81] Chaudhuri S., Kolaitis P. G. Can Datalog Be Approximated?[J]. JCSS, 1997, 55(2):355–369.

[82] Shang Z., Yu J. X. Auto-Approximation of Graph Computing[J]. PVLDB, 2014.

[83] Zhang S., Yang J., Jin W. SAPPER: Subgraph Indexing and Approximate Matching in Large Graphs[J]. PVLDB, 2010.

[84] Page L., Brin S., Motwani R., et al. The PageRank citation ranking: Bringing order to the web.[R], 1999.

[85] Yin D., Hu Y., Tang J., et al. Ranking Relevance in Yahoo Search[A]. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]., 2016:323–332.

[86] Jansen B. J., Spink A. Analysis of document viewing patterns of web search engine users[M]. , Web mining: Applications and techniques[C]., 2005:339–354.

[87] The Value of Google Result Positioning[EB/OL]. https://chitika.com/2013/06/07/the-value-of-google-result-positioning-2/.

[88] Peng D., Dabek F. Large-scale Incremental Processing Using Distributed Transactions and Notifications[A]. 9th USENIX Symposium on Operating Systems Design and Implementation[C]., 2010:251–264.

[89] Lappas T., Liu K., Terzi E. Finding a team of experts in social networks[A]. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]., 2009:467–476.

[90] Gajewar A., Sarma A. D. Multi-skill Collaborative Teams based on Densest Subgraphs[A]. Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim[C]., 2012:165–176.

[91] Rangapuram S. S., Bühler T., Hein M. Towards realistic team formation in social networks based on densest subgraphs[A]. 22nd International World Wide Web Conference[C]., 2013:1077–1088.

[92] Kargar M., An A. Discovering top-k teams of experts with/without a leader in social networks[A]. Proceedings of the 20th ACM Conference on Information and Knowledge Management[C]., 2011:985–994.

[93] Fan W., Wu Y., Xu J. Adding Counting Quantifiers to Graph Patterns[A]. Proceedings

of the 2016 International Conference on Management of Data[C]., 2016:1215–1230.

[94] Zou L., Chen L., Özsu M. T. DistanceJoin: Pattern Match Query In a Large Graph Database[J]. PVLDB, 2009, 2(1):886–897.

[95] Goldberg A. V. Finding a Maximum Density Subgraph[A]. Technical Report CSD-84-171[C]., 1984.

[96] Valari E., Kontaki M., Papadopoulos A. N. Discovery of Top-k Dense Subgraphs in Dynamic Graph Collections[A]. Scientific and Statistical Database Management - 24th International Conference[C]., 2012:213–230.

[97] Andreev K., Räcke H. Balanced Graph Partitioning[J]. Theory Comput. Syst., 2006, 39(6):929–939.

[98] Karypis G., Kumar V. Multilevel k-way Partitioning Scheme for Irregular Graphs[J]. J. Parallel Distrib. Comput., 1998, 48(1):96–129.

[99] Fagin R., Lotem A., Naor M. Optimal aggregation algorithms for middleware[J]. J. Comput. Syst. Sci., 2003, 66(4):614–656.

[100] Citation[EB/OL]. https://aminer.org/billboard/citation/.

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

[102] Lawler E. L. A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem[J]. Management Science, 1972, 18(7):401–405.

[103] Twitter[EB/OL]. http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/.

[104] Ilyas I. F., Beskales G., Soliman M. A. A survey of top-*k* query processing techniques in relational database systems[J]. ACM Comput. Surv., 2008, 40(4):11:1–11:58.

[105] Marian A., Amer-Yahia S., Koudas N., et al. Adaptive Processing of Top-K Queries in XML[A]. Proceedings of the 21st International Conference on Data Engineering[C]., 2005:162–173.

[106] Chen L. J., Papakonstantinou Y. Supporting top-K keyword search in XML databases[A]. Proceedings of the 26th International Conference on Data Engineering[C]., 2010:689–700.

[107] Fan W., Wang X., Wu Y. Diversified Top-k Graph Pattern Matching[J]. PVLDB, 2013, 6(13):1510–1521.

[108] Cakmak A., Özsoyoglu G. Taxonomy-superimposed graph mining[A]. 11th International Conference on Extending Database Technology[C]., 2008:217–228.

[109] Aizenbud-Reshef N., Barger A., Guy I., et al. Bon voyage: social travel planning in the enterprise[A]. CSCW '12 Computer Supported Cooperative Work[C]., 2012:819–828.

[110] YAGO[EB/OL]. http://www.mpi-inf.mpg.de/yago.

[111] Wasserman S., Faust K. Social network analysis: Methods and applications[M], 1994.

[112] Czygrinow A. Maximum dispersion problem in dense graphs[J]. Oper. Res. Lett., 2000, 27(5):223–227.

[113] Micali S., Vazirani V. V. An O(sqrt(|v|) |E|) Algorithm for Finding Maximum Matching in General Graphs[A]. 21st Annual Symposium on Foundations of Computer Science[C]., 1980:17–27.

[114] Wu Y., Yang S., Yan X. Ontology-based subgraph querying[A]. 29th IEEE International Conference on Data Engineering[C]., 2013:697–708.

[115] Yang S., Wu Y., Sun H., et al. Schemaless and Structureless Graph Querying[J]. PVLDB, 2014, 7(7):565–576.

[116] Ren X., Wang J. Multi-Query Optimization for Subgraph Isomorphism Search[J]. PVLDB, 2016, 10(3):121–132.

[117] Le W., Kementsietsidis A., Duan S., et al. Scalable Multi-query Optimization for SPARQL[A]. IEEE 28th International Conference on Data Engineering[C]., 2012:666–677.

[118] Freire C., Gatterbauer W., Immerman N., et al. The Complexity of Resilience and Responsibility for Self-Join-Free Conjunctive Queries[J]. PVLDB, 2015, 9(3):180–191.

[119] Roy S., Suciu D. A formal approach to finding explanations for database queries[A]. International Conference on Management of Data[C]., 2014:1579–1590.

[120] Cheney J., Chiticariu L., Tan W. C. Provenance in Databases: Why, How, and Where[J]. Foundations and Trends in Databases, 2009, 1(4):379–474.

[121] Papadimitriou C. H. Computational Complexity[M], 1994.

附录 A　章节三定理证明

A.1　定理 3.1 证明

本文利用以下引理证明定理 3.1。

引理 A.1 (图模拟传递性): 给定模式图 Q_1、Q_2 和 Q_3，如果 $Q_2 \prec Q_2$，$Q_2 \prec Q_3$，则

- $Q_1 \prec Q_3$，并且
- $R(Q_1, Q_2) \circ R(Q_2, Q_3) \subseteq R(Q_1, Q_3)$。

其中 $R(Q,G)$ 是从 Q 到 G 的最大二元匹配关系；$R(Q_1, Q_2) \circ R(Q_2, Q_3) = \{(u_1, u_3) \in V_{Q_1} \times V_{Q_3} \mid \exists u_2 \in V_{Q_2}, (u_1, u_2) \in R(Q_1, Q_2), (u_2, u_3) \in R(Q_2, Q_3)\}$。 □

证明　令 $R_{13} = R(Q_1, Q_2) \circ R(Q_2, Q_3)$。为了证明引理 A.1，首先证明 R_{13} 是一个从 Q_1 到 Q_3 的二元匹配关系。

（1）根据 R_{13} 的定义，对于每个 $(u, v) \in R_{13}$，存在 Q_2 中节点 w 使得 $(u, w) \in R(Q_1, Q_2)$。因此，$l(u) = l(w) = l(v)$。

（2）此外，对于 Q_1 的每个节点 u，一定存在一个 Q_2 的节点 w 和 Q_3 的节点 v 使得 $(u, w) \in R(Q_1, Q_2)$ 且 $(w, v) \in R(Q_2, Q_3)$。

（3）考虑任意 Q_1 的边 (u, u')。因为 $(u, w) \in R(Q_1, Q_2)$，根据二元匹配关系 $R(Q_1, Q_2)$ 的定义，Q_2 中一定存在边 (w, w') 使得 $(u', w') \in R(Q_1, Q_2)$。与此类似，因为 $(w, v) \in R(Q_2, Q_3)$，Q_3 中一定存在边 (v, v') 使得 $(w', v') \in R(Q_2, Q_3)$。因此，根据 R_{13} 的定义，$(u', v') \in R_{13}$。

由（1）、（2）、（3）得出 R_{13} 是一个从 Q_1 到 Q_3 的二元匹配关系，即 $Q_1 \prec Q_3$。因为 $R(Q_1, Q_3)$ 是从 Q_1 到 Q_3 的最大二元匹配关系，因此 $R_{13} \subseteq R(Q_1, Q_3)$ 成立。 □

下面本文利用引理 A.1 证明定理 3.1。为了简化证明，本文首先证明定理 3.1（2），而后证明定理的其他部分。

定理 3.1（2）证明

$\boxed{\Rightarrow}$ 假设 $Q \sqsubseteq^c_{\mathcal{U}} Q_u$。根据 $\sqsubseteq^c_{\mathcal{U}}$ 的定义，对任意数据图 G，则 $Q(G) \subseteq Q_u(G)$ 成立。因此，$Q(Q) \subseteq Q_u(Q)$。观察可知 $Q(Q) = V_Q$ 并且 $Q_u(Q) \subseteq V_Q$。因此 $Q_u(Q) = V_Q \neq \emptyset$。所以，$Q_u \prec Q$ 且 $V_{I_Q[Q_u]} = Q_u(Q) = V_Q$。

$\boxed{\Leftarrow}$ 假设 $Q_u \prec Q$ 并且 $V_Q = V_{I_Q[Q_u]}$。为了证明 $Q \sqsubseteq^c_{\mathcal{U}} Q_u$，需要证明对任意数据图 G，$Q(G) \subseteq Q_u(G)$ 成立。观察可知，如果 $Q \not\prec G$，则 $Q(G) = \emptyset$。因此 $Q(G) \subseteq Q_u(G)$ 显然成立。下面考虑 $Q \prec G$ 情况。对任意二元关系 R，用 $\mathsf{LHS}(R)$ 表示 $\{u \mid \exists v, (u,v) \in R\}$, 用 $\mathsf{RHS}(R)$ 表示 $\{v \mid \exists u, (u,v) \in R\}$。因为 $Q_u \prec Q$ 并且 $Q \prec G$，根据引理 A.1，$Q_u \prec G$ 和 $R(Q_u, Q) \circ R(Q, G) \subseteq R(Q_u, G)$ 成立。又由于 $Q_u(Q) = V_{I_Q[Q_u]} = V_Q$，所以有 $\mathsf{RHS}(R(Q_u, Q)) = V_Q$。因此, $R(Q, G)$ 中的所有匹配关系在 $R(Q_u, Q) \circ R(Q, G)$ 的计算过程中全部被保留。因此，$\mathsf{RHS}(R(Q_u, Q) \circ R(Q, G)) = \mathsf{RHS}(R(Q, G))$。又由于 $R(Q_u, Q) \circ R(Q, G) \subseteq R(Q_u, G)$，所以有 $\mathsf{RHS}(R(Q, G)) = \mathsf{RHS}(R(Q_u, Q) \circ R(Q, G)) \subseteq \mathsf{RHS}(R(Q_u, G))$。因为 $Q \prec G$，$Q(G) = \mathsf{RHS}(R(Q, G))$ and $Q_u(G) = \mathsf{RHS}(R(Q_u, G))$。因此，$Q_u \prec G$ 并且 $R(Q_u, G)$ 是 Q_u 到 G 的最大二元匹配关系。因此，$Q(G) \subseteq Q_u(G)$，也即 $Q \sqsubseteq^c_{\mathcal{U}} Q_u$。

定理 3.1（1）证明

$\boxed{\Rightarrow}$ 假设 $Q \sqsubseteq_{\mathcal{U}} Q_u$。根据 $\sqsubseteq_{\mathcal{U}}$ 的定义，则一定存在 Q 的诱导子图 Q_s 使得对于任意数据图 G，$Q_s(G) \subseteq Q_u(G)$。因此，$Q_s(Q_s) \subseteq Q_u(Q_s)$。又因为 $Q_s(Q_s) = V_{Q_s}$ 且 $Q_u(Q_s) \subseteq V_{Q_s}$。因此，$Q_u(Q_s) = V_{Q_s} \neq \emptyset$。所以，$Q_u \prec Q_s$。因为 Q_s 是 Q 的子图，所以 $Q_u \prec Q$。

$\boxed{\Leftarrow}$ 假设 $Q_u \prec Q$。为了证明 $Q \sqsubseteq_{\mathcal{U}} Q_u$，只需证明存在 Q 的诱导子图 Q_s 满足 $Q_s \sqsubseteq^c_{\mathcal{U}} Q_u$。令 Q_0 表示 Q_u 在 Q 中的镜像 $I_Q[Q_u]$。则 Q_0 是 Q 的诱导子图。又因为 $Q_u \prec Q_0$ 且 $V_{Q_0} = V_{I_{Q_0}[Q_u]}$。根据定理 3.1（2），则 $Q_u \sqsubseteq^c_{\mathcal{U}} Q_0$ 成立。因此，$Q_u \sqsubseteq_{\mathcal{U}} Q$。

定理 3.1（3）证明

$\boxed{\Rightarrow}$ 假设 $Q_l \sqsubseteq_{\mathcal{L}} Q$。则一定存在 Q 的诱导子图 Q_s 满足 $Q_l \sqsubseteq^c_{\mathcal{L}} Q_s$。根据定理 3.1（2），$Q_s \prec Q_l$ 并且 $V_Q = V_{I_{Q_l}[Q_s]}$ 成立。

$\boxed{\Leftarrow}$ 假设存在 Q 的诱导子图 Q_s 满足 $Q_s \prec Q_l$ 和 $V_{Q_l} = V_{I_{Q_l}[Q_s]}$。根据定理 3.1（2），则 $Q_l \sqsubseteq^c_{\mathcal{U}} Q_s$ 成立。因此 $Q_l \sqsubseteq^c_{\mathcal{L}} Q_s$。又因为 Q_s 是 Q 的诱导子图，所以 $Q_l \sqsubseteq_{\mathcal{L}} Q$ 成立。

定理 3.1（4）证明

$\boxed{\Rightarrow}$ 假设 $Q_l \sqsubseteq^c_{\mathcal{L}} Q$。根据完全上界近似和完全下界近似的概念可知，$Q_l \sqsubseteq^c_{\mathcal{L}} Q$ 当且仅当 $Q_l \sqsubseteq^c_{\mathcal{U}} Q$ 根据定理 3.1（2）可知，$Q \prec Q_l$ 并且 $V_{Q_l} = V_{I_{Q_l}[Q]}$ 成立。

$\boxed{\Leftarrow}$ 假设 $Q \prec Q_l$ 并且 $V_{Q_l} = V_{I_{Q_l}[Q]}$。根据定理 3.1（2），$Q_l \sqsubseteq^c_{\mathcal{U}} Q$ 成立。因此 $Q_l \sqsubseteq^c_{\mathcal{L}} Q$。

A.2 定理 3.2 证明

定理 3.2（a）证明

$\boxed{\Rightarrow}$ 假设 Q 存在一个基于 $\mathcal{V}$ 的上界近似查询 Q_u。根据定理 3.1（1），$Q_u \prec Q$ 成立。因为 Q_u 能够由 $\mathcal{V}$ 中视图回答，根据引理 3.1可知，$E_{Q_u} = \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{Q_u}[\mathsf{V}]}$ 成立。因此，$\mathcal{V}$ 中一定存在一个视图 $\mathsf{V} \in \mathcal{V}$ 满足 $E_{I_{Q_u}[\mathsf{V}]} \neq \emptyset$，即 $\mathsf{V} \prec Q_u$。因为 $Q_u \prec Q$，根据引理 A.1可知，$\mathsf{V} \prec Q$ 成立，也即 $\mathsf{V}(Q) \neq \emptyset$。

$\boxed{\Leftarrow}$ 假设 $\mathcal{V}$ 中存在视图 V 使得 $\mathsf{V}(Q) \neq \emptyset$，即 $\mathsf{V} \prec Q$。根据定理 3.1（1），$\mathsf{V} \sqsubseteq_{\mathcal{U}} Q$。因此，$\mathsf{V}$ 是 Q 的基于 $\mathcal{V}$ 的上界近似查询。

定理 3.2（b）证明

$\boxed{\Rightarrow}$ 假设 Q 存在一个基于 $\mathcal{V}$ 的完全上界近似查询 Q_u。根据定理 3.1（1）可知，$Q_u \prec Q$ 并且 $V_Q = V_{I_Q[Q_u]} = Q_u(Q)$ 成立。用 $\mathcal{S}$ 表示集合 $\{\mathsf{V} \in \mathcal{V} \mid \mathsf{V} \prec Q_u\}$。因为 Q_u 能够用视图 $\mathcal{V}$ 回答，根据引理 3.1，$E_{Q_u} = \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{Q_u}[\mathsf{V}]}$ 成立。因此，$\mathsf{RHS}(\bigcup_{\mathsf{V}\in\mathcal{S}} R(\mathsf{V}, Q_u)) = V_{Q_u}$（用 $R(Q,G)$ 表示从 Q 到 G 的最大二元匹配关系）。由于对于任意 $\mathsf{V} \in \mathcal{S}$，$Q_u \prec Q$ 并且 $\mathsf{V} \prec Q_u$ 成立。利用引理 A.1，可以得到 $\bigcup_{\mathsf{V}\in\mathcal{S}} R(\mathsf{V}, Q_u) \circ R(Q_u, Q) = \bigcup_{\mathsf{V}\in\mathcal{S}} (R(\mathsf{V}, Q_u) \circ R(Q_u, Q)) \subseteq \bigcup_{\mathsf{V}\in\mathcal{S}} R(\mathsf{V}, Q)$。又因为 $\mathsf{RHS}(\bigcup_{\mathsf{V}\in\mathcal{S}} R(\mathsf{V}, Q_u)) = V_{Q_u}$，$\mathsf{RHS}(\bigcup_{\mathsf{V}\in\mathcal{S}} R(\mathsf{V}, Q_u) \circ R(Q_u, Q)) = \mathsf{RHS}(R(Q_u, Q)) = V_{I_Q[Q_u]} = V_Q$。因此 $\mathsf{RHS}(\bigcup_{\mathsf{V}\in\mathcal{S}} R(\mathsf{V}, Q)) \supseteq V_Q$ 成立。因为对任意 $\mathsf{V} \in \mathcal{S}$，$\mathsf{RHS}(R(\mathsf{V}, Q)) \subseteq V_Q$。所以 $\mathsf{RHS}(\bigcup_{\mathsf{V}\in\mathcal{S}} R(\mathsf{V}, Q)) = V_Q$ 成立。因此，$\bigcup_{\mathsf{V}\in\mathcal{S}} V_{I_Q[\mathsf{V}]} = \bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]} = V_Q$。

$\boxed{\Leftarrow}$ 假设 $V_Q = \bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]}$。定义 Q_u 为 Q 中包含所有 $\bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_Q[\mathsf{V}]}$ 中边的子图。则 $V_{Q_u} = \bigcup_{\mathsf{V}\in\mathcal{V}} V_{I_Q[\mathsf{V}]} = V_Q$。因此，$Q_u \prec Q$ 并且 $V_Q = V_{I_Q[Q_u]}$ 成立。根据引理 3.1（2）可知，$Q \sqsubseteq^c_{\mathcal{U}} Q_u$。

A.3 定理 3.3 证明

定理 3.3（a）证明

$\boxed{\Rightarrow}$ 假设 Q 存在基于 $\mathcal{V}$ 的完全下界近似查询 Q_l，即 $Q_l \sqsubseteq^c_{\mathcal{L}} Q$ 并且 Q_l 能够被视图 $\mathcal{V}$ 回答。根据引理 3.3，一定存在 $\hat{Q}$ 的子图 Q' 满足 $V_{Q'} = V_{\hat{Q}} = V_Q$。如果 Q' 是 Q 的子图，则 $E_Q \subseteq E_{Q'} = \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{Q'}[\mathsf{V}]}$。根据引理 3.1 可知，对于任意 $\mathsf{V} \in \mathcal{V}$，$E_{I_{Q'}[\mathsf{V}]} \subseteq E_{I_{\hat{Q}}[\mathsf{V}]}$ 成立。因此，$E_Q \subseteq \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{\hat{Q}}[\mathsf{V}]}$。如果 Q' 不是 Q 的子图，定义 Q'' 为由所有 Q 和 Q' 中边组成的图。由于 $Q' \sqsubseteq^c_{\mathcal{L}} Q$，所以 $Q'' \sqsubseteq^c_{\mathcal{L}} Q$ 成立。此外，由于 Q' 能够被 $\mathcal{V}$ 回答并且 $Q' \sqsubseteq^c_{\mathcal{L}} Q''$，所以 Q'' 也能够被视图 $\mathcal{V}$ 回答。因此，$E_Q \subseteq E_{Q''} = \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{Q''}[\mathsf{V}]} \subseteq \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{\hat{Q}}[\mathsf{V}]}$。

$\boxed{\Leftarrow}$ 假设 $E_Q \subseteq \bigcup_{\mathsf{V}\in\mathcal{V}} E_{I_{\hat{Q}}[\mathsf{V}]}$。下面构建一个 Q 的基于 $\mathcal{V}$ 的完全下界近似查询 Q^c_l。定义

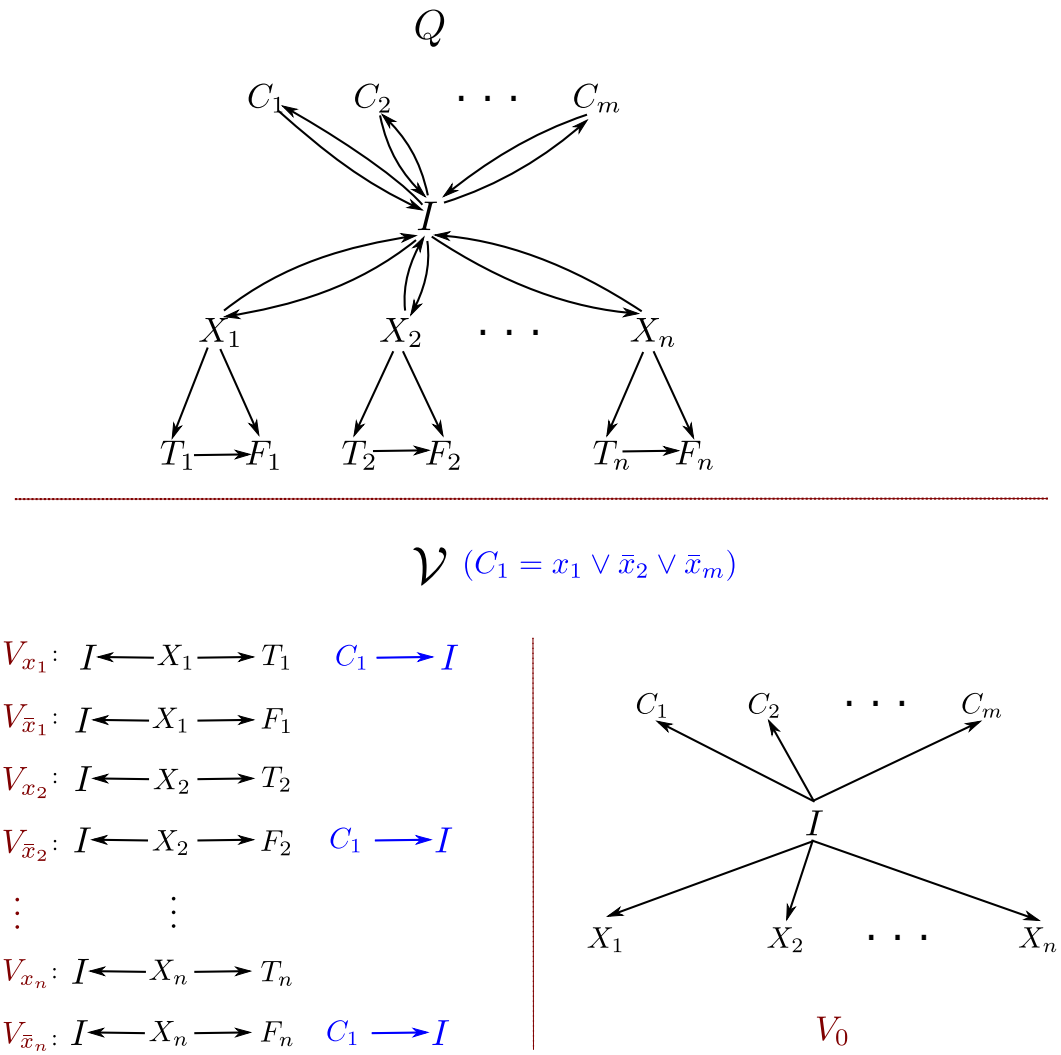


图 A.1　定理 3.3(c) 证明所用到的 Q 和 $\mathcal{V}$ 的构造

Q_l^c 为由 $\bigcup_{V \in V} E_{I_{\hat{Q}}[V]}$ 中所有边组成的图。根据引理 3.1 可知，Q_l^c 能够被视图 $\mathcal{V}$ 回答。又由于 $E_Q \subseteq E_{Q_l^c}$，所以 $Q \prec Q_l^c$ 并且 $V_{Q_l^c} = V_Q$ 成立。根据定理 3.1（4）可知，$Q_l^c \sqsubseteq_{\mathcal{L}}^c Q$ 成立。因此，Q_l^c 是 Q 的基于视图 $\mathcal{V}$ 的完全下界近似查询。

定理 3.3（c）证明

证明 ELA 是 NP 类问题。本文通过为 ELA 设计一个 NP 算法来证明 ELA 问题是 NP 类问题。给定任意模式图 Q，视图集合 $\mathcal{V}$，该算法的主要步骤如下所示：

（a） 猜想 Q 的一个子图 Q_s；

（b） 验证 Q_s 是否是基于 $\mathcal{V}$ 的下界近似查询。如果是，则返回“是”；否则继续步骤（b）。

上述算法是 ELA 问题的 NP 算法。因为根据定理 3.3（a），第（2）步能在 PTIME 时

间内执行完毕。

证明 ELA 是NP-难问题。本文通过将 3SAT 问题归约到 ELA 问题来证明 ELA 是 NP-难问题。3SAT 问题的一个实例是，给定一个变量集合 $S_\varphi = \{x_1, \ldots, x_n\}$，一个定义在 S_φ 上的命题公式 $\varphi = C_1 \wedge \cdots \wedge C_m$（其中对任意 $i \in [1,m]$，φ 的从句 C_i 具有语法形式 $\ell_1^i \vee \ell_2^i \vee \ell_3^i$：对于任意 $j \in [1,3]$，ℓ_j^i 或者是 S_φ 中的变量 x_l，或者是 x_l 的否定）。给定任意 φ，3SAT 问题即判定 φ 是否为可满足的，即是否存在一个对 S_φ 中变量的真值赋值 μ 能够使 φ 为真。3SAT 问题是 NP-完全问题 [121].

给定 3SAT 的一个实例 φ，归约过程构建 ELA 的一个实例，即模式图 Q，视图集合 $\mathcal{V}$，使得 φ 是可满足的当且仅当 Q 具有一个基于 $\mathcal{V}$ 的下界近似查询。构建方法如下。

（a）模式图 Q 的构造如图 A.1 所示。Q 的节点集包括：（i）节点 $C_1, \ldots, C_m$，编码了 φ 中的 m 个子句；（ii）节点 $X_1, \ldots, X_n$，编码了 S_φ 中的 n 个变量；（iii）节点 $T_1, F_1, \ldots, T_n, F_n$，编码了变量 x_1、…、x_n 的所有可能的真值赋值；（iv）一个额外节点 I。Q 的边集包括：对于每个 $i \in [1,m]$ 和 $j \in [1,n]$，(C_i, I)、(I, C_i)、(X_j, I)、(I, X_j)、(X_j, T_j)、(X_j, F_j) 和 (T_j, F_j)。

（b）$\mathcal{V}$ 包含以下 $2n+1$ 个视图：

- 对于每个 S_φ 中变量 x_j $(j \in [1,n])$, $\mathcal{V}$ 包含两个视图 V_{x_j} 和 $V_{\bar{x}_j}$，用于编码 x_j 的真值赋值以及 φ 中涉及到 x_j 的子句。具体而言，V_{x_j} 包含边 (X_j, I) 和 (X_j, T_j)；$V_{\bar{x}_j}$ 包含边 (X_j, I) 和 (X_j, F_j)。此外，对于任意 φ 的子句 C_i，如果 C_i 涉及到 x_j（即 C_i 包含 x_j 或者 $\bar{x}_j$），则 V_{x_j} 也包含边 (C_i, I)。所以 V_{x_j} 能够编码以下语义：如果 x_j 为 *true*，则 C_i 也为 *true*。与此类似，如果 C_i 包含 $\bar{x}_j$，则 $V_{\bar{x}_j}$ 也额外包含边 (C_i, I)。图 A.1 给出了一个针对 $C_1 = x_1 \vee \bar{x}_2 \vee \bar{x}_m$ 的 φ 的归约对应的 V_{x_1}、$V_{\bar{x}_1}$、…、V_{x_n}、$V_{\bar{x}_n}$ 的示例。
- $\mathcal{V}$ 还额外包含一个特殊视图 V_0（如图 A.1 所示）。该视图由从 I 到 C_1、…、C_m、X_1、…、X_n 的边构成。

下面证明 φ 是可满足的当且仅当存在一个 Q 的基于 $\mathcal{V}$ 的下界近似查询。

$\boxed{\Rightarrow}$ 假设 φ 是可满足的，即存在变量 S_φ 的真值赋值 μ 使得 $\mu(\varphi) = \mathit{true}$。考虑 $\mathcal{V}$ 的包含以下视图的子集 $\mathcal{V}'$：

- 如果 $\mu(x_j) = \mathit{true}$ $(j \in [1,n])$，则 V_{x_j} 被包含在 $\mathcal{V}'$ 中；否则 $V_{\bar{x}_j}$ 被包含在 $\mathcal{V}'$ 中；
- 此外，$\mathcal{V}'$ 还包含 V_0。

定义 Q 的子图 Q_l 为由所有 $\mathcal{V}'$ 中视图在 Q 上的镜像 $I_Q[\mathrm{V}]$ 的边组成的模式图。下面证明 Q_l 是 Q 的基于 $\mathcal{V}$ 的下界近似查询。由于 $\mu(\varphi) = true$，所以对于所有 C_i ($i \in [1,m]$)，一定存在某个变量 x_j ($j \in [1,n]$) 满足（i）C_i 包含 x_j 或者 $\bar{x}_j$；（ii）如果 C_i 包含 x_j，则 $\mu(x_j) = true$；如果 C_i 包含 $\bar{x}_j$，则 $\mu(x_j) = false$。因此，根据视图 V_{x_j} 和 $V_{\bar{x}_j}$ 的定义，所有从 C_i ($i \in [1,m]$) 到 I 的边也都是 Q_l 中的边。

此外，由于 $\mathcal{V}'$ 或者包含 V_{x_j}，或者包含 $V_{\bar{x}_j}$，因此 Q_l 一定也包含从 X_j ($j \in [1,n]$) 到 I 的所有边。另外，由于 $V_0 \in \mathcal{V}'$，Q_l 包含了所有从 I 到 C_i ($i \in [1,m]$) 的边以及从 I 到 X_j ($j \in [1,n]$) 的边。由于对每个节点 X_j，或者 (X_j, T_j) 是 Q_s 的边，或者 (X_j, F_j) 是 Q_s 的边，但是两者不能同时为 Q_s 的边。因此，根据定理 3.3（a）可知，Q_l 是 Q 的基于 $\mathcal{V}$ 的下界近似查询。

$\boxed{\Leftarrow}$ 假设 Q 存在一个基于 $\mathcal{V}$ 的下界近似查询 Q_l。用 $\mathcal{V}'$ 表示回答 Q_l 时需要使用到的 $\mathcal{V}$ 中视图组成的集合。首先，假设 $V_0 \notin \mathcal{V}'$，则 Q_l 不包含任何从 I 到 C_i ($i \in [1,m]$) 或者到 X_j ($j \in [1,n]$) 的边。如果 $\mathcal{V}'$ 包含某个视图 V_{x_j} 或者 $V_{\bar{x}_j}$，则 Q_l 一定包含至少一条从 I 到 X_j 的边。因此，根据定理 3.3（a），以及 Q_l 不包含 (X_j, I) 可知，Q_l 不可能是 Q 的基于 $\mathcal{V}$ 的下界近似查询。因此，$\mathcal{V}'$ 一定包含 V_0。

下面证明 $\mathcal{V}'$ 一定包含 V_{x_j} 或者 $V_{\bar{x}_j}$，但是不能同时包含 V_{x_j} 和 $V_{\bar{x}_j}$。首先，假设 $\mathcal{V}'$ 既不包含 V_{x_j} 又不包含 $V_{\bar{x}_j}$。则 Q_l 一定不包含边 (X_j, I)。因为 $V_0 \in \mathcal{V}'$，所以 (I, X_j) 是 Q_l 的一条边。因此，根据定理 3.3（a），Q_l 不可能是 Q 的基于 $\mathcal{V}$ 的下界近似查询。进一步假设 $\mathcal{V}'$ 同时包含 V_{x_j} 和 $V_{\bar{x}_j}$。则 Q_l 同时包含边 (X_j, T_j) 和 (X_j, F_j)，但是不包含边 (T_j, F_j)。因此，Q_l 不可能为 Q 的基于 $\mathcal{V}$ 的下界近似查询。所以 $\mathcal{V}'$ 包含 V_0，并且 $\mathcal{V}'$ 还包含 V_{x_j} 或 $V_{\bar{x}_j}$ ($j \in [1,n]$)，但是不同时包含 V_{x_j} 和 $V_{\bar{x}_j}$。上述过程确保 $\mathcal{V}'$ 编码了 S_φ 中变量的一个合法的真值赋值，记为 $\mu_{\mathcal{V}'}$，其定义如下：

- 如果 $\mathcal{V}'$ 包含 $V_{\bar{x}_j}$ ($j \in [1,n]$)，则 $\mu_{\mathcal{V}'}(x_j) = true$；否则
- $\mu_{\mathcal{V}'}(x_j) = false$。

下面证明 $\mu_{\mathcal{V}'}(\varphi) = true$。由于 Q_l 是 Q 的下界近似查询，并且由于 Q_l 一定包含所有从 I 到 C_i 的边 (I, C_i) ($i \in [1,m]$)（因为 $V_0 \in \mathcal{V}'$），所以 Q_l 一定包含边 (C_i, I)。根据 $\mathcal{V}$ 的构造可知，C_i 或者包含 x_j 并且 $V_{x_j} \in \mathcal{V}'$，或者包含 $\bar{x}_j$ 并且 $V_{\bar{x}_j} \in \mathcal{V}'$。无论哪种情况，$\mu_{\mathcal{V}'}(C_i) = true$ 成立。因此，$\mu_{\mathcal{V}'}(\varphi) = true$，即 φ 是可满足的。

A.4 定理 3.5 证明

定理 3.5（1）证明

本文首先证明 DCLA^c 是 NP-完全问题。而后，本文证明 DCLA 是 NP-完全问题。

DCLA^c 是 NP-完全问题。本文首先证明 DCLA^c 问题是 NP 类问题，而后再证明该问题是 NP-难问题。

证明 DCLA^c 是 NP 类问题。本文通过设计 NP 算法来证明 DCLA^c 问题是 NP 类问题。给定模式图 Q，视图集合 $\mathcal{V}$，一个正整数 k，该算法按如下步骤判定是否存在一个完全下界近似查询 Q_l 满足 $\mathrm{clo}(Q_l, Q) \leq k$。

（a） 猜测一个 $\mathcal{V}$ 的子集 $\mathcal{V}'$；

（b） 验证 $\mathcal{V}'$ 中所有视图 $\mathrm{V} \in \mathcal{V}'$ 的镜像合并 $\bigcup_{\mathrm{V}\in\mathcal{V}'} I_Q[\mathrm{V}]$ 是否为 Q 的一个完全下界近似查询，并且满足 $\mathrm{clo}(Q_l, Q) \leq k$。如果满足，则返回“是”；否则继续步骤（a）。

上述算法是 DCLA^c 的 NP 算法。因为该算法返回“是”当且仅当 Q 存在一个基于 $\mathcal{V}$ 的完全下界近似查询。并且步骤（b）可以在 PTIME 时间内执行完毕。

证明 DCLA^c 是 NP-难问题。本文通过将 *Set cover* 问题归约到 DCLA^c 问题来证明 DCLA^c 是 NP-难问题。*Set cover* 问题的一个实例是，给定一个全域集合 U 包括 n 个元素 $x_1, \ldots, x_n$，一个集合 $\mathcal{S}$ 包括 m 个 U 的子集 $S_1, \ldots, S_m$，一个正整数 K，判定 $\mathcal{S}$ 中是否存在一个子集 $\mathcal{S}' \subseteq \mathcal{S}$ 满足 $\bigcup_{S\in\mathcal{S}'} S = U$ 和 $|\mathcal{S}'| \leq K$ 成立，则称 $\mathcal{S}'$ 是 U 的一个覆盖。*Set cover* 问题是 NP-难问题 [121]。

给定一个 *Set cover* 问题的实例，即 $U, \mathcal{S}$ 和 K，归约过程构建 DCLA 问题的一个实例，即模式图 Q，视图集合 $\mathcal{V}$，正整数 k，满足 U 存在一个包含子集个数不大于 K 的覆盖 $\mathcal{S}'$ 当且仅当 Q 存在一个基于 $\mathcal{V}$ 的完全下界近似查询 Q_l，使得 $\mathrm{clo}(Q, Q_l) \leq k'$ 成立。构建方法如下。

（1）模式图 Q 包括 n 条边：$(u_1, u_1'), \ldots, (u_n, u_n')$，编码了 U 中所有元素。除此之外，Q 还包括 $m+1$ 个节点：$v_1, \ldots, v_m$ 编码了 $\mathcal{S}$ 的 m 个子集；以及 v_0。

（2）视图集合 $\mathcal{V}$ 包括 m 个视图 $V_1, \ldots, V_m$。其中 V_i 编码了 $\mathcal{S}$ 中的 S_i。具体而言，V_i 包括 $|S_i|+1$ 条边：（i）Q 的 $|S_i|$ 条边编码了 S_i 中的 $|S_i|$ 个元素；以及（ii）一条不属于 Q 的额外边 (v_0, v_i)。

（3）令 $k = K$

下面证明 U 存在一个包含子集个数不大于 K 的覆盖当且仅当 Q 存在一个基于 $\mathcal{V}$ 的完全下界近似查询 Q_l，使得 $\text{clo}(Q, Q_l) \leq K$ 成立。

$\boxed{\Rightarrow}$ 假设存在一个 $\mathcal{S}$ 的子集 $\mathcal{S}' \subseteq \mathcal{S}$ 满足 $\mathcal{S}'$ 是 U 的一个覆盖，并且 $|\mathcal{S}'| \leq K$。归约过程构建一个 $\mathcal{V}$ 的子集 $\mathcal{V}' \subseteq \mathcal{V}$ 使得包括 $\mathcal{V}'$ 中所有边的 Q 的子图 Q' 是 Q 的基于 $\mathcal{V}$ 的一个完全下界近似查询，并且 $\text{clo}(Q, Q') \leq K$ 成立。具体而言，$\mathcal{V}'$ 包括 $|\mathcal{S}'|$ 个编码了 $\mathcal{S}'$ 的视图。其中，由于 U 中所有元素都被 $\mathcal{S}'$ 覆盖，所以 Q' 肯定包括了 Q 中所有边。此外，由于 Q' 的节点同样也是 Q 的节点，所以 Q' 是 Q 的一个完全下界近似查询。观察可知 Q' 中存在 $|\mathcal{S}'|$ 条不属于 Q 的边 (v_0, v_i)。所以，$\text{clo}(Q, Q') = |\mathcal{S}'| \leq K$ 成立。

$\boxed{\Leftarrow}$ 假设存在一个 Q 的基于 $\mathcal{V}$ 的完全下界近似查询 Q' 满足 $\text{clo}(Q, Q') \leq K$。所以一定存在 $\mathcal{V}$ 的子集 $\mathcal{V}' \subset \mathcal{V}$ 使得 Q' 包括 $\mathcal{V}'$ 中所有边。令 $\mathcal{S}'$ 表示 $\mathcal{S}$ 的一个子集，满足 $\mathcal{S}$ 中的 S_i 属于 $\mathcal{S}'$ 当且仅当 $\mathcal{V}'$ 中存在视图 V_i 编码了 S_i。由于 Q' 是 Q 的基于 $\mathcal{V}$ 的完全下界近似查询，所以 Q' 肯定包括了 Q 中所有边。所以 U 中的所有元素都被 $\mathcal{S}'$ 中的子集覆盖。此外，由于 $\text{clo}(Q, Q') \leq K$ 并且 Q' 中不属于 Q 的每条边都属于 $\mathcal{V}'$ 中一个不同的视图，所以 $|\mathcal{S}'| \leq K$ 成立。

DCLA 是 NP-完全问题。本文首先证明 DCLA 问题是 NP 类问题，而后再证明该问题是 NP-难问题。

证明 DCLA 是 NP 类问题。本文通过设计 NP 算法来证明 DCLA 问题是 NP 类问题。给定模式图 Q，视图集合 $\mathcal{V}$，一个正整数 k，该算法按如下步骤判定是否存在一个下界近似查询 Q_l 满足 $\text{clo}(Q_l, Q) \leq k$。

（a） 猜测一个 $\mathcal{V}$ 的子集 $\mathcal{V}'$；

（b） 验证 $\mathcal{V}'$ 中所有视图 $V \in \mathcal{V}'$ 的镜像合并 $\bigcup_{V \in \mathcal{V}'} I_Q[V]$ 是否为 Q 的一个下界近似查询，并且满足 $\text{clo}(Q_l, Q) \leq k$。如果满足，则返回“是”；否则继续步骤（a）。

上述算法是 DCLA 的 NP 算法。因为该算法返回“是”当且仅当 Q 存在一个基于 $\mathcal{V}$ 的下界近似查询，并且步骤（b）可以在 PTIME 时间内执行完毕。

证明 DCLA 是 NP-难问题。由定理 3.3（c）可知 ELA 问题已经是 NP-难问题，所以 ELA 的优化问题必然为 NP-难问题。

定理 3.5（2）证明

对于 OCLA^c 问题，可以验证上述归约过程同样是一个由 *Minimum set cover* 问题到 OCLA^c 问题的 AP-归约过程 [73]。由于 *Minimum set cover* 问题是 APX-难问题，所以 OCLA^c 问题也是 APX-难问题。对于 OCLA 问题，由于其存在性问题 ELA 已经是 NP-难问题，所以 OCLA 不属于 APX 类。

A.5 引理 3.2 证明

本文证明引理 3.2 对于上界近似查询成立。对于完全上界近似查询的证明可以采用相同的方法。

对于任意 Q 的基于 $\mathcal{V}$ 的上界近似查询 Q_u，下面证明 $I_Q[Q_u]$ 也是一个 Q 的基于 $\mathcal{V}$ 的上界近似查询，使得 $\mathrm{clo}(I_Q[Q_u], Q) \leq \mathrm{clo}(Q_u, Q)$ 并且 $I_Q[Q_u]$ 与 Q_u 等价。具体而言，首先因为 $Q_u \prec I_Q[Q_u]$ 以及 $I_Q[Q_u] \prec Q_u$ 成立，所以 $I_Q[Q_u]$ 与 Q_u 等价。用 Q^s 表示 Q_u 和 Q 的最大公共子图。所以 $Q^s \prec Q$ 成立。因为 $Q_u \prec I_Q[Q_u]$ 并且 Q^s 是 Q_u 的一个子图，所以 $R(Q^s, Q) \cup R(Q_u, Q) \subseteq R(Q_u, Q) = R(Q_u, I_Q[Q_u])$（回顾引理 A.1 中定义 $R(Q, G)$ 为 Q 在 G 中基于图模拟的最大二元匹配关系）。所以 Q^s 一定是 $I_Q[Q_u]$ 的一个子图。因此，$\mathrm{clo}(I_Q[Q_u], Q) \leq \mathrm{clo}(Q_u, Q)$ 成立。也就是说，$I_Q[Q_u]$ 与 Q_u 等价并且 $\mathrm{clo}(I_Q[Q_u], Q) \leq \mathrm{clo}(Q_u, Q)$ 成立。

A.6 引理 3.3 证明

本文证明引理 3.3 对于完全下界近似查询成立。对于下界近似查询的证明可以采用相同的方法。

对于任意 Q 的基于 $\mathcal{V}$ 的完全下界近似查询 Q_l，令 Q_0 为 $\hat{Q}$ 中的最小子图使得 $V_{Q_0} = V_{\hat{Q}} = V_Q, E_{Q_0} \supseteq E_Q$ 并且 Q_0 是 Q_l 在 $\hat{Q}$ 中的镜像。注意对于任意完全下界近似查询 Q_l，这样的 Q_0 一定存在。下面证明 Q_0 也是 Q 的基于 $\mathcal{V}$ 的完全下界近似查询，使得 $\mathrm{clo}(Q_l, Q) \geq \mathrm{clo}(Q_0, Q)$，并且 Q_0 与 Q_l 等价。由于 Q_0 是 Q_l 在 $\hat{Q}$ 中的镜像，所以 Q_0 与 Q_l 基于图模拟语义等价。此外，显然 Q_0 是一个完全下界近似查询。

（1）如果 Q_l 是 Q 的一个子图，根据定理 3.3，则 Q 自身即是 Q 的基于 $\mathcal{V}$ 的完全下界近似查询，并且满足 $\mathrm{clo}(Q, Q) < \mathrm{clo}(Q_l, Q)$，$I_Q[Q_l] = Q$。所以在这种情况下，$Q_0$ 即为 Q，并且 $\mathrm{clo}(Q_0, Q) \leq \mathrm{clo}(Q_l, Q)$。（2）如果 Q_l 是 Q 的一个超图，根据 Q_0 的定义，由于 $E_{Q_l} \supseteq E_Q$ 并且 $V_{Q_l} = V_Q$，则 $\mathrm{clo}(Q_0, Q) \leq \mathrm{clo}(Q_l, Q)$ 成立。（3）如果 Q_l 既不是 Q 的

子图也不是 Q 的超图。用 Q' 表示属于 Q 但不属于 Q_l 的子图部分，用 Q'_l 表示 Q' 和 Q_l 的合并。由于 Q' 是 $I_Q[Q_l]$ 的子图，所以 Q'_l 与 Q_l 等价。所以，$\mathsf{clo}(Q'_l, Q) \leq \mathsf{clo}(Q_l, Q)$ 成立。根据 Q_0 的定义，由于 $E_{Q'_l} \supseteq E_Q$ 和 $V_{Q'_l} = V_Q$，所以，Q_0 与 Q'_l（Q_l）等价，并且 $\mathsf{clo}(Q_0, Q) \leq \mathsf{clo}(Q'_l, Q)$ 成立。

附录 B　章节四定理证明

B.1　定理 4.1 证明

本文通过设计时间复杂度为 $O(|Q|^2)$ 的算法，并给出其正确性证明来证明该定理。模式图 Q 的可满足性可以由以下算法判定：

（a） 计算 $Q \prec Q$ 的最大二元匹配关系 R_M；

（b） 对于每个 $(u,v) \in R$，其中节点 u 上的容量区间为 $[x_u, y_u]$，节点 v 上的容量区间为 $[x_v, y_v]$，判断 $x_v \le y_u$ 是否成立。

可以验证，通过调用图模拟（无向图）查询过程 ugSim（表 10），步骤（a）可以在 $O(|Q|^2)$ 时间内执行完毕；步骤（b）可以在 $O(|Q|^2)$ 时间内执行完毕，因为 R_M 的规模最大为 $|Q|^2$。所以该算法的时间复杂度为 $O(|Q|^2)$。

下面证明该算法的正确性。

（I）首先证明当算法返回“是”时，模式图 Q 是可满足的。这是因为当算法返回“是”时，说明 $Q \prec Q$，其最大二元匹配关系为 R_M，并且 R_M 中的节点满足所有容量区间，则一定存在一个数据图 G 满足 $Q \lhd_r G$。数据图 G 可以按照如下方法构建：（i）根据 R_M 计算 Q 中节点的等价类，使得节点 u 和 w 属于同一等价类当且仅当 $(u,w) \in R$ 和 $(w,u) \in R$ 同时成立；（ii）为每个等价类创建一个等价类节点，其节点容量区间为该等价类包含的所有节点上容量区间的交集；（iii）创建一条连接两个等价类节点的边当且仅当 Q 中存在端点分别在上述两个等价类的边。另外，将 r 设置为 G 的半径，可以验证 $Q \lhd_r G$ 成立。

（II）而后证明当算法返回“否”时，模式图 Q 是不可满足的。如果算法返回“否”，则一定是以下两种情况之一：（i）$Q \not\prec Q$，或（ii）$Q \prec Q$ 但是不满足节点容量要求。本文用反证法证明上述结论。首先证明情况（i），当 $Q \not\prec Q$ 时，假设 Q 是可满足的。为了表述方便，用 $Q_1 \not\prec Q_2$ 表示 $Q \not\prec Q$，其中 Q_1，Q_2 和 Q 是完全相同的模式图。由 Q 是可满足的可知，存在一个数据图 G 满足 $Q_1 \lhd_r G$，$Q_2 \lhd_r G$ 并且 $Q_1 \lhd_r G_s$，其中 G_s 是 G 中的完美匹配子图。根据组合模拟和图模拟语义的定义，由 $Q_2 \lhd_r G$ 可以得到 $G_s \prec Q_2$ 成立；由 $Q_1 \lhd_r G_s$ 可以得到 $Q_1 \prec G_s$ 成立；由 $Q_1 \prec G_s$ 和 $G_s \prec Q_2$ 可以得到 $Q_1 \prec Q_2$ 成立。这与假设矛盾，结论得证。再证明情况（ii），当 $Q \prec Q$，其最大二元匹配关系为 R_M，存在一个节点对 $(u,v) \in R_M$，其中 u 的容量区间为 $[x_u, y_u]$，v 的容量区间为 $[x_v, y_v]$，并且

$x_v > y_u$ 时，假设 Q 是可满足的。由 Q 是可满足的可知，存在一个数据图 G 满足 $Q \lhd_r G$。由 $Q \prec Q$ 并且 $(u,v) \in R$ 可知，对于 G 中任意节点 w，如果 w 与模式图节点 v 匹配，则 w 一定匹配模式图节点 u。也就是说，与节点 u 匹配的节点数目一定大于与节点 v 匹配的节点数目。然而，这与 u 的容量上界即 y_u 小于 v 的容量下界即 x_v 假设矛盾。

B.2　定理 4.2 证明

本文首先证明定理 4.2（1）成立。已知 $Q \prec \hat{G}[v,t]$，假设 M_t 是 Q 在 $\hat{G}[v,t]$ 中的最大二元匹配关系。由于 $\hat{G}[v,t]$ 是 $\hat{G}[v,r]$ 的一个子图，所以一定存在一个二元匹配关系 M_r，满足对于任意 $(u,v) \in M_t$，$(u,v) \in M_r$ 成立。其中，u 是一个模式图节点，v 是 u 在 $\hat{G}[v,t]$ 中的匹配节点。由于 M_t 是 Q 在 $\hat{G}[v,t]$ 中的最大二元匹配关系并且 $M_t \subset M_r$ 成立，根据图模拟的定义，则 $Q \prec \hat{G}[v,r]$ 成立并且 M_r 是 Q 在 $\hat{G}[v,r]$ 中的二元匹配关系。

本文通过以下引理证明定理 4.2（1）成立。

引理 B.1: 对于任意数据图 G_1 和 G_2，和模式图 Q，如果 G_1 是 G_2 的一个子图，则 $M_1 \subset M_2$ 成立。其中，$M_i\ (i = 1,2)$ 是 Q 在 G_i 中基于图模拟语义的最大二元匹配关系。　□

证明　本文用反证法证明该引理成立。已知 G_1 是 G_2 的一个子图，并且假设 $M_1 \not\subset M_2$ 成立。也就是说，存在节点对 $(u,v) \in M_1$，使得 $(u,v) \notin M_2$。其中，u 是一个模式图节点，v 是与 u 匹配的数据图节点。根据图模拟语义的定义，由 $(u,v) \in M_1$ 可知，对于 Q 中 u 的任意子节点 u'，G_1 中存在 v 的子节点 v' 满足 $(u',v') \in M_1$。由 $(u,v) \notin M_2$ 可知，Q 中节点 u 存在一个子节点 u''，但是 G_1 中节点 v 不存在子节点 v'' 使得 $(u'',v') \in M_2$ 成立。计算基于图模拟的最大二元匹配关系的过程是一个迭代过程，该过程从初始化后的匹配关系 M 中迭代的去除经验证不满足匹配关系的数据图节点。由于 G_2 中不存在节点 u'' 使得 $(u'',v'') \prec M_2$ 成立，所以将 (u,v) 从 M_2 中去除。由于 G_1 是 G_2 的子图，需要将 (u,v) 从 M_1 中去除，所以 $(u,v) \notin M_1$。这与 $(u,v) \in M_1$ 的假设矛盾，引理得证。　□

根据引理 B.1，由于 $\hat{G}[v,t]$ 是 $\hat{G}[v,r]$ 的一个子图，所以 $M' \subset M$ 成立。根据图模拟语义匹配图的定义，根据 $M' \subset M$ 可知，G'_s 是 G_s 的子图。

B.3　定理 4.3 证明

本文采用 locally persistent 算法[49] 来证明增量问题计算复杂性。本文采用这类算法的概念来证明 kDGPM 问题为无界增量问题。

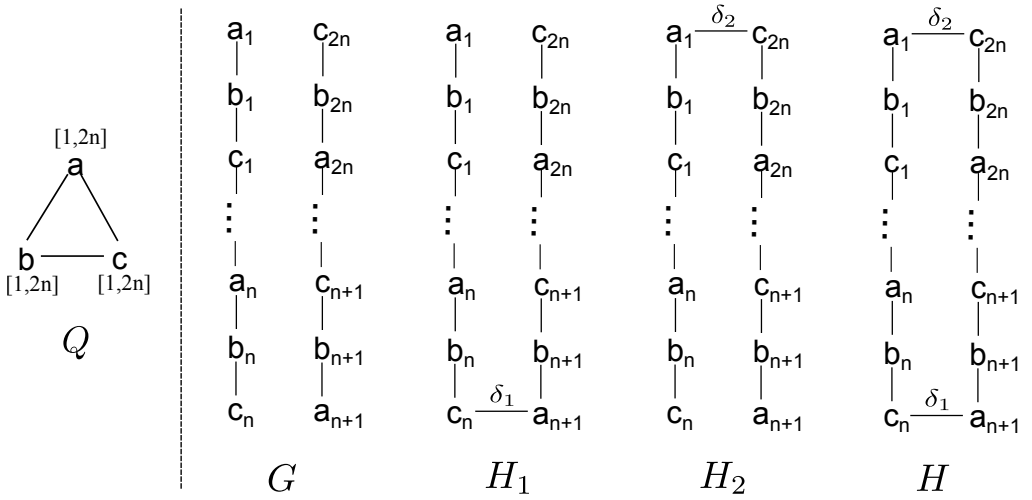


图 B.1　定理 4.3 证明所用到的单元数据图更新构造

以下证明严格遵照 [49] 中增量问题证明的步骤。

（*I*）*k*DGPM 对于单元数据图更新是无界问题。以图 B.1 中的模式图、数据图和数据图更新为例。数据图 G 由两个链状子图构成，即 $(a_1, b_1, c_1, \ldots, a_n, b_n, c_n)$ 和 $(a_{n+1}, b_{n+1}, c_{n+1}, \ldots, a_{2n}, b_{2n}, c_{2n})$，其中 a_i 的标签为 A，b_i 的标签为 B，c_i 的标签为 C。模式图 Q 是由 a，b 和 c 三个节点构成的三角形，其中 a 的标签为 A，b 的标签为 B，c 的标签为 C。分别考虑两个单元数据图更新 $\delta_1 = (c_n, a_{n+1})^+$ 和 $\delta_2 = (c_{2n}, a_1)^+$。令 $k = 1$，$r = 6n$。用 H_1 表示 $G \oplus \delta_1$，用 H_2 表示 $G \oplus \delta_2$。显然，$L_k(Q, G) = L_k(Q, H_1) = L_k(Q, H_2) = \emptyset$，并且 $L_k(Q, H_1 \oplus \delta_2) \neq \emptyset$。假设 kDGPM 问题存在一个 locally persistent 增量算法 $\mathcal{A}$。用 $Trace(G', \delta')$ 表示算法 $\mathcal{A}$ 在处理数据图 G' 上的数据图更新 δ' 时的所有计算步骤。考虑以下两种情况：处理 G 上的更新 δ_2 和处理 H_1 上的更新 δ_2。显然，上述两中情况的计算过程肯定不同，$Trace(G, \delta_2)$ 与 $Trace(H_1, \delta_2)$ 是不同的（由于 $H_1 \oplus \delta_2$ 中的很多节点是受影响节点，然而 $G \oplus \delta_2$ 中不存在受影响节点）。由于 locally persistent 算法不使用全局数据，所以 $Trace(G, \delta_2)$ 和 $Trace(H_1, \delta_2)$ 必定访问了 G 和 H_1 中的同一节点 w，而该节点在 G 和 H_1 中分别存储了不同的局部信息。然而，H_1 是由在 G 上施加更新 δ_1 得出，所以 G 中节点 w 上的信息肯定在处理 δ_1 时被改变了。所以，$Trace(G, \delta_1)$ 一定访问了节点 w。Locally persistent 算法的一个特征是，如果在处理 G' 上更新 δ' 时访问了一个节点 w，则从 δ' 中一个信息被改变的节点到节点 w 在 G' 中路径上的所有节点一定都会被访问。所以，$Trace(G, \delta_1)$ 包含从 δ_1 中节点到节点 w 的访问，$Trace(G, \delta_2)$ 包含从 δ_2 中节点到节点 w 的访问。所以，$Trace(G, \delta_1)$ 包含了从节点 c_n 到节点 c_{2n} 的路径上所有节点的访问，$Trace(G, \delta_2)$ 包含了从节点 a_{n+1} 到节点 a_1 的路径上所有节点的访问。所以，处理 G 上更新 δ_1 和处理 G 上

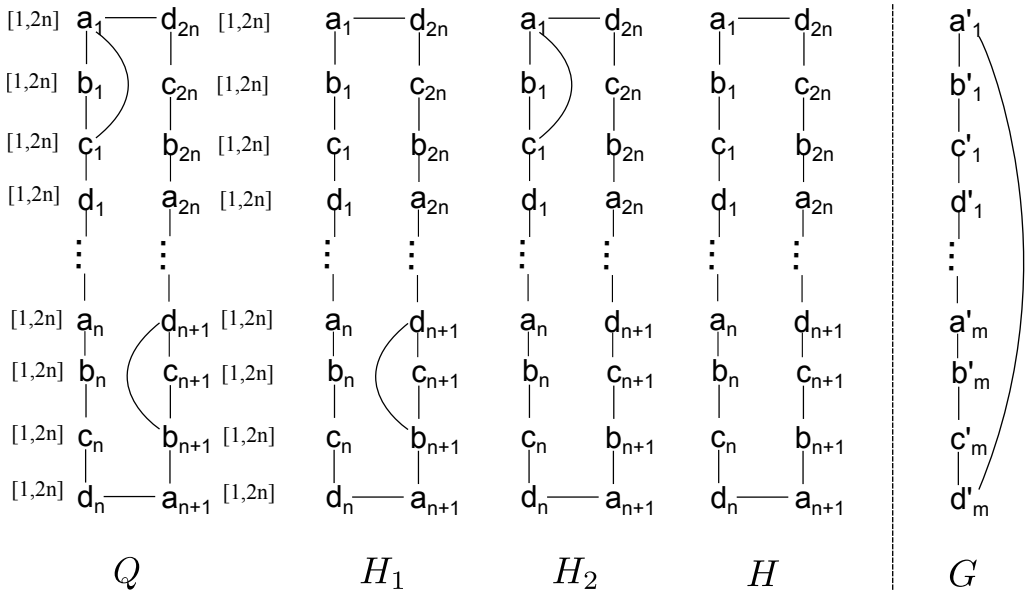


图 B.2　定理 4.3 证明所用到的单元模式图更新构造

更新 δ_2 的时间之和一定不小于 c_n 到 c_{2n} 的距离或者是 a_{n+1} 到 a_1 的距离，也就是 n，所以更新处理时间不是常数。然而，|AFF| 在两种情况下均为 1，从而增量算法 $\mathcal{A}$ 的时间复杂度无法用 |AFF| 的函数表示。所以，$\mathcal{A}$ 不是有界 locally persistent 增量算法。

所以，即使当 $k = 1$ 并且处理单元数据图更新时，kDGPM 依然是无界增量问题。

（*II*）*k*DGPM 对于单元模式图更新是无界问题。以图 B.2 中的模式图、数据图和模式图更新为例。数据图 G 是一个环状图，即 $(a'_1, b'_1, c'_1, d'_1, \ldots, a'_m, b'_m, c'_m, d'_m, a'_1)$，其中 a'_i 的标签为 A，b'_i 的标签为 B，c'_i 的标签为 C，d'_i 的标签为 D。模式图 Q 由一个环状图 $(a_1, b_1, c_1, d_1, \ldots, a_n, b_n, c_n, d_n, a_{n+1}, b_{n+1}, c_{n+1}, d_{n+1}, \ldots, a_{2n}, b_{2n}, c_{2n}, d_{2n}, a_1)$，和两条额外边 (a_1, c_1) 和 (b_{n+1}, d_{n+1}) 构成。其中 a_i 的标签为 A，b_i 的标签为 B，c_i 的标签为 C，d_i 的标签为 D。分别考虑两个单元模式图更新 $\delta_1 = (a_1, c_1)^-$ 和 $\delta_2 = (b_{n+1}, d_{n+1})^-$。令 $k = 1$，$r = 4m$。用 H_1 表示 $Q \oplus \delta_1$，用 H_2 表示 $Q \oplus \delta_2$。显然，$L_k(Q, G) = L_k(H_1, G) = L_k(H_2, G) = \emptyset$, 并且 $L_k(H_1 \oplus \delta_2, G) \neq \emptyset$。假设 kDGPM 问题存在一个 locally persistent 增量算法 $\mathcal{A}$。用 $Trace(Q', \delta')$ 表示算法 $\mathcal{A}$ 在处理模式图 Q' 上的模式图更新 δ' 时的所有计算步骤。考虑以下两种情况：处理 Q 上的更新 δ_2 和处理 H_1 上的更新 δ_2。显然，上述两中情况的计算过程肯定不同，$Trace(Q, \delta_2)$ 与 $Trace(H_1, \delta_2)$ 是不同的（由于 $H_1 \oplus \delta_2$ 在 G 中的很多节点是受影响节点，然而 $Q \oplus \delta_2$ 在 G 中不存在受影响节点）。由于 locally persistent 算法不使用全局数据，所以 $Trace(Q, \delta_2)$ 和 $Trace(H_1, \delta_2)$ 必定访问了 Q 和 H_1 中的同一节点 w，而该节点在 Q 和 H_1 中分别存储了不同的局部信息。然而，H_1 是由在 Q 上施加更

新 δ_1 得出，所以 Q 中节点 w 上的信息肯定在处理 δ_1 时被改变了。所以，$Trace(Q,\delta_1)$ 一定访问了节点 w。根据 locally persistent 算法的特征，如处理单元数据图更新部分所述，$Trace(Q,\delta_1)$ 包含从 δ_1 中节点到节点 w 的访问，$Trace(Q,\delta_2)$ 包含从 δ_2 中节点到节点 w 的访问。所以，$Trace(Q,\delta_1)$ 包含了从节点 a_1 到节点 b_{n+1} 的路径上所有节点的访问，$Trace(Q,\delta_2)$ 包含了从节点 c_1 到节点 d_{n+1} 的路径上所有节点的访问，也就是 $4n$，所以更新处理时间不是常数。然而，|AFF| 在两种情况下均为 1，从而增量算法 $\mathcal{A}$ 的时间复杂度无法用 |AFF| 的函数表示。所以，$\mathcal{A}$ 不是有界 locally persistent 增量算法。所以，即使当 $k=1$ 并且处理单元模式图更新时，kDGPM 依然是无界增量问题。

（I）与（II）一起证明了 kDGPM 问题是无界增量问题，即使当特殊情况 $k=1$ 并且只处理单元模式图更新或单元数据图更新时，kDGPM 依然是无界增量问题。

B.4 定理 4.4 证明

本文通过归纳法证明该定理成立。给定模式图 $Q(V_Q,E_Q)$ 和模式图划分 $\{Q_{f1}, \ldots, Q_{fh}, C\}$，用 $Q^C(V_{Q^C},E_{Q^C})$ 表示节点集为 V_{Q^C}=V_Q，边集为 E_{Q^C}=E_Q/C 的模式图。计算基于图模拟的图匹配查询过程是一个迭代过程，该过程从初始化后的匹配关系中迭代的去除经验证不满足匹配关系的数据图节点，如图模拟查询过程 ugSim 所示（表 10）。用 M_C^k 表示 Q^C 在 $\hat{G}$ 中第 k 个迭代轮次的二元匹配关系；用 M_i^k 表示 Q_{fi} 在 $\hat{G}$ 中第 k 个迭代轮次的二元匹配关系；用 M^k 表示 Q 在 $\hat{G}$ 中第 k 个迭代轮次的二元匹配关系。根据图模拟的定义可知 $M_C^k=\bigcup_{i=1}^h M_i^k$ 成立。本文通过证明 $M^k \subseteq M_C^k$ 在每个迭代轮次都成立来证明 $M \subseteq \bigcup_{i=1}^h M_i$ 成立。

（1）对于 $k=0$，即图模拟算法的初始化步骤，该算法为每个模式图节点计算其在数据图中的候补匹配节点集合。由于 Q 和 Q^C 具有相同的节点集合，所以 $M^0=M_C^0$ 成立。

（2）对于 $k=n$ $(n\geq 0)$，如果 $M^n \subseteq M_C^n$ 成立，下面证明 $M^{n+1} \subseteq M_C^{n+1}$ 在第 $(n+1)$ 个迭代轮次成立。假设 $(u,w)\in M^n$ 和 $(u,w)\in M_C^n$ 成立，并且假设在第 $(n+1)$ 轮，由于 u 在 Q_C 中存在一个邻居节点 u'，但是 w 在 G 中不存在邻居节点 u' 使得 $(u',w')\in M_C^n$ 成立。所以需要将节点对 (u,w) 从 M_C^n 中去除掉。并且由于 $E_{Q^C}\subseteq E_Q$，也就是说 E_{Q^C} 中的边 (u,u') 一定也属于 E_Q，所以也要将 (u,w) 从 M^n 中去除。所以，$M^{n+1}\subseteq M_C^{n+1}$ 成立。

根据（1）和（2），$M \subseteq \bigcup_{i=1}^h M_i$ 得证。

B.5　定理 4.5 证明

模式图优化分块问题的判定问题 dOFGP(Q,h,r_1,r_2) 指给定模式图 Q 和正整数 h，判定是否存在 Q 的一个 h-分块 $\{Q_{f1},\ldots,Q_{fh},C\}$，满足（a）$\max_i|Q_{fi}|\leq r_1\frac{|Q|}{h}$ 并且（b）$|C|\leq r_2|Q|$ 成立。

证明 dOFGP 是 NP 类问题。本文首先通过设计 dOFGP 问题的 NP 算法来证明 dOFGP 问题是 NP 类问题。该算法的主要步骤如下所示：

（a） 猜测 Q 的一个 h-分块。

（b） 验证 h-分块是否满足 dOFGP 问题的两个约束条件（a）和（b）。如果满足，则返回“是”；否则继续步骤（a）。

上述算法是 dOFGP 问题的 NP 算法，因为步骤（b）可以在 PTIME（线性时间）内执行完毕。

证明 dOFGP 是 NP-难问题。本文通过将 3SAT 问题归约至 dOFGP 问题来证明 dOFGP 是 NP-难问题。3SAT 问题的一个实例是，给定一个公式 $\psi=C_1\wedge\ldots\wedge C_m$，其中每个从句 C_i 都具有语法形式 $C_i=\ell_1^i\vee\ell_2^i\vee\ell_3^i$ $(i\in[1,m])$，其中 ℓ_j^i $(j\in[1,3])$ 是全域集合 $U=\{x_1,\ldots,x_n\}$ 中的一个变量 x_k 或者是 x_k 的否定，即 $\bar{x}_k$ $(k\in[1,n])$。给定一个公式 ψ，3SAT 问题即判定 ψ 是否是可满足的，即是否存在一个对 U 中变量的真值赋值 μ，使得 ψ 在 μ 下为真。3SAT 问题是 NP-难问题 [121]。

给定 3SAT 问题的一个公式 ψ，归约过程构建 dOFGP 问题的一个实例，即模式图 Q，分块正整数 h，两个比率 r_1 和 r_2，使得 ψ 是可满足的当且仅当 dOFGP(Q,h,r_1,r_2) 为真。构建方法如下。

（a）模式图 Q 按以下两个步骤构建：（a.1）构建模式图 Q'；（a.2）扩展 Q' 为 Q。

（a.1）Q' 按如下方法生成节点：对于每个 C_i $(i\in[1,m])$，生成一个由三个节点 u_1^i，u_2^i，u_3^i 构成的节点集合 V_i。所以总共生成了 $V_1,\ldots,V_m$ 中的 $3m$ 个节点。直观理解，节点 u_j^i $(i\in[1,m],j\in[1,3])$ 编码了 ℓ_j^i。而后，按如下方法生成边：对于每对正整数 $i,j\in[1,m]$ 并且 $i\neq j$，如果 V_i 中节点 u_s^i 和 V_j 中节点 u_t^j $(s,t\in[1,3])$ 没有编码两个互补的变量，即 $\ell_t^j\neq\bar{\ell}_s^i$，则生成边 (u_s^i,u_t^j)。换而言之，当以下两种情况时，u_t^j 和 u_s^i 间不存在边：对于任意 $x\in U$，$\ell_s^i=x$ 并且 $\ell_t^j=\bar{x}$ 时，或者 $\ell_s^i=\bar{x}$ 并且 $\ell_t^j=x$ 时。

（a.2）而后，按如下方法扩展 Q' 为 Q：对于 Q' 中每个节点集合 V_i，为其连接一个 $(m-2)$-

clique K_i，使得对于 V_i 中每个节点 u_j^i ($j \in [1,3]$)，u_j^i 与 K_i 中每个节点相连。

（b）令 $h = m + 1$。

（c）令 $r_1 = (m+1) * (C_m^2 + m)/|Q|$，即 $r_1 * \frac{|Q|}{h} = C_m^2 + m$ 是一个 m-clique 的大小。

（d）令 $r_2 = \frac{|E_Q| - (C_m^2 + m) - m * (C_m^2 + m - 1)}{Q}$。

可以证明，ψ 是可满足的当且仅当存在一个 Q 的 h-分块 $\{Q_{f1}, \ldots, Q_{fh}, C\}$，满足 $\max_i |Q_{fi}| \leq r_1 \frac{|Q|}{h}$ 和 $|C| \leq r_2|Q|$ 成立。该结论可以由以下归约过程的两个性质证明：

- ψ 是可满足的当且仅当 Q' 存在一个 m-clique；
- Q 只存在一种 $m+1$-分块，满足：（i）m-clique 是最大的分块，（ii）割边集大小不大于 $|E_Q| - (C_m^2 + m) - m * (C_m^2 + m - 1)$：$m$-clique 肯定是从 Q' 中得到的，使得 Q 中扩展的 m 个 $(m-2)$-clique 为剩下的 $m-1$ 个分块，并且每个分块的大小不小于一个 m-clique 的大小减 1。

B.6 定理 4.7 证明

定理 4.7 的正确性由以下性质保证。当模式图更新 ΔQ 产生时，完美匹配子图只会出现在与 $Q \oplus \Delta Q$ 的所有分块都匹配的球结构中。算法 IdABall 能够标识出与 Q 的所有分块都匹配的球结构，即受更新影响的球结构集合 AffBs。否则，如果存在 Q 的一个分块 Q_{fi} 与一个球结构 AffB 无法匹配，则 ΔQ 中一定存在 Q_{fi} 上的删除边/节点操作，使得上述球结构存在与更新后的模式图分块 $Q_{fi} \oplus \Delta Q$ 匹配的可能。所以，算法 IdABall 过滤掉所有与 Q 中模式图分块不匹配并且该分块上不存在删除边/节点操作的球。

B.7 定理 4.9 证明

定理 4.9 的证明与定理 4.7 的证明类似。算法 IdABall 过滤掉所有不能与 Q 中所有模式图分块匹配的球，并且该球中不存在数据图更新。

附录 C　章节五定理证明

C.1　定理 5.3 证明

（1）本文首先证明 $\mathsf{kPR}_{\mathsf{DF}}$ 问题是 NP 类问题，而后再证明该问题是 NP-难问题。$\mathsf{kPR}_{\mathsf{DF}}$ 的判定问题定义如下：

- 输入：模式图 Q，语义分类图 T，正整数 μ 和 k，正实数 B。
- 问题：判断是否存在 Q 的基于 T 的由 k 个 μ-有界查询松弛组成的集合 S 满足 $F(Q,\Delta) \leq B$。

证明 $\mathsf{kPR}_{\mathsf{DF}}$ 是 NP 类问题。本文首先通过设计 $\mathsf{kPR}_{\mathsf{DF}}$ 问题的 NP 算法来证明 $\mathsf{kPR}_{\mathsf{DF}}$ 问题是 NP 类问题。该算法的主要步骤如下所示：

（a） 猜测一个 Q 的基于 T 的由 k 个 μ-有界查询松弛组成的集合 S。

（b） 验证 S 中的查询松弛是否都是 μ-有界查询松弛并且满足 $F(Q,S) \leq B$ 成立。如果成立，则返回“是”；否则继续步骤（a）。

证明 $\mathsf{kPR}_{\mathsf{DF}}$ 是 NP-难问题。本文通过将 *K-clique* 问题归约到 $\mathsf{kPR}_{\mathsf{DF}}$ 问题来证明 $\mathsf{kPR}_{\mathsf{DF}}$ 是 NP-难问题。*K-clique* 问题的一个实例是，给定数据图 $G(V_G, E_G)$ 和正整数 K，判定 G 中是否存在一个子图是由 k 个节点构成的完全图，即 K-clique。给定一个 *K-clique* 问题的实例，即数据图 $G(V,E)$ 和正整数 K，归约过程构建 $\mathsf{kPR}_{\mathsf{DF}}$ 问题的一个实例，即模式图 Q，数据图 G'，语义分类图 T，正整数 μ 和 k，正实数 $\lambda \in [0,1]$ 和 B，满足 G 中存在一个 K-clique 当且仅当 Q 存在基于 T 的由 k 个 μ-有界查询松弛组成的集合 S，使得 $F(Q,S) \leq B$ 成立。构建方法如下。

（a）模式图 Q 包含一个孤立节点 u，节点标签为 $f_Q(u) = A_0$。

（b）语义分类图 $T(V_T, E_T, l_T)$ 按如下方法构建：

- 节点集 V_T 包括 $|V| + |E| + 1$ 个节点 $u_0, u_1, \ldots u_{|V|}, w_1, \ldots, w_{|E|}$。
- 边集 E_T 包括 $(u_0,u_1), \ldots, (u_0,u_{|V|})$, 和 (u_i,w_j)。其中，对于任意 $(v_i,e_j) \in V_G \times E_G$ 并且在 G 中节点 v_i 不是边 e_j 的端点，则 (u_i,w_j) 为 T 中的边。
- $l_T(u_i) = A_i$ $(i \in [0,|V_G|])$；$l_T(w_j) = B_j$ $(j \in [1,|E_G|])$。其中，$A_0, \ldots, A_{|V|}, B_1, \ldots, B_{|E|}$ 是互不相同的标签。

直观理解，$u_1, \ldots, u_{|V|}$ 编码了数据图 G 中节点，$w_1, \ldots, w_{|E|}$ 编码了数据图 G 中的边。

（c）数据图 G' 与语义分类图 T 完全相同。

（d）令 $\lambda = 0$；$\mu = 1$；$k = K$；$B = 0$。

下面证明 G 中存在一个 K-clique 当且仅当 Q 存在基于 T 的由 k 个 1-有界查询松弛组成的集合 S，满足 $F(Q,S) \leq B$ 成立。

$\boxed{\Rightarrow}$ 假设 G 中存在一个 K-clique $G_K = (V_K, E_K)$。归约过程按如下方法构建一个查询松弛 K-集合 S：对于 V_K 中每个节点 v_i，构建一个查询松弛 $\{A_0 \to A_i\}$ 并入集合 S 中。观察可知，对于 S 中任意两个查询松弛 $\Delta_i = \{A_0 \to A_i\}$ 和 $\Delta_j = \{A_0 \to A_j\}$，$\theta(\Delta_i, \Delta_j) = 0$ 成立。并且对于任意两个不属于 S 的查询松弛 $\Delta_p = \{A_0 \to A_p\} \notin S$ 和 $\Delta_q = \{A_0 \to A_q\} \notin S$，$\theta(\Delta_i, \Delta_p) > 0$ 并且 $\theta(\Delta_i, \Delta_q) > 0$ 成立。因此 $F(Q,S) = 0 = B$。

$\boxed{\Leftarrow}$ 假设存在 Q 的基于 T 的由 k 个 1-有界查询松弛组成的集合 S 满足 $F(Q,S) \leq B$，则 $F(Q,S) = 0$。下面证明 S 编码了 G 中的一个 K-clique。考虑 G 中的一个节点子集 $V_K \subseteq V$。其中，G 中的节点 v_i 属于 V_K 当且仅当 $\{A_0 \to B_i\}$ 为 S 中的一个查询松弛。分析可知，对于 V_K 中任意两个节点 v_i 和 v_j，因为 $\Delta_i = \{A_0 \to A_i\}$ 和 $\Delta_j = \{A_0 \to A_j\}$ 属于 S，所以 $\theta_Q(\Delta_i, \Delta_j) = 0$ 成立。由 T 的构建方式可知，G 中一定存在一条边 e 与节点 v_i 和 v_j 均相连。所以 G 中由节点 V_K 产生的诱导子图为一个 K-clique。

（2）可以验证上述归约过程同样是一个由 *Maximum clique problem* 问题到 $\mathsf{kPR}_{\mathsf{DF}}$ 问题的 AP-归约过程 [73]。由于 *Maximum clique problem* 问题是 APX-难问题，所以 $\mathsf{kPR}_{\mathsf{DF}}$ 问题也是 APX-难问题。

C.2　定理 5.5 证明

本文首先证明 MRE 问题是 NP 类问题，而后再证明该问题是 NP-难问题。MRE 的判定问题定义如下：

- 输入：模式图 Q，数据图 G，语义分类图 T，由算法 relDF 返回的 Q 的基于 T 的 k 个查询松弛 $\Delta_1, \ldots, \Delta_k$，正整数 i，$(Q \oplus \Delta_i)(G)$ 中节点 v，正整数 B。
- 问题：判断是否存在一个 v 的基于 Δ_i 的溯源解释 $\mathcal{E}_{\Delta_i}(v)$ 满足 $|\mathcal{E}_{\Delta_i}(v)| \leq B$。

证明 MRE 是 NP 类问题。本文首先通过设计 MRE 问题的 NP 算法来证明 MRE 问题是 NP 类问题。该算法的主要步骤如下所示：

（a） 猜测查询松弛 Δ_i 的一个子集 Δ'。

（b） 验证 $v \in (Q \oplus \Delta')(G)$ 和 $|\Delta'| \leq B$ 是否成立。如果成立，则返回“是”；否则继续步骤（a）。

上述算法是正确的。因为该算法最多执行 $|\Delta|$ 的指数多次猜测步骤（a），并且步骤（b）可以在 $|Q|$，$|G|$，$|T|$ 和 $|\Delta|$ 的多项式时间内执行完毕。

证明 MRE 是 NP-难问题。本文通过将 *Set cover* 问题归约到 MRE 问题来证明 MRE 是 NP-难问题。*Set cover* 问题的一个实例是，给定一个全域集合 U 包括 n 个元素 $e_1, \ldots, e_n$，一个集合 F 包括 m 个 U 的子集 $S_1, \ldots, S_m$，整数 K，判定 F 中是否存在一个子集 C 满足 $\bigcup_{S \in C} S = U$ 和 $|C| \leq K$ 成立，则称 C 是 U 的一个覆盖。

给定一个 *Set cover* 问题的实例，即 U, F 和 K，归约过程构建 MRE 问题的一个实例，即模式图 Q，数据图 G，语义分类图 T，k 个查询松弛 $\Delta_1, \ldots, \Delta_k$，正整数 $p \in [1, k]$，节点 $v \in (Q \oplus \Delta_p)(G)$ 和整数 B，满足 U 存在一个包含子集个数不大于 K 的覆盖 C 当且仅当存在一个 v 的基于 Δ_p 的溯源解释 $\Delta'(v)$，使得 $|\Delta'(v)| \leq B$ 成立。构建方法如下。

（1）模式图 $Q(V_Q, E_Q, f_Q)$ 按如下方法构建：

- V_Q 包括 $1 + m + n$ 个节点：$u_U, u_{S_1}, \ldots, u_{S_m}, u_{e_1}, \ldots, u_{e_n}$。
- E_Q 包括 $n + \sum_{S \in F} |S|$ 条边：对于每个 $i \in [1, n]$，(u_U, u_{e_i}) 为 E_Q 中的边；对于每个 $i \in [1, m]$ 和每个满足 $e_j \in S_i$ 的 j，(u_{e_j}, u_{S_i}) 为 E_Q 中的边。
- $l_Q(u_U) = l_U$；对于每个 $i \in [1, m]$，$l_Q(u_{S_i}) = l_{S_i}$；对于每个 $i \in [1, n]$，$f_Q(u_{e_i}) = l_{e_i}$。

（2）数据图 $G(V_G, E_G, l_G)$ 按如下方法构建：

- V_G 包括节点 $v_U, v_{S_1}, \ldots, v_{S_m}, v_{e_1}, \ldots, v_{e_n}, w_U, w_{S_1}, \ldots, w_{S_m}, w_{e_1}, \ldots, w_{e_n}$;
- E_G 包括：对于每个 $i \in [1, n]$，(v_U, v_{e_i}) 和 (w_U, w_{e_i}) 为 E_G 中的边；对于每个 $i \in [1, m]$ 和每个满足 $e_j \in S_i$ 的 j，(v_{e_j}, v_{S_i}) 和 (w_{e_j}, w_{S_i}) 为 E_G 中的边。
- $f(u_U) = f(w_U) = l_U$; 对于每个 $i \in [1, m]$, $f(u_{S_i}) = l_{S_i}$, $f(w_{S_i}) = l'_{S_i}$; 对于每个 $j \in [1, n]$, $f(u_{e_j}) = f(w_{e_j}) = l_{e_j}$。

（3）语义分类图 $T(V_T, E_T)$ 按如下方法构建（这里用标签表示 V_T 中节点）：

- V_T 包括节点 $l'_{S_1}, \ldots, l'_{S_m}, l_{S_1}, \ldots, l_{S_m}$;
- E_T 包括：对于每个 $i \in [1, m]$，(l'_{S_i}, l_{S_i}) 为 E_T 中的边。

（4）令 $k = 2$。

（5）令 $\Delta_1 = \emptyset$ 并且 Δ_2 为 $\{(l_{S_i} \to l'_{S_i}) \mid i \in [1, m]\}$。注意通过在 G 和 T 中增加冗余节点该结果一定能被算法 relDF 计算得出。

（6）令 $p = 2$ 并且 v 为 w_U。注意 w_U 为图匹配查询结果 $(Q \oplus \Delta_2)(G)$ 中的一个节点。

（7）令 $B = K$。

下面证明存在一个 v 的基于 Δ_p 的溯源解释 $\Delta'(v)$ 并且 $|\Delta'(v)| \leq B$ 当且仅当存在 $C \subseteq F$，满足 $\bigcup_{S \in C} S = U$ 和 $|C| = K$ 成立。

$\boxed{\Rightarrow}$ 假设存在一个 v 的基于 Δ_2 的溯源解释 $\Delta(v) \subseteq \Delta_2$ 满足 $|\Delta(v)| \leq K$。归约过程按如下方法构建一个 U 的覆盖 C：对于 $\Delta(v)$ 中的每个查询松弛 $l_{S_i} \to l'_{S_i}$，将集合 S_i 并入 C 中。所以 $|C| = |\Delta(v)| \leq K$ 成立。因为 $w_U \in (Q \oplus \Delta(v))(G)$，所以节点 $w_{e_1}, \ldots, w_{e_n}$ 必然都是 $(Q \oplus \Delta(v))(G)$ 中图匹配查询结果。匹配节点 w_{e_i} 对应于 U 中元素 e_i。所以 C 是 U 的一个覆盖，即 U 存在一个 K-覆盖。

$\boxed{\Leftarrow}$ 假设 U 存在一个 K-覆盖 $C = \{S_1, \ldots, S_K\}$。归约过程按如下方法构建一个 v 的基于 Δ_2 的溯源解释 $\Delta(v)$：对于 C 中每个 S_j $(j \in [1, K])$，将查询松弛 $l_{S_j} \to l'_{S_j}$ 并入 $\Delta(v)$ 中。由于 C 是 U 的一个覆盖，节点 $w_{e_1}, \ldots, w_{e_n}$ 与节点 w_{S_j} $(j \in [1, K])$ 相连。根据图模拟语义定义，所以 $w_U \in (Q \oplus \Delta(v))(G)$ 成立，并且 $\Delta(v)$ 是 v 的基于 Δ_2 的溯源解释，满足 $|\Delta(v)| = |C| = K$ 成立。

C.3　定理 5.6 证明

根据算法 relTF，如果 Δ 是第 k 个查询松弛结果，则对于任意一个查询松弛 $\Delta' \subsetneq \Delta$，$\Delta'$ 肯定是第 m $(m < k)$ 个查询松弛结果。对于任意一个 $(Q \oplus \Delta_i)(G)$ 中节点 v，一定存在 $j \in [1, i]$ 满足 $\Delta_j \subsetneq \Delta_i$ 并且 Δ_j 是 v 的基于 Δ_i 的最小溯源解释。

攻读博士学位期间取得的学术成果

一、发表论文

1、 **Jia Li**, Yang Cao, Shuai Ma: Relaxing Graph Pattern Matching With Explanations. CIKM 2017:1677-1686, DOI: 10.1145/3132847.3132992. (EI, CCF B 类会议)

2、 **Jia Li**, Yang Cao, Xudong Liu: Approximating graph pattern queries using views. CIKM 2016:449-458, DOI: 10.1145/2983323.2983766. (EI, CCF B 类会议)

3、 Shuai Ma, **Jia Li**, Chunming Hu, Xuelian Lin, Jinpeng Huai: Big graph search: challenges and techniques. FCS 2016, 10(3), 387-398, DOI: 10.1007/s11704-015-4515-1. (SCI, CCF C 类期刊)

4、 Shuai Ma, **Jia Li**, Chunming Hu, Xudong Liu, Jinpeng Huai: Graph Pattern Matching for Dynamic Team Formation. arXiv:1801.01012，2018. (Preprint)

5、 马帅，**李佳**，刘旭东，怀进鹏: 大数据时代的图搜索技术. 《信息通信技术》2013 年第 6 期，44-51 页。

6、 马帅，**李佳**，刘旭东，怀进鹏: 图查询：社会计算时代的新型搜索. 《中国计算机学会通讯》2012 年第 8 卷第 11 期，26-31 页。

二、申请专利

1、 马帅，**李佳**，胡春明，刘旭东，怀进鹏．一种应用于社交网络中的动态图匹配查询方法，中国发明专利（201711435481.9），已受理。

2、 马帅，**李佳**，曹洋，刘旭东，怀进鹏．一种基于查询松弛结果增强的图匹配查询方法，中国发明专利（201710569486.4），公开日 2017 年 12 月 8 日。

三、主要参与项目

1、 2014-2018，网络信息空间大数据计算理论（2014CB340300）子课题“大数据的计算复杂性与算法理论”，国家重点基础研究发展计划（973 计划）项目；

2、 2014.01-2016.12, 数据库理论与系统（61322207），国家自然科学基金委优秀青年科学基金资助。

致谢

首先，我要感谢我的两位导师刘旭东教授和马帅教授。非常感谢刘老师宝贵的指导、信任、支持和耐心。感谢刘老师对我的指导，感谢刘老师在每当我遇到瓶颈时如启明灯塔般的指引；感谢刘老师对我的信任，感谢刘老师对我每一个深思熟虑的决定提供强有力的支持；感谢刘老师对我的耐心，感谢刘老师尽管工作任务繁重却总是为学生提供及时的指导与帮助。刘老师对待科研的严谨认真、对待工作的踏实勤奋、对待问题的真知灼见、对待学生春风化雨般的关怀都是我辈学习的楷模。非常感谢马老师在科研道路上给予我一路的教诲、栽培、奉献和熏陶。感谢马老师在第一次对我的指导中就为我树立正确的科研标杆，指明清晰的努力方向，那些教导时至今日依然言犹在耳、掷地有声；感谢马老师在科研进程中与我无数次的讨论，马老师对研究方向的把握、对问题精辟的分析和对研究精益求精的态度令我每次讨论后都受益匪浅；感谢马老师教书育人时的孜孜不倦，感谢马老师在夜以继日的忙碌于工作之时，只要我告知遇到了问题，总会在第一时间安排讨论与指导；感谢马老师在人生道路上对我的鼓励与支持，感谢马老师一路给予我的帮助与耐心。再次感谢刘老师和马老师，我非常幸运能够得到两位导师的教导和支持。

我同样要感谢英国爱丁堡大学的樊文飞教授。感谢樊老师为我提供宝贵的爱丁堡大学一年访问学习机会，感谢樊老师为我提供强有力的支持，让我在博士生涯中能够在世界一流大学和世界一流科研团队体验学习，让我能有幸聆听数据库领域世界顶级专家的教诲。樊老师渊博的学识、敏捷的思维、高端的科研品味让我见识到世界级大师的风采。

感谢怀进鹏院士成立 ACT 实验室，让我自本科大三进入实验室至今，能够在实验室渡过八年最美好的时光。感谢沃天宇老师在我攻读博士期间给予我的帮助与鼓励。感谢马殿富教授、韩军老师、李欢老师、胡春明老师、李建新老师、孙海龙老师、张日崇老师、林学练老师、赵永望老师、李博老师、邓婷老师、王旭老师给予我的帮助，感谢各位老师在博士讨论班期间的交流与指导。感谢各位老师对实验室的付出和辛劳，让实验室具备如此优异的科研环境。

感谢赵庶娟师姐、李春娥师姐、王莹师妹、袁薇、龚显丽，感谢你们在我攻读博士期间不同阶段的陪伴，感谢你们明媚了我的生活，感谢你们为我分享快乐和忧伤。感谢胡仁君师弟、段亮师弟，感谢你们每当在我遇到困难之时，总是第一时间帮我排忧解难，感谢那些年一起奋斗的日子。感谢王罡师弟、朱孟笑师妹、张振宇师弟，感谢你们给予

我的帮助，感谢你们在我于英国访问学习之时帮助我处理繁杂的事务。感谢课题组的所有博士生和硕士生，感谢与你们共同经历过的同窗岁月。

特别感谢我的父母。感谢父母对我的至爱，为我撑起一片澄澈的天空，让我任意的徜徉、自由的成长；感谢父母的谆谆教导，教育我立身立德立言立行的美好品质；感谢父母给予我倍至的呵护，却又鼓励我勇于探索不断找寻更好的自己；感谢我在异乡时，远方的那一丝牵挂；感谢在我人生的道路上，有父母的嘱托，感谢父母给予我逐梦的力量。还要感谢我的丈夫，感谢我的丈夫在生活和科研上给予我莫大的帮助；感谢他带着一切美好的词语走进我的生活；感谢他如启明星辰一般照亮我的世界。

最后，我要感谢各位评阅老师，感谢您为这篇论文能够顺利完成付出的精力与时间。感谢您为我提出的宝贵的指导意见和建议。

作者简介

李佳，女，1989 年 11 月 8 日出生于河南省安阳市。2008 年 9 月考入北京航空航天大学计算机学院攻读学士学位，后于 2012 年 9 月继续在北京航空航天大学计算机学院攻读工学博士学位，并在此期间获得北航博士生短期出国访学基金、美国计算机学会 SIGIR 会议资助奖学金等荣誉。

电子邮箱: lijia@act.buaa.edu.cn