分类号 密级

UDC

学 位 论 文

图数据流上增量子图相似性匹配技术的研究与实现

|  |  |  |  |
| --- | --- | --- | --- |
| 作者姓名： | 臧楠棋 | | |
| 指导教师： | 谷峪 副教授 | | |
|  | 东北大学信息科学与工程学院 | | |
| 申请学位级别： | 硕士 | 学科类别：工学 |  |
| 学科专业名称： | 计算机技术 | | |
| 论文提交日期： | 2015年6月 | 论文答辩日期： | 2015年6月 |
| 学位授予日期： |  | 答辩委员会主席： |  |
| 评阅人： |  | | |

东 北 大 学

2015年6月

##### A Thesis in Computer Software and Theory

**Incremental Subgraph Similarity Match**

**Technology research and implementation over Streaming Graph**

By Zang Nanqi

Supervisor: Associate Professor Gu Yu

**Northeastern University**

**June 2015**

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人己经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均己在论文中作了明的说明并表示谢意。

学位论文作者签名：

日 期：

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年 □ 一年□ 一年半□ 两年□

学位论文作者签名： 导师签名：

签字日期： 签字日期：

图数据流上增量子图相似性匹配技术的研究与实现

# 摘 要

近年来，随着图数据结构在通信网络，模式识别，化学/生物信息，社交网络等领域的广泛应用以及数据多样性的不断发展，图数据流模型受到了越来越多专家学者的关注。目前，虽然静态图上的子图匹配技术已得到了广泛并深入的研究，然而图数据流上的子图匹配问题作为图数据流上许多其他操作的基石，还没有得到深入的研究，而这种图数据流模型在网络安全、社交网络等很多方面都有着广泛的使用价值和深远的现实意义。

针对上述问题，本文对图数据流上增量子图相似性匹配问题进行研究。

首先，本文考虑图数据流上的简单更新，即当一条边已经存在时不可再次插入。为了实现本文增量维护的思想，为图数据流设计了一种带有剪枝功效的分区策略——最近邻分区法，并对该方法进行增量维护。这样，当每次更新到达时，通过对发生改变的分区重新进行匹配来取代一次完整的匹配过程。考虑到图数据流不断变化的特性，本文通过为查询图创建生成树集合的方式完成子图匹配，将图-图之间的匹配转化为树-图之间的匹配从而提高了匹配的速度。同时，提出最大匹配的思想，避免了处理冗余的代价。最后在真实数据集和合成数据集上进行实验，并与传统静态图上的方法进行对比，证明了方法的正确性和有效性。

此外，本文还考虑了图数据流上的复杂更新，即同一条边可以被插入或删除多次。该问题是上述问题的进一步深入，因此在最近邻分区的基础上又提出了基于权重的最近邻分区方法，该方法通过权重的判断对图数据流进一步进行剪枝，以加快匹配的速度。同时为了进一步满足图数据流实时响应的特性，提出了一种近似的匹配方法。该方法通过对匹配结果候选集合的维护，在每次更新时直接在结果上进行增量维护，从而进一步加快了匹配的速度，并通过提出的近似误差率保证匹配结果的准确性。最后通过实验验证了提出方法的有效性。

总之，本文从越来越接近实际应用的图数据流模型的子图相似性匹配的典型特征和挑战出发，针对图数据流上的简单更新和复杂更新展开研究。基于增量维护的思想，提出了有效的图数据流上增量子图相似性匹配的处理方法。本文增量维护、最近邻分区等思想为相关课题的展开铺平了道路。

关键词：图数据流；相似性子图匹配；增量维护；最近邻分区；带权图数据流

Incremental Subgraph Similarity Match Technology Research

And Implementation over Streaming Graph

# Abstract

In recent years, with the graph structure is widely used in communication networks, pattern recognition, chemical/biological information, social network and the fast development of the diversity of data, streaming graph model has attracted more and more attention from experts and scholars. At present, the subgraph match technology over static graph has been extensive and in-depth research, however, the subgraph match technology over streaming graph as the foundation of many other operations has not been thorough research, and the steaming graph model has a wide range of use value and profound realistic significance on many aspects, such as network security, social network and so on.

According to the above problem, in this paper, we focus on the research on the problem of incremental subgraph similarity match over steaming graph.

Firstly, this paper consider simple update on steaming graph, namely if there is an edge has already in the steaming graph, we cannot insert the edge again. In order to achieve the ideal of incremental maintenance, this paper designed a partition method with pruning effect for steaming graph, namely the nearest neighbor partition method, and incremental maintain for it. For this, when an update operation arrived, we just need to maintain the changed ones of the partition, instead of a complete matching process. Considering the changing characteristics of steaming graph, we implement the subgraph match by creating spanning tree set for query graph, which changing the matching from graph-graph into tree-graph , to further improves the speed of subgraph matching, and provide the idea of maximum matching, which avoiding the expense of dealing with redundant. Finally, execute the experiments on real data sets and synthetic data sets, and compared with the traditional static graph method, to prove the correctness and effectiveness of the proposed method.

In addition, this paper also considers the complex updates over steaming graph, namely the same edge can be inserted or deleted more than once. The problem is the

further one of the above problem; so based on the nearest neighbor partition we provide the nearest neighbor partition method based on weighted, the method further pruning the steaming graph by weighted judgment, to speed up the matching speed. At the same time in order to further satisfy the real-time response characteristics of steaming graph, we put forward an approximate match method. When an update operation arrived, this method directly incremental maintenance on the results by maintaining the matching results candidate set, so that further accelerate the speed of matching, and ensure the accuracy of the matching results through the proposed approximate error rate. Finally the effectiveness of the proposed method is verified by experiment.

All in all, this paper begin with the typical characteristic and challenge of the subgraph similarity match of the more and more close to the actual application’s steaming graph. We focus the research on simple updates and complex updates. Based on the idea of incremental maintenance, we put forward an effective method on incremental subgraph similarity match over streaming graph. The idea of incremental maintenance, nearest neighbor partition, and so on in this paper has paved the way for the expansion of related topics.

**Key words:** streaming graph; similarity; incremental subgraph match; nearest neighbor partition

目录

[独创性声明 I](#_Toc421107602)

[摘 要 II](#_Toc421107603)

[Abstract III](#_Toc421107604)

[第1章 引言 1](#_Toc421107605)

[1.1研究背景 1](#_Toc421107606)

[1.2问题提出 3](#_Toc421107607)

[1.3本文研究的内容及面临的挑战 4](#_Toc421107608)

[1.3.1主要研究内容 4](#_Toc421107609)

[1.3.2 面临的挑战 4](#_Toc421107610)

[1.4本文主要贡献 5](#_Toc421107611)

[1.5本文组织结构 6](#_Toc421107612)

[第2章 相关工作 7](#_Toc421107613)

[2.1静态图上子图匹配 7](#_Toc421107614)

[2.1.1静态图上子图匹配 7](#_Toc421107615)

[2.1.2静态图上子图全匹配 8](#_Toc421107616)

[2.2带权图上子图匹配 10](#_Toc421107617)

[2.3图数据流 11](#_Toc421107618)

[2.3.1图数据流上的简单更新 11](#_Toc421107619)

[2.3.2图数据流上的复杂更新 13](#_Toc421107620)

[2.4本章小结 14](#_Toc421107621)

[第3章 图数据流上增量子图相似性全匹配 15](#_Toc421107622)

[3.1问题定义 15](#_Toc421107623)

[3.1.1基本概念定义 15](#_Toc421107624)

[3.1.2图数据流上增量子图相似性全匹配问题的定义 18](#_Toc421107625)

[3.2最近邻分区 21](#_Toc421107626)

[3.2.1结构剪枝 22](#_Toc421107627)

[3.2.2最近邻判断 25](#_Toc421107628)

[3.2.3分区 30](#_Toc421107629)

[3.2.4动态维护 31](#_Toc421107630)

[3.3生成树集合 35](#_Toc421107631)

[3.3.1为查询图创建生成树集合 35](#_Toc421107632)

[3.3.2有效的存储方式 41](#_Toc421107633)

[3.4 子图全匹配 42](#_Toc421107634)

[3.4.1 QI-Sequence 43](#_Toc421107635)

[3.4.2子图全匹配 44](#_Toc421107636)

[3.4.3子图全匹配增量维护 46](#_Toc421107637)

[3.5实验 47](#_Toc421107638)

[3.5.1实验环境 47](#_Toc421107639)

[3.5.2数据集 47](#_Toc421107640)

[3.5.3实验结果 48](#_Toc421107641)

[3.6总结 48](#_Toc421107642)

[第4章 带权图数据流上增量子图近似全匹配 49](#_Toc421107643)

[4.1问题定义 50](#_Toc421107644)

[4.1.1基本概念定义 50](#_Toc421107645)

[4.1.2 带权图数据流上增量子图近似全匹配 53](#_Toc421107646)

[4.2子图全匹配 55](#_Toc421107647)

[4.2.1基于权重的最近邻分区 55](#_Toc421107648)

[4.2.2基于权重的最近邻分区的动态维护 63](#_Toc421107649)

[4.2.3创建生成树集 67](#_Toc421107650)

[4.2.4子图全匹配 68](#_Toc421107651)

[4.2.5子图全匹配的增量维护 70](#_Toc421107652)

[4.3实验 75](#_Toc421107653)

[4.3.1实验环境 75](#_Toc421107654)

[4.3.2数据集 75](#_Toc421107655)

[4.3.3运行结果 76](#_Toc421107656)

[4.4总结 76](#_Toc421107657)

[第5章 结论 77](#_Toc421107658)

[5.1本文的主要贡献和结论 77](#_Toc421107659)

[5.2进一步的工作 78](#_Toc421107660)

[参考文献 79](#_Toc421107661)

[致谢 83](#_Toc421107662)

[攻读硕士学位期间的项目情况 85](#_Toc421107663)

# 引言

图搜索已经作为一种有效的工具用于在图形数据库中查询有用的子图结构。例如，在生物化学中可以利用子图搜索分析新发现的化合物的功能；在网络安全中管理员可以对网络流量数据进行模式匹配(子图)，来检测可能的恶意攻击。然而随着科学技术的不断发展，图数据不再拘泥于传统的静态形式，而是以一种流的形式不断更新变化。由于静态图上的子图同构是NP完全问题[1,2,3,4]，不可能实时处理图数据流中的每一个时间戳。所以为了满足图数据流动态更新的特性，近年来国内外广大学者对图数据流上的操作进行了广泛而深入的研究。然而，目前还没有研究集中在图数据流上的子图全匹配问题上，因此本文提出的在图数据流上的增量子图全匹配算法是很有意义的。

本章作文全文的开篇章节，起到抛砖引玉的作用，主要对本文的研究背景，主要研究内容，以及面临的挑战等问题进行了概述。首先对当前研究问题的背景进行了介绍，然后介绍了本文所研究问题的相关实际应用，从而引出了本文所要研究的主要内容，接下从几个方面介绍了本文面临的挑战，因而进一步介绍了本文主要的贡献，最后介绍了文章的组织结构。

## 1.1研究背景

随着相关技术的快速发展，图数据普遍存在于各种实际应用中，包括通信网络，模式识别，化学/生物信息，社交网络等。因此，图上的相关操作也得到了广泛并深入的研究，如最短路径问题[27,28]，可达性问题[17,18]，包含查询问题[5]，关键字搜索问题[32,33]及子图匹配问题[1,2,3,4]等。其中，子图全匹配作为图上的一种重要查询，广泛的应用于模式匹配及信息检索等应用中。然而，随着科学和技术的不断发展，图数据不在只是传统的单纯的静态形式，而是以一种流的形式不断更新变化，从而形成了一种新的数据结构——图数据流（简称图流）。图数据流除了拥有静态图固有的属性外，还增加了自己特有的属性，其中最为关键的就是它是不断动态更新的，为了适应图数据流的特性，就需要为其设计特有的方法来实现相关操作。

对于图数据流上的更新操作通常可以分为简单更新和复杂更新两种，所谓简单更新是指图数据流上的边只能有存在和不存在两种情况，也就是说如果一条边已经存在，这条边就不能再一次被插入，即同一条边不能到达或删除多次；复杂更新则是指图数据流上的同一条边可以多次到达或多次删除。目前大多数图数据流上的研究都是针对简单更新进行的。

虽然，到目前为止已经有许多专家学者对于图数据流上的操作进行了研究，但是还没有研究集中在图数据流上增量子图全匹配的问题上。然而，图数据流上的子图全匹配却是许多其他图流上相关操作的基石，并且在现实生活中也有着广泛的应用。

例如：在蛋白质网络中，给定一组已知的小型蛋白质网络集合（即一组小图集合），研究者们希望能在新发现的大型的蛋白质网络中确定它与哪些已知的小型蛋白质网络相匹配，从而进一步研究新发现蛋白质的性质和功能；此外，在编写大型程序的过程中，给定一组具有某写特定功能的程序集合，编程人员希望在一个新的大型程序中找到包含哪些特定的功能，从而了解程序的功能，并便于对程序进行查询修改。

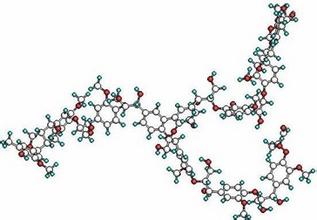
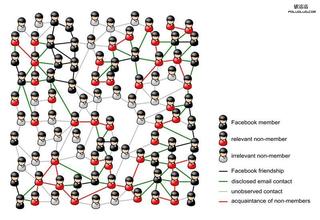
 

图1.1应用背景（化合物和社会网络）

Fig.1.1 Application background (compounds and social network)

然而，在一些情况中简单的边的插入和删除不能满足现实中的需求，此时要求同一条边可以到达多次，同时也可以删除多次。在本文中使用带有权重的图来描述这种可重复到达和删除边并进行更新的图，其中每条边的权重值表示该边存在的次数，即插入则加1，删除则减1。如果两个顶点之间不存在一条边，那么认为两个顶点之间存在一条权重为0的边。例如，在社交网络中不同用户之间每联系一次我们就认为这两个用户之前的边的权值增加一次，在很多情况中，研究者希望查询联系人之间满足某种特定模式，并且联系次数满足一定要求的所有匹配子图，以便了解整个社交网络的结构和相关信息。

由此可见，本文所研究的内容具有重大的现实应用价值，广泛应用在生物化学，社交网络，生物医学等各个领域当中，实现本文所研究的方法，具有广泛的实用价值。

## 1.2问题提出

虽然目前为止对于传统静态图上的子图全匹配问题在[2]，[3]中已经得到了深入的研究，同时图数据流上的一些操作如SimRank评估[16]，pagerank评估[15]，图流上的包含查询[5]，密集子图维护[8,9]等问题也受到了一些学者的关注，但是却没有研究集中在图数据流上增量子图全匹配问题上。然而，这种模型确是图数据流上许多其他操作的基石，同时在现实生活中也有着非常重要的意义和价值。现实生活中图数据流上子图全匹配的模型如下：

（1）在生物化学中，通常使用图结构来建模一种蛋白质网络结构，其中顶点表示蛋白质分子，顶点之间的边用来表示蛋白质分子之间的联系。蛋白质网络结构往往随着某些特定条件的改变而改变（如温度、酸碱性、湿度等），这样使得用来表示蛋白质结构的图由传统的静态图形式转变为随着时间不短更新变化的图数据流形式。这样，为了进一步了解蛋白质结构的结构特点和功能特性，研究人员往往想知道在这种变化过程中该蛋白质网络结构和哪些已知的模式（子图）相匹配，并且确定具体的匹配位置及匹配个数。

（2）在程序设计中，一个包含很多功能的大的程序，程序在不断的进行更新和维护，可以通过子图全匹配来确定这个程序中都包含哪些具有特定功能的小程序，从而了解该程序的功能和结构，并且便于对该程序进行修改和维护。

（3）在社交网络中，图结构用于表示用户之间的交互关系，其中顶点表示不同的用户，边表示用户之间的联系，边上的权重值表示两个用户之间联系的频率。明显地，人与人之间的交互在不断发生变化，边与边之间的频率在不断的随之改变，可以通过子图全匹配技术确定社交网络中存在哪些满足某些边上频率要求的特定的模式，以便了解整个社交网络的结构。

（4）在IP网络中，可以用图来表示IP地址之间的联系，其中顶点表示一个IP地址，顶点之间的边表示两个IP地址之间进行一次联系，边上的权重表示联系的次数。随着用户对计算机的不断使用，不同IP地址之间的联系不断增加，可以将一些经常出现的IP网络攻击方式作为一种模式保存下来，并通过实时的对IP网络与这些已知的IP网络攻击模式进行子图全匹配检测，就可以用于实时检测IP网络攻击。

虽然之前已经存在很多静态图上的关于子图全匹配算法的研究，但是关于图数据流上的这类研究却还没有出现。而这种图数据流上增量子图全匹配的技术如上所述在社交网络，生物医学，网络安全等各种领域都具有广泛的现实意义，因此解决本文所研究内容十分重要，同时也十分迫切。

## 1.3本文研究的内容及面临的挑战

### 1.3.1主要研究内容

图数据流上的子图全匹配问题简单来讲，是指在一个图数据流不断更新的过程中，判断图数据流在不同时间戳上所对应的静态图中有多少个子图与对应的查询图相匹配，并且确定满足条件的每一个子图。

本文主要是针对图数据流上的增量子图全匹配技术进行研究，主要分为两大部分内容，第一部分是图数据流上增量子图相似性全匹配问题的研究，这一部分所针对的图数据流中的更新主要是简单更新，也就是说边只能插入和删除一次，而不能多次重复进行操作。要求在每个时间戳t自动地返回满足要求的匹配结果，这一部分进行的是相似性匹配（准确匹配视为编辑距离为0的特殊相似性匹配）。这一部分中首先对带标签的图，子图同构，图数据流等基本概念进行定义，从而引出图数据流上的增量子图相似性全匹配问题的定义。然后提出了一种最近邻分区的方法对图数据流进行分区，之后通过为查询图创建生成树集合，来进行子图全匹配，并在结果上进行增量的维护。

第二部分是带权图数据流上增量子图近似全匹配问题，这一部分针对的图数据流上的更新是复杂更新，通过权重值的改变允许边可以多次到达和删除。有些时候对于图数据流上的匹配，减小了对匹配准确性的要求，增加了对实时性响应的要求。为了满足进一步这种实时性，这一部分采用一种近似匹配的方法，近似地查找每一个时间戳上满足要求的匹配结果。通过对第一部分中的内容进行改进，使之能够满足带权图的特性，然后通过改变编辑距离的值进行多次匹配，维护不同编辑距离下得到的匹配结果，通过在结果上进行增量的维护来得到最终匹配结果。最后通过使用不同的数据集，对两种问题所提出的方法的有效性和效率进行验证，并对实验结果进行分析，然后评估其性能。

### 1.3.2 面临的挑战

由于图数据流的特性，在解决图数据流上的增量子图全匹配问题上所面临的挑战主要有以下几点：

（1）静态图上的子图同构是一个NP完全问题[1,2,3,4]。传统静态图上的子图全匹配问题虽然已经得到了解决，但是由于图数据流动态更新的特性，使得每一次更新之后就相当于产生一个新的静态数据图，需要重新进行一次子图全匹配的过程，这样已有的静态图上子图全匹配的方法势必会导致时间代价过高，因此不适用于图数据流上。那么如何设计一种子图全匹配方式能够对每次更新之后的匹配结果进行增量的维护是本文面临的挑战之一。

（2）不同于静态图上的分区，图数据流上的每次更新都有可能导致分区的结果发生改变，如果在每次更新后都需要重新进行一次更新必然会使得时间代价过大，那么如何设计一种合理的分区方法能够对图数据流的每一次更新进行增量的维护就是本文面临的挑战之二。

（3）图与图之间的匹配操作是NP完全问题，虽然已经存在一些解决的方法，但是普遍效率较低，因此采用一种怎么的全匹配方式来加快每次匹配的速度就是本文面临的挑战之三。

（4）复杂更新导致图数据流上边的权重在不断发生变化，使得已提出的方法不能直接用在带权图数据流上，那么如何对已有方法进行改进使其能够应用在带权图数据流上是本文面临的挑战之四。

（5）图数据流的一个重要特点就是其在不断的更新变化，而且更新速度往往很快，有些时候想要实时地返回满足要求的匹配结果，那么采用何种方式进行带权图数据流上子图全匹操作配来满足图数据流实时响应的特性就是本文面临的挑战之五。

## 1.4本文主要贡献

通过对已有的关于静态图上的子图全匹配技术和图数据流上相关算法的学习和总结，本文提出了图数据流上增量子图全匹配的问题。根据之前的静态图上子图全匹配计算的基本算法，并结合图数据流的自身特点，为图数据流上增量子图全匹配的问题提出新的适用方法。本文的主要贡献点如下：

（1）对图数据流上的基本概念和相关操作进行了定义，进而对图数据流上增量子图相似性全匹配的问题和带权图数据流上增量子图近似全匹配问题进行了定义；

（2）提出了最近邻分区的方法在图数据流上进行分区，从而实现图数据流上对分区和结果的增量维护；

（3）本文采用为查询图创建生成树集合，并通过为生成树集合中的每一颗生成树在图数据流中寻找符合条件的匹配，从而引入最大相似性匹配的方式进行全匹配，将图与图之间的匹配操作转化为树与图之间的匹配操作，从而大大加快了匹配的速度；

（4）本文提出计算所有相似性最大匹配而不是所有的相似性匹配，从而避免了进一步删除冗余的代价；

（5）本文对提出的方法进行改进，使其能够应用在带权图数据流上，并提出了通过对不同编辑距离得到的匹配结果进行增量维护的方式，增量进行更新，从而解决了带权图数据流上增量子图近似全匹配问题。

## 1.5本文组织结构

本文分为五个章节，具体内容如下：

第一章是引言。主要介绍了本文的研究背景，介绍了本文研究内容现实意义，并在此基础上提出了本文的研究内容。然后介绍了本文的主要研究内容及面临的挑战，最后对本文的主要贡献进行了总结。

第二章是相关工作。首先介绍了现阶段静态图上子图匹配和子图全匹配问题的一些解决方法，并进行比较和总结，指出这些方法无法适用于动态的图数据流上。然后介绍了目前对于图数据流上已有的一些相关研究，分别通过图数据流上的简单更新和复杂更新两个部分进行介绍。

第三章是图数据流上的增量子图相似性全匹配问题，是针对简单更新提出的一种相似性匹配问题。首先对相关的基本概念进行定义，从而引出本章的主要研究内容的定义。然后提出了一种最近邻分区的方法为图数据流进行分区，并能满足图数据流实时更新的特性。接下来通过创建查询图的生成树集合，并为集合中的每一颗生成树在图数据流中查找满足要求的匹配的方式进行子图全匹配，并实现每次更新后对结果进行增量的维护。最后通过实验验证方法的准确性和有效性。

第四章是带权图数据流上增量子图相似性全匹配，是针对复杂更新提出的一种近似匹配问题。先对基本概念进行了定义，并引出了本章所要研究的内容。然后通过对第三章提出方法的改进使其能够应用在带权图数据流上，在基本的最近邻分区上加上权重的判断后提出了基于权重的最近邻分区方法，最后通过对不同编辑距离得到的匹配结果进行增量维护，在每个时间戳近似地得到最终想要的匹配结果，当误差率大于一定的阈值时，重新进行一次匹配，以提高匹配的准确性。这样既保证了图数据流上实时响应的特性，又保证了匹配结果的准确性。

第五章是总结。对本文研究的内容进行总结，并对未来将要研究的工作进行了进一步的展望。

# 相关工作

图数据普遍存在于已有的数据应用之中，包括有线或无线的互联建模(通信)，2D/3D对象(模式识别)，化合物或者蛋白质网络(化学/生物信息)，电路(计算机辅助设计)，松散模式的数据(XML)，社会或信息网络(Web)等[5]。因此，图数据上的相关操作也得到了越来越多学者的重视，并进行了广泛而深入的研究，例如子图匹配问题，最短路径问题，包含查询问题等。其中，子图匹配作为图上的一种重要查询，广泛的应用于模式匹配及信息检索等应用中。本文主要是针对图数据流上的子图相似性匹配问题进行增量的研究，虽然现阶段还没有针对此问题进行的研究，但传统静态图上的子图匹配问题以及图流上的一些其他操作却已得到了深入的研究，本章主要对现在已经存在的静态图上的子图匹配问题的解决方法以及图数据流上的相关研究进行简单的介绍。

## 2.1静态图上子图匹配

静态图上的子图匹配问题根据匹配的方式可以具体分为子图匹配问题和子图全匹配问题，根据匹配结果的准确性可以分为准确匹配和近似匹配两方面，带有编辑距离的匹配称为相似性匹配，其中准确匹配可以视为编辑距离为0的带有相似性的准确匹配。

### 2.1.1静态图上子图匹配

随着数据量的不断增大，图作为一种有效地表示数据之间复杂关系的数据结构，广泛应用于社交网络，生物医学等各种领域当中，因此图上的相关操作得到了广泛并深入的研究，其中子图匹配作为图上的基本操作之一，被广大学者进行了深入的研究。

有很多文章介绍了静态图上的子图查询，Cheng et al[19]，He et al[26]，Cordella et al[38]基于传统的子图同构检测方式使用标签相似性进行查询，这些方法使用剪枝策略进行过滤，减少空间的使用。Khan et al[6]提出了一种用于查询现实网络的基于邻居的子图匹配技术NeMa。为测量匹配质量，提出了一个新的子图匹配代价度量标准，基于这个标准，公式化了最小匹配代价问题。

Wu et al[1]提出的方法与传统的基于子图同构的子图查询不同，它利用了本体信息来扩展了子图查询，从而来确实语义上相关的匹配。本文主要介绍了以下三部分内容：首先提出了基于本体图的子图查询，从而引出了TOP-K子图匹配问题；其次提出了一个包括创建索引、过滤和验证三个阶段的filtering-and-verification 框架，用来解决TOP-K子图匹配问题，并提出了本体索引的概念；然后通过对影响区域的维护来实现本体索引的动态维护。在filtering-and-verification 框架中，每收到一个查询时，在过滤阶段，该框架利用本体索引在多项式时间内，要么提取一个包含所有匹配的数据图的小的子图，要么确定不存在的匹配；在验证阶段，该框架直接从小的子图中提取最好的匹配，而不需要搜索整个数据图。

Shasha et al[10]提出了GraphGrep方法在过滤时作为索引特征来枚举路径。由于路径特征的限制，Yan et al[11]提出了gIndex来索引频繁的子图作为索引特征。由于反单调性，只要一个子图模式不是频繁的，那么任何包含它的超图也不会是频繁的。在查询处理时，使用一个代价模型来评估检索哪个索引入口。但是gIndex需要在每个时间戳挖掘频繁的子图（特征），不满足图数据流实时响应的特性。而GraphGrep虽然可能满足实时响应的特性，但是它是单独使用路径来过滤掉候选，过滤后仍然会存在很多误报的情况，即不是真实的结果却被报成了真实结果。

Chen et al[12]对子图匹配的特征进行了系统的研究，并提出了一个对比的基于子图的索引模型，叫做clndex。使用识别冗余的特征选择过程，clndex 可以解决一系列重要的，不同的对比子图并且最大化其索引功能。Zhao et al[21]发现索引频繁的树模式再加上几个子图模式，子图搜索性能比索引所有子图模式要好。Zou et al[34]提出GCoding用于图搜索。与其他方法相比，它的效率很高。但是，对于流来说，计算特征值特性的代价太高了。

上述算法主要是用于静态图上的子图匹配问题，除了子图匹配，同时还存在一些子图全匹配问题来达到不同的问题研究。

### 2.1.2静态图上子图全匹配

子图全匹配与子图匹配问题不同，他不仅要求确定一个小的查询图是否是一个大的数据图的子图，还进一步要求确定该数据图中包含多少这样的小的查询图并且确定这些满足要求的子图在数据图中的位置。

在静态图上计算满足的所有相似性匹配的一种简单方法是枚举所有比查询图至多少条边的的连通子图，然后为每一个这样的连通子图在查询图中找到它所有的准确匹配。

Zhang et al[3]提出了SAPPER算法，在索引中利用混合社区单元结构，SAPPER利用了预生成的随机生成树和一个细心设计了的枚举图的较优序列，解决了在大的数据图中确定与查询图相似性匹配的所有的子图。SAPPER的子图全匹配分为四个步骤，顶点匹配，创建查询图的随机生成树，产生（表示与查询图满足编辑距离为的，与相似性同构的图的集合）中图的匹配顺序和最终的图匹配。具体地说，首先基于混合社区索引为查询图中的每个顶点确定其在数据图中的候选匹配集合，用来进行顶点匹配。其次随机的产生查询图的生成树集合，并根据第一步的顶点匹配确定这些生成树的匹配，这些生成树的匹配用于进行图的近似匹配。由于有许多图需要进行匹配，就需要确定一个这些图进行匹配的合理的顺序来加快匹配速度。最后确定这些图的所有匹配。利用这种方法可以有效的将时间复杂度由变为，其中是指查询图中边的个数，指编辑距离，其中得到的相似匹配图只能比查询图少边而不能多边[3]。但是SAPPER的性能在增加到3时大幅度下降。

在SAPPER算法的基础上，Zhu et al[2]提出了TreeSpan算法来进一步解决子图相似性全匹配问题。这篇文章中，提出了一种新的模式来执行相似性全匹配。TreeSpan使用以下三个阶段来执行最大相似性全匹配：第一阶段Seeding，生成查询图的最小的生成树集合来覆盖所有的子图，其中要求这些子图与查询图的编辑距离至多为，也就是任何至多少条边的的生成子图使用QT中的至少一颗生成树作为它的生成树。同时为了尽量减少匹配过程中重复生成同一个子树，TreeSpan算法为生成树集合中的每个生成树附加一个不包含的边的集合，该集合既限制了不同的生成树所获得的匹配子图避免了重复性，同时又满足子图匹配的完整性。第二阶段Exact All-Matching：对于每个生成树，得到所有从到数据图的子图同构映射，也就是准确的全匹配。第三阶段Inducing Match：对于每一个子图同构映射，引入符合的从得到的的相似性匹配。为了节省存储空间，TreeSpan算法使用树的结构采用共享前缀的方式对生成树进行存储。

与SAPPER算法相比TreeSpan算法有三点优势：首先，总要比小得多，在两种极端情况下，因此大大的减少了执行准确全匹配的时间，也就是从减少到。TreeSpan算法中按需枚举的策略进一步减少了执行准确匹配的时间。其次，由于生成树结构上比图结构简单得多，使得在生成树上执行准确全匹配的代价远小于在图上执行准确全匹配的代价。最后，TreeSpan算法通过共享计算在生成树集合上执行准确全匹配，而SAPPER算法在中的每个图上分别执行准确全匹配。

然而这些方法均不适用于动态的图数据流上，因为图数据流不断更新的特性，使得每次更新之后都要对整个匹配过程进行一次从头到尾的完正匹配，这势必会导致极高的时间复杂度并且也不能满足实时匹配的要求。

## 2.2带权图上子图匹配

Umeyama et al[35]提出了一种特征分解方法用于解决带权图上的匹配问题。这篇文章讨论的是一个近似的解决方法来解决有向图和无向图上的带权图匹配问题（the weighted graph matching problem ，简称WGMP）。WGMP是指在每个弧都带有权重的图上，找到两个带权图之间的最佳匹配。该文采用了一种分析的而不是组合或迭代的方法来解决这样的最佳匹配问题。在无向图匹配问题中使用邻接矩阵的特征分解方法，并且在有向图匹配问题中使用来源于邻接矩阵的艾尔米特矩阵的情况下，当两个图互相足够接近时，能够有效地 找到最佳匹配。当两个图互相不能足够接近时，文中提出的方法有的时候就不能找到最佳的匹配结果。然而，由于得到的匹配反映了图之间的全局匹配，使得爬山法能够提高在这种情况下获得的匹配。当带权图与同构图区别很大时，文中提出的方法的效果就不能像几乎同构的情况那么好。而且该方法要求待匹配的图大小相同，这是一个重要的限制，而在现实生活中大多数情况下都很难得到满足。

Almohamad et al[36]提出了线性规划的方法来解决带权图的匹配问题。线性规划是通过以L1规范公式化图匹配问题，然后将结果的二次优化问题转化为线性的，其中L1规范具体如下：

 （2.1）

其中表示L1规范，也就是如果是一个矩阵，那么，其中。和分别是有向带权图和的邻接矩阵，是 的排列矩阵。线性规划通过基于单形的算法得到解决，然后通过在线性规划实际解决方案上应用匈牙利法得到近似结果。提出算法的复杂度为，其中n为匹配图的大小。

Gold et al[37]提出了一种毕业分配算法用于带权图上的子图匹配问题，该方法即使在存在高噪音的情况下也可以快速准确地找到匹配结果。通过结合毕业的凹凸性，双向（分配）限制和稀疏性，大大提高了匹配的精度和速度。在有噪声情况下的低位计算复杂性和健壮性使得作者提出的方法优于传统的组合方法。作者提出的算法不局限于任何特殊类型的图，可以应用于子图同构、带权图匹配和属性关系图匹配。

同样，由于图数据流的不断更新，这些方向也都不适用于动态的带权图数据流上，因为每次更新后都需要从头开始重新进行一次完整的匹配过程既会导致昂贵的时间代价。

## 2.3图数据流

随着科学和技术的不断发展，图数据不在只是单纯的静态形式，而是以一种流的形式不断更新，从而形成了一种新的结构——图数据流。图数据流除了拥有图本身的复杂结构和性质外，还增加了自己特有的性质，其中最为关键的就是它是不断动态更新的，并且边的更新速度通常很快。图数据流自身的特性的存在使得传统的用于静态图上的一些操作方法不再适用，因此广大学者开始对图流上的操作进行了研究。图数据流的更新根据更新方式的不同可以分为简单更新和复杂更新两中类型，其中简单更新是指图数据流上的边只能有存在和不存在两种情况，也就是说如果一条边已经存在，这条边就不能再一次被插入，即同一条边不能到达或删除多次；复杂更新则是指图数据流上的同一条边可以多次到达或多次删除，通常用权重进行衡量，表示边出现的频率。目前大多数图数据流上的研究都是针对简单更新进行的。

### 2.3.1图数据流上的简单更新

Wang et al[4]解决了图流上近似子图匹配的问题，也就是说给定一组图流<G1，G2，……，Gm>和一组查询图模式<Q1，Q2，……，Qn>，在每个时间戳t连续报告所有可能的连接对<G(i,t)，Qj>，其中Qj是G(i,t)的子图，。首先使用一种结构节点邻居树（Node-Neighbor-Tree，简称NNT）来捕获每个节点周围的局部结构。NNT是指给定一个图G和一个深度值L，对于任何u∈G，u的NNT记做NNT(u)，是一颗以u为根节点的树，并包含G中所有的从u出发，长度小于等于L的简单路径。其中，简单路径指没有重复边的路径。为了满足图数据流动态更新的特性，作者通过创建节点树索引集和边树索引集来记录节点邻居树中每个节点以及每条边的位置，从而实现节点邻居树的动态维护。在为数据图和查询图中的每个节点建立完节点邻居树之后，由于子树同构代价很大，所以将这个过程近似为比较两个节点邻居树的简单路径。为了加速这个比较过程，作者提出了一个映射模式将一棵节点邻居树映射为一个数字的向量，称为节点映射向量（node-projected-vector，简称NPV）。一个节点u的映射向量NPV(u)是存储一个NNT(u)中每一维出现数量的向量。为了满足图数据流动态更新的特性，将每个节点的NPV映射到向量空间中，通过支配集覆盖的方法在向量空间进行搜索。用这种方法，在某一时间戳上图数据流进行更新时，G中相应节点的节点映射向量在空间向量中的位置改变，对于每一维，只需要更新数据图覆盖向量的个数即可。此外还提出了一种skyline提前终止法，与支配集覆盖法检测图数据流是否覆盖查询图的所有向量不同，这种方法是搜索不被图数据流的任意向量统治的查询向量，也就是说想要在查询图中为图数据流的向量集找到skyline。为了进一步加快查询速度，作者分别从以下三个方面进行优化处理。首先，离线计算查询图向量集的Monochromatic Skyline点，并且只检测这些点是否只是流向量集的Dichromatic Skyline点；其次，调整查询点被处理的顺序，为了最大化到达一个提前终止点的可能，作者首先根据一些简单的条件排序查询点，使得能首先检测成为Skyline点概率高的那些点；最后，不需要枚举流中所有向量，而是只在查询向量不为0的维内，执行子空间搜索。

Angel et al[8]提出了用于故事识别的流边权重更新下的密集子图维护问题，用以高效地确定实体图中的密集结构。本文以实时确定新兴的故事为动机，为了广泛地测量实体联系，公式化了在大小约束下的，对于边权重更新流的密集子图维护问题；基于由一个单个边权重的更新引起的最大可能改变的量化，提出了一个有效的算法DYNDENS，通过维护少数稀疏子图，DYNDENS能够有效地，并增量地计算密集子图；同时设计了一个有效的密集子图索引——在前缀树中嵌入反转列表，可以减少内存消耗和处理工作；为DYNDENS提出了两种理论上的启发式方法MAXEXPLORE和DEGREEPRIORITIZE，可以提供进一步改进的性能，这两种方法都与限制exploration和cheap exploration的执行次数相关。MAXEXPLORE改进了探索迭代的界限，将图中更新边的邻居和被探索子图基数考虑在内；DEGREEPRIORITIZE是一种组织搜索空间的方法，因此可以避免执行冗余搜索。

Sarma et al[15]着重于大图（如web图）上的计算，其中这个大图的边是以流的形式出现的。这篇文章主要目标是在图流上评估PageRank时减小内存的使用，以及执行较少的遍数。分别提出了SingleRandomWalk算法，MultipleRandomWalk算法和ModifiedMultipleRandomWalk算法来实现目标。Bahmani et al[14]提出了一种在流模型中用于寻找密集子图的一种新算法，并应用在了被越来越多的大规模数据处理应用程序所采用的两种计算模型流和MapReduce中。本文展示了一个简单的算法，使得经过图的次数减少，并且获得了一个（2+є）近似的密集图。然后，分别在子图被规定为一个特定的大小和图是有向图这两种情况下降算法进行扩展。

Yu et al[16]提出的方法的主要思想是公式化SimRank的评估之后，将矩阵与矩阵之间的乘积转化为向量与向量的乘积和向量与矩阵的乘积，从而加快计算速度。具体地讲，首先通过秩为1的西尔维斯特矩阵方程描述SimRank的更新矩阵ΔS，并设计了一个快速增量算法在O (Kn2)时间内来计算n2个节点对的相似性。然后提出了一个剪枝的技术来捕获ΔS的影响区域，减去非影响区域来减少不必要的计算。可以进一步加快计算速度这这些，其中影响区域的大小小于节点对数。现实生活中，影响区域的大小远远小于节点对数。

### 2.3.2图数据流上的复杂更新

Cormode et al[23]提出了一个用于总结数据流的新的亚线性空间数据结构，Count-Min略图。Count-Min略图由一个二维数组组成，宽w和深度d由用户指定的参数决定。该数据结构选择了d个成对的相互独立的哈希函数h1,h2,……,hd，每个hi统一地映射到一个在[0,w-1]之间的随机的整数，并对应到这个哈希函数对应的一维数组中的d个格中的一个去。然而，由于图流的特性，这种方法不适应于动态的图流，再次方法基础上，Zhao et al[13]将Count-Min略图和略图分区方法相结合，提出了一种新的略图结构，gSketch。主要对两种基本的查询，边查询和聚合子图的查询进行评估。根据图流全局异构和局部相似的特点提出了在全局略图上进行分区。分区是通过平分内存空间和以使总体相对误差最小的点作为分区节点的方式进行的，这样就可以使有相似频率的边在同一个分区中，从而提高查询评估的准确性。作者分析了只能获得数据样本和数据样本及工作负载均可获得两种不同的情况，并在两种情况下分别公式化了总体相对误差，用来确定分区时的区分节点。

从上面的介绍可以看出，虽然目前已经有很多关于图数据流上相关操作的研究，但是还没有研究集中在大的图数据流上的增量子图匹配技术的研究上，因此本文将研究主要集中在这个问题上，并通过对上述方法的学习和深入研究找到解决本文研究内容的途径。

## 2.4本章小结

本章主要对现阶段图以及图数据流上的一些相关理论和研究现状进行了简单的介绍。首先介绍了静态图上子图匹配和子图全匹配的一些方法，接下来介绍了对于图数据流上的已有的一些相关操作的研究，最后根据静态图上的全匹配和数据流图上的相关操作相结合提出了本文要研究的问题，即图数据流上增量子图全匹配问题。

# 图数据流上增量子图相似性全匹配

随着科学技术的快速发展，数据的种类已经变得越来越多，同时数据量也变得越来越大，而且数据的结构也同样变得越来越复杂，因此通过图数据结构对数据进行建模，并通过分析图数据的结构特征来得到有效的信息的方式已经受到了广泛的关注。尤其是近几年来对于图数据中子图全匹配及相关的各种变体操作也已经得到了广大学者的深入且广泛的研究，并应用到各种不同领域当中。然而，已有的图数据上子图全匹配问题的研究，主要是针对静态图上进行的，而现在的实际应用中，图数据不再拘泥与传统的静态形式，而是以一种流的方式不断进行更新变化。由此，引出了本章所要研究的主要问题，即图数据流上增量子图全匹配问题。

本章主要研究的内容是图数据流上增量子图全匹配问题，即在图数据流不断更新变化的过程中，自动地每革一段时间t返回在不同时间上图数据流所对应的静态图与已有查询图相匹配的满足编辑距离阈值的每一个子图。本章中首先提出了最近邻分区的方法进行分区，然后通过查询图创建最小生成树集的方法，进行子图全匹配，使得每次更新后不需要重新进行一次完整的全匹配操作，而是可以在原有的结果基础上进行一种增量的维护，尽可能减少每次更新后的各种维护代价，从而大大提高匹配的速度。

## 3.1问题定义

在对本章所研究问题进行具体阐述之前，首先通过本节介绍一些与本章研究内容相关的基本定义，如顶点带标签的图，图数据流的改变操作，图数据流等。接下来通过这些定义的基本概念正式提出本文所研究的问题，即图数据流上增量子图全匹配问题。

### 3.1.1基本概念定义

**定义3.1**（顶点带标签的图）：一个顶点带标签的图是一个四元组，，其中是的顶点集合，是的边集合，是的顶点的标签集合，标签函数定义了映射.



图3.1顶点带标签的图

Fig.3.1 Vertex labeled graph G

**定义3.2**（子图）：给定两个顶点带标签的图和，如果图的顶点集合和边集合分别为图的顶点集合和边集合的子集，（即，且），并且和有相同的标签函数，（即），则图为图的子图。

**定义3.3**（子图同构）：给定两个顶点带标签的图和，子图同构是存在一个单射函数，使得：

（1）；（2）；此时就认为图和图是子图同构的，也可以说包含或者说中包含的准确匹配。

**定义 3.4**（生成子图）：给定一个图，如果是连通的且中的所有顶点都包含于当中，则说子图生成，也可称作是的生成子图。

同理，可以按照生成子图的定义方式推广到一个图的生成子树的定义，也就是说任何的生成树生成。

**定义3.5**：假设是的生成子图，且，由引入的在中的相似性匹配为，其中。

**定义3.6**（图改变操作）：给定一个图，对于，边的插入和删除可以通过一个三元组表示，其中表示边的插入或删除操作。由此得到图改变操作的正式定义如下：给定一组图的操作，其中。可以通过转变为，记为。

注意：本章中所研究的图数据流的更新针对的是简单更新，即图数据流上的边只有存在和不存在两种可能，也就是说同一条边不能被插入和删除多次。

**定义3.7**（图改变操作流）：给定一个初始图，图的改变操作流是一组图改变操作的序列，其中t代表时间戳，记录改变操作发生的时间。。

如下图3.2所示是一个图改变操作流的实例，即执行

操作。具体地说，当第一组图改变操作到达时，在初始图上执行相应的更新操作，从而得到图，类似地可以通过更新操作和分别得到和。可以看到当执行操作时，由于边的删除导致一些顶点成为孤立顶点，如图中的。此时我们并不将孤立顶点删除，以便之后继续进行边的更新操作。对于孤立顶点具体的处理方法，将在本章后续内容中详细介绍。



图3.2图改变操作流实例

Fig.3.2 Illustration of a graph change operation stream

**定义3.8**（图数据流）给定一个初始图和一个图改变操作流，图数据流表示一组随着时间戳连续变化的图的集合[4]。定义如下：



如图3.2所示亦是一个图数据流，其中是初始图，从初始图经过最终转变成的这个过程即为一个图数据流。图改变操作流。图数据流，其中，且。

### 3.1.2图数据流上增量子图相似性全匹配问题的定义

本章主要讨论图数据流上增量子图全匹配问题，将通过准确子图全匹配和相似性子图全匹配两个方面来具体阐述问题的研究。然而一般来说准确子图全匹配可以看作是相似性子图全匹配的一种特殊形式，即阈值为0的情况，因此本章内容以介绍图数据流上相似性增量子图全匹配问题为主。

本章假设在查询开始之前存在一个初始的图数据流和一个查询图，然后随着时间推移，图数据流的改变操作从时间戳开始依次到来。本研究的期望结果是在每个时间戳，可以自动返回该时间戳下对应的静态数据图中有哪几个子图和已有的查询图相匹配。问题的正式定义如下：

**问题3.1**（图数据流上准确全匹配）：给定一个图数据流和一个查询图，从时间戳为开始，在每个时间戳上，想要报告该时间戳下的所有子图，其中满足在时刻下与查询图是子图同构的。

**定义3.9**（编辑距离）给定两个顶点带标签的图和，从图转化为图时需要增加的边的个数即为图到图的编辑距离，记为。

也就是说，在本文中对于计算编辑距离时只考虑少边的情况，而不考虑多边的情况。目前大部分与匹配相关的文章中都是只考虑少边的情况，而不对多边的情况进行考虑，如[2]，[3]。

**问题3.2**（图数据流上相似性子图全匹配）：给定一个图数据流，一个查询图和一个编辑距离阈值，从时间戳为开始，在每个时间戳上，我们想要连续报告该时间戳下的所有子图，其中是在时刻下与查询图是满足阈值为的子图同构关系。

如图3.3所示。初始时给定一个图数据流和查询图，查询图3.3(a)所示。在每个时间戳上，将查询图和该时间戳下对应的静态图进行子图全匹配操作，如果找到满足阈值（准确匹配时=0）的子图，则返回该时间戳下对应的子图。具体地说，在时刻，对应的静态图如图3.3(b)所示。在该时间戳下，与查询图进行准确子图匹配（=0）的结果如图3.3(c)所示，即映射函数：，此时返回时间戳和图(c)。与查询图满足=1的相似性全匹配结果如图3.3(c)-(h)所示。(c)-(g)是通过映射函数：得到的，(f)是通过映射函数：得到的。其中图(d)-(g)均是比准确匹配的结果图(c)少一条边的子图，这些子图是我们不想考虑的。因此，在本文中只考虑满足编辑距离阈值的最大子图全匹配。对于最大子图全匹配问题的定义如下所示。

图3.3图数据流上子图全匹配问题

Fig.3.3 The problem of subgraph all match over graph stream

**定义3.10**（最大匹配）查询图在数据图中的一个匹配是最大的当且仅当不存在另一个的子图使得是的一个适当的超图（即是的超图，且），并且和有相同的映射函数。

**问题3.3**（图数据流上最大相似性子图全匹配）：给定一个图数据流，一个查询图和一个编辑距离阈值，从时间戳为开始，在每个时间戳上，我们想要报告该时间戳下的所有子图，其中是在时刻下中与查询图满足阈值为的不同的最大相似性匹配。

通过上述定义可知，在时刻下，虽然与查询图满足=1的相似性全匹配结果为(c)-(f)，然而图(d)-(g)均为比准确匹配图(c)少一条边的子图，不满足最大匹配的概念，此时只有图(c)和图(f)满足要求，因此在时刻返回的与查询图满足=1的相似性全匹配结果为时间戳和图(c)，以及图(f)。

值得注意的是，满足阈值的相似性匹配可能是一个准确匹配。通常找到所有满足阈值的相似性匹配比准确全匹配要困难，因为准确全匹配是相似性全匹配的一种特殊情况，即。所以本章在主要研究图数据流上满足阈值的相似性全匹配问题。在数据图中找到查询图的一个准确全匹配就是NP-Completeness问题。[31]因此本章所研究的图数据流上满足阈值的最大相似性子图全匹配问题也是NP-Completeness问题。

本文提出的解决图数据流上的增量子图全匹配问题的直接方法如下：

给定一个初始的图数据流，其中图数据流随着时间不断更新变化即，一个查询图，一个时间间隔T和一个编辑距离阈值。从时刻开始，在每个时间戳上，自动地返回该时刻下图数据流所对应的静态图中包含的所有满足与查询图满足编辑距离阈值的最大匹配子图的步骤如下：

（1）初始的图数据流是已知的，所以在离线的情况下首先使用最近邻分区的方法为图数据流进行分区；

（2）生成查询图中的一个最小生成树集合，用来覆盖所有比至多少条边的连通子图。也就是说，对于每一个比至多少条边的连通子图，使用中的一颗生成树作为它的生成树；

（3）在图数据流中，为中的每个生成树查找所有准确匹配，用来引入最终的所有相似性匹配。

本节首先介绍了图上的一些基本操作如顶点带标签的图、子图同构、图改变操作、图数据流等，然后通过这些基本概念的定义，正式引出了本章主要研究问题图数据流上准确子图全匹配和图数据流上近似子图全匹配问题的定义。然后提出了一种最近邻分区的方法为图数据流进行分区，并能满足图数据流实时更新的特性。接下来通过创建查询图的生成树集合，并为集合中的每一颗生成树在图数据流中查找满足要求的匹配的方式进行子图全匹配，并实现每次更新后对结果进行增量的维护。最后通过实验验证的方法的准确性和有效性。

## 3.2最近邻分区

在本章中假设初始的图数据流和查询图都是已知的。从时刻开始图数据流进行更新变化，并在变化过程中对查询图进行全匹配操作，在每个时间戳上自动地返回相应时间戳上的匹配结果。通过对匹配结果的特征的分析，我们得到的匹配结果需要满足以下条件：（1）作为结果的匹配子图必须是连通的，（2）作为结果的匹配子图的顶点个数必须等于查询图中顶点的个数，（3）作为结果的匹配子图的节点标签集合必须等于查询图的节点标签集合，即，（4）作为结果的匹配子图边的个数不多于查询图中边个数。

图数据流最大的特点就是边和顶点不断的进入，使得整个图数据流在不断的进行更新。如果使用原有的静态图上的方法，就需要在每一次更新时都对整个图数据流进行一次完整的匹配操作，这样势必会导致时间复杂度过高，并且无法满足图数据流实时返回匹配结果的要求。由观察我们可以知道，图数据流上的更新往往遵循局部性，这样我们就考虑是否可以将这个初始的图数据流提前根据查询图进行分区，当有新的边或者顶点到达进行更新时，我们先判断到达的更新属于哪个分区，然后对于没有更新的分区结果不需要改变，而只需要对更新了的分区的结果进行维护即可，这样就满足了本文增量更新的思想。由于图数据流不断更新的特性，可能导致每一次更新之后使得原有的分区结果会发生相应的变化，那么静态图上的分区方法就不能适用在图数据流上。因此，本文基于相邻越远的两个顶点之后被连接上的可能性越小，相邻越近的两个顶点之后被断开的可能性也最小这种假设，提出了一种分区方法，即最近邻分区。 这种分区方法使得被分开的各个区之间的联系尽可能地达到最小，也就是说尽可能地使在之后的更新中插入的边是跨两个不同分区的边的可能性达到最小，从而使得随着图数据流的更新对分区进行维护的代价尽可能地达到最小。

最近邻分区的步骤如下：

（1）结构剪枝：将图数据流中不包含于查询图的顶点标签集合的顶点和与该顶点相连的边删除；

（2）最近邻判断：根据查询图判断数据图中对于顶点的最近邻，并适当增加一些虚边，以尽量减少之后合并两个分区的可能性；

（3）分区：通过连通性来判断分区。

### 3.2.1结构剪枝

在进行子图全匹配时，图数据流往往是包含大量不同种类标签的大规模图型，而查询图往往是包含标签种类较少的小规模图形。这样通常会导致这样一种现象，即数据图中包含的一部分顶点标签的种类是查询图中所没有的，而这种顶点和与这种顶点相连接的边一定不会成为最后与查询图相匹配的结果中的一部分。因此在进行具体分区之前先对初始的图数据流进行结构剪枝，剪掉符合上述条件的顶点和边，具体的操作过程如下：

首先我们初始化集合，它们分别表示通过结构剪枝后所得图的顶点集合，边集合，顶点标签集合以及标签函数，其中。同时初始化集合，用于记录还未被遍历的顶点。然后，对于大的图数据流，我们利用广度优先遍历的方法对整个图进行遍历。初始时，从图数据流的顶点集合中随机选取一个顶点作为遍历的初始顶点，判断顶点对应的标签是否包含于查询图的顶点标签集合中，若不包含，则在集合中删除该顶点，随机选择另外一个顶点进行判断；若包含则将加入到集合中，并将加入到集合中，同时在集合中删除顶点。此时，，，。如果的邻居顶点能够与查询图中的某个顶点的标签相匹配，即的标签包含于查询图的顶点标签集当中，那么更新集合，，，和，即将添加到集合中，将加入到中，并将从中删除。同时将顶点和之间的边加入到集合中。此时，，，。如果的邻居节点不能够与查询图中的某个顶点相匹配，即的标签不包含于查询图的顶点标签集当中，那么在集合中删除顶点，然后处理的下一个相邻节点；重复上面操作，直至没有顶点能够加入到集合中。此时，在未匹配的节点集合中再随机选择一个顶点作为遍历的初始顶点，重复执行上述操作，直到集合为空，即图数据流中的所有顶点都被遍历完为止。最后得到一个新的图。这个新的图中所有顶点的标签一定都包含于查询图的标签集合当中。



图3.4图数据流上的结构剪枝实例

Fig.3.4 Illustration of a structural pruning over graph streaming

例如对于图3.4中的图数据流和查询图，可知查询图的顶点标签集合。首先初始化集合和未匹配顶点集合，其中，。然后对于大的图数据流进行广度优先遍历。随机选择顶点作为遍历的初始顶点，可知顶点的标签包含于查询图的顶点标签集合中，因此将加入到集合中，并将加入到集合中，同时在集合中删除顶点。此时，，，。对于的邻居节点，它的标签不包含于的顶点标签集合中，直接将从未匹配顶点集合中删除，此时。接下来处理的下一个邻居节点，由于、、的标签都包含于查询图的顶点标签集合中，且均未被处理过，将三个顶点都加入到中，更新各个集合，此时，，，；然后从新插入的顶点开始，继续进行广度优先遍历，由于的邻居节点的标签不包含于查询图的顶点标签集合中，故顶点不加入到中，因此次遍历后得到的结果为：，，，；由于此时没有满足条件的顶点可以插入到集合中，因此在未匹配顶点集中再随机选择一个顶点，继续重复执行上述操作，直至集合为空时终止。最后，得到的最终结果如图3.4(c)所示。此时的图数据流是一个由三个部分组成的非连通图。

表3.1 图的结构剪枝算法

Table 3.1 Graph Structural Pruning Algorithm

|  |
| --- |
| **算法3.1** |
| **输入：**：查询图  ：初始的数据流图  ：查询图的顶点标签集  **输出：**结构剪枝后的图   1. 初始化，其中; 2. 初始化未匹配顶点集 ; 3. 初始化一个队列 ; 4. If( is not null) 5. 随机选择一个顶点; 6. Push() into ,; 7. If() 8. ,; 9. While( is not null) 10. ; 11. For each neighbor of ，并且; 12. ; 13. If() 14. Push() into ; 15. ,; 16. ; 17. End If 18. End For 19. End While 20. End If 21. End If 22. Return |

### 3.2.2最近邻判断

将初始的图数据流进行结构剪枝之后，图数据流被分为几个不相连的部分，对于静态图来说，此时各个部分就可以作为几个不同的分区，并在此基础上进行其他操作。然而，由于图数据流动态更新的特性，对于这几个不相连的部分，在之后的更新中很有可能到达一条跨越两个不同的部分的边，这样就会使得每次更新后对分区的维护代价很大。

通常我们认为随着图数据流的动态更新，两个距离越远的顶点，之后被连接上的可能性越小，反之两个距离越近的顶点，之后被连接上的可能性越大；两个距离越近的顶点，之后被断开的可能性越小，反之两个距离越远的顶点，之后被断开的可能性也越大。

为了记录查询图中不同顶点标签的最近邻，我们定义表示标签为的顶点与标签为的顶点距离最短的距离为。我们使用一种双数据链表存储。具体说明如下图：



图3.5双数据链表结构

Fig.3.5 Double data linked list structure

对于头结点，data1用于存储所要查询的标签，每个链表的头结点所对应的标签均不相同，可以作为唯一的标识进行索引。data2用于存储查询图中符合与data1所存标签距离为的顶点标签种类的个数。Next项用于存储指向下一个位置的指针。对于非头结点，data1用于存储在查询图中与头结点的data1所存标签距离为的顶点标签，data2用于存储两标签节点的具体距离。Next项用于存储指向下一个位置的指针。对于尾节点，next标记为NULL，表示后面没有符合条件的节点存在。在本文中。只考虑查询图中距离为1的顶点。在本文中使用几组标签不同的双数据链表结构组成NN索引链表，用来检索查询图中不同标签顶点距离为1的顶点标签。



图3.6NN索引链表

Fig.3.6 NN Index linked list

例如查询图q如图3.4(b)中所示，通过索引结构可以得到查询图中各标签的最近邻集合NN，且，，，。因为，所以，同理，，。然后按照如上述所示方式进行存储，结果如图3.6所示。

基于随着图数据流的更新，两个距离越远的顶点，之后被连接上的可能性越小，反之两个距离越近的顶点，之后被连接上的可能性越大；而两个距离越近的顶点，之后被断开的可能性越小，反之两个距离越远的顶点，之后被断开的可能性也越大这种观点，本文的主要思想是在确定分区前先对每个顶点根据查询图的结构进行最近邻判断，通过增加一些辅助边，使一些以后被连接可能性大的顶点尽量在一个分区当中，从而减少每次更新后对图分区进行维护的代价。具体地说，在本文中要求根据查询图的结构判断后，距离为1的两个顶点，必须在同一个分区内，对于距离大于等于2小于等于3的两个顶点，为两条顶点之间加一条辅助边，使得满足这种距离要求的两个顶点尽量在同一个分区中。当选取的范围较大时会使得大部分顶点都不能被有效地分配到不同的区域当中，从而使得分区效果较差，所以本文中使用两顶点之间距离大于等于2小于等于3作为界线进行判断。具体操作步骤如下：

首先通过索引结构得到查询图中每个顶点对应的索引集和查询图中各标签的最近邻集合。初始化集合，已访问过的顶点集合和辅助边集合（auxiliary edge set），其中，，。然后根据最近邻集合对图数据流进行遍历。从查询图中选取匹配数量最少的顶点作为一个初始顶点，选择中的第一个与顶点相匹配的顶点开始遍历，根据查找与标签中标签相匹配且与距离最近的顶点，将加入到集合中，若与距离是1，则将边插入到边集合中，若距离大于等于2小于等于3则将边插入到边集合和中。接下来依次处理其他顶点，当没有其他满足条件的顶点可以处理时，将查询图中的各顶点对应的索引集中所包含的所有中的顶点都删除。若此时查询图各顶点对应的索引集均为空则遍历结束，否则在不为空的集合中随机找到一个初始顶点按上述过程继续遍历，直到查询图各顶点对应的索引集均为空时结束。



图3.7最近邻判断

Fig.3.7 Judgment of nearest neighbor

例如，首先通过索引结构可以得到查询图中每个顶点对应的索引集，， 。同时通过索引结构也可以得到查询图中各标签的最近邻集合如图3.6所示，即，， ，。初始化集合，已访问过的顶点集合和辅助边集合，其中，，。然后根据最近邻集合对图数据流进行遍历。从中选择与相匹配的顶点开始遍历，的顶点标签，在查询图中的邻居为，图数据流中与距离最近的标签为的顶点为和，因此将边和边放入集合中，且将顶点加入到已访问过的顶点集合中，此时，，。接下来访问顶点，，在查询图中的邻居为和，与距离最近标签为的顶点只有，由于已经被访问过，所以不再重复被处理，与距离最近标签为的顶点是，所以将边加入到集合中，同时加入到集合中，即，。然后访问顶点，，同样，在查询图中的邻居为和，与距离最近标签为的顶点也只有，所以同样不再重复被处理，与距离最近标签为的顶点是，距离为2，此时在和之间增加一条辅助边，并将分别加入到集合和中，以上述方式继续进行处理，当处理到时我们发现没有符合条件的顶点可以继续被处理，此时边集合，，。将查询图中的各顶点对应的索引集中所包含的所有中的顶点都删除，则此时，，。继续选择作为初始遍历顶点按上述方法继续遍历，直至所有顶点均被访问过，即均为空时截止。此时图数据流变为如图3.7所示，其中虚线即为插入的辅助边，如图边,和。具体算法如下：

表3.2 最近邻判断算法

Table 3.2 Nearest Neighbor Judgement Algorithm

|  |
| --- |
| **算法3.2** |
| **输入：**：初始的图数据流  ；结构剪枝后的图数据流  ：查询图中每个顶点的当前索引集  ：查询图中每个顶点的最近邻集  **输出：**   1. 初始化，其中   ;   1. 初始化辅助边集合; 2. 初始化已访问顶点集; 3. 初始化一个队列; 4. While(只要中有一项不为空) 5. 选择数量最少的一个作为起始，选择中的第一个顶点； 6. Push() into ; 7. While( is not null) 8. ; 9. If() 10. ; 11. 找到标签为的列表入口，并且让   ,…,,其中    For each   1. 从中找到它的最近邻节点   使得;   1. If(&&) 2. Push() into ; 3. If() 4. ; 5. If() 6. ; 7. ; 8. End If 9. End For 10. End If 11. End While 12. ; 13. End While 14. Return |

### 3.2.3分区

如上所述，初始的图数据流去除掉与查询图不相关的顶点和边后变为图，然后再通过最近邻判断增加适当的辅助边后成为一个由几个相互独立的部分组成的非连通图。在这之后我们通过判断此时图数据流中子图的连通性来对图数据流进行分区，最后，把每个分区中所包含的第二步插入的虚边删除，将所得结果作为最后的分区结果。值得注意地，将图数据流按上述方法进行分区后所得的分区可能不是连通图，因为删除虚边之后很有可能将原本的连通图断开成为一个非连通图。为了便于之后的匹配操作，我们不仅分别保存每个分区的顶点集合和边集合，对于每个分区中的各个连通图的顶点和边也同时要分别进行存储。也就是说，最终分区结果，其中，表示分区中的第个连通图。

例如根据图3.7中最近邻判断后所得结果通过子图连通性判断可以得到两个连通的子图，再根据辅助边集合将插入的边删除，由于插入的辅助边是真实不存在的，只是用于最近邻判断时将两个可能分开的部分连接在一起，所以在进行匹配之前必须将辅助边删除。由此可以得到最终的分区结果如图3.8所示。值得注意的是第二步最近邻判断时插入的辅助边在第三步分区时会删除。我们最终将初始的图数据流，分解得到最终的分区结果其中， 。

图3.8最终分区结果

Fig.3.8 The final result of partition

### 3.2.4动态维护

图数据流与传统静态图之间最大的区别就在于图数据流不断更新的特性，即边和顶点不断的进入。而图数据流不断进入的边很有可能是一条跨越不同分区的边，这样就使得原有的分区结果发生变化，从而需要重新进行一次分区，这样势必会导致时间复杂度过高，并且无法满足图数据流实时返回匹配结果的要求。本文中为了解决这个问题，提出了最近邻分区方法，通过最近邻判断将之后很有可能被连接上的顶点放入同一个分区当中，从而减少每次更新是对图数据流分区进行维护的代价。对于图数据流上的更新主要分为边的插入和删除，注意图中顶点的插入和删除均可以视为边的插入和删除，其中边的插入操作又具体可以分为以下几种形式：

（1）插入边中有一个顶点的标签不包含于查询图的标签集合中；

（2）插入边的两个顶点的标签均包含于查询图的标签集合中，但和中只有一个顶点存在；

（3）插入边的两个顶点的标签均包含于查询图的标签集合中，且和两个顶点均已存在，并属于同一分区当中，但属于不同的连通图；

（4）插入边的两个顶点的标签均包含于查询图的标签集合中，且和两个顶点均已存在，并属于同一分区当中，同时两个顶点属于同一个连通图；

（5）插入边的两个顶点的标签均包含于查询图的标签集合中，且和两个顶点均已存在，但属于不同的分区；

第一种情形中对应的边，不需要插入到图数据流中，因为本文所研究的匹配要求顶点标签必须相同，对于查询图的标签集合中没有的标签，其顶点和对应的边一定不会成为最终的匹配结果；对于第二种情况本文先找到存在的那个顶点具体所在的位置，然后将边直接进行插入，同时将顶点插入到对应的顶点集合当中；第三种情况中本文将两个顶点所属的同一分区的两个不同连通图进行合并，即合并其顶点集合，边集合以及顶点标签集合。然后将边插入到合并之后的边集合当中；第四种情况的插入更新是一种最简单的情形，直接将边插入到该边所对应图的边集合中即可；最后一种情况是最为复杂的一种情况，本文使用的最近邻分区方法使这种情况的边尽可能达到最少，以减少维护分区的代价，对于这种边本文按如下方式处理：先将两个分区合并为同一个分区，然后再根据顶点所在位置按上述情形（4）中所述方法进行操作。

为了加快对图数据流上分区的维护，在插入边之前，本文先对边进行过滤，然后再按上述方式执行操作。按照本文中对编辑距离的定义，我们只考虑比查询图少边的情况，目前大部分与匹配相关的文章中都只考虑这种情况，如[2]，[3]。由此可知，如果在查询图中，与插入边的两个顶点所对应的节点标签和所对应的顶点之间的距离大于1，那么这条边一定不会成为最后的匹配结果中的一部分，因此满足这种条件的边本文不进行插入操作，而是直接将其过滤掉。

综上所述，插入边时图数据流分区更新的具体算法如下：

表3.3 动态维护分区——边的插入

Table 3.3 Dynamic maintenance of partition-Insert edge

|  |
| --- |
| **算法3.3** |
| **输入：**：图数据流的分区  ：查询图  ：插入的边  **输出：**更新后的分区   1. If(&&) 2. If(in ) 3. If(&&) 4. 找到所在图，然后执行; 5. Else if(&&) 6. 找到所在图，然后执行; 7. Else 8. If(&&) 9. If(&&,where ) 10. ; 11. Else 12. ; 13. Else 14. ; 15. 找到 和 的位置，并插入; 16. End If 17. End If 18. End If 19. End If 20. Return P; |

对于图数据流的更新操作，除了边的插入外还包括边的删除，在本文中与对结点的删除均视为删除边的删除，即删除与该顶点相连的所有边。同样，对于删除边，存在以下几种情况：

1. 删除边不存在；
2. 删除边存在，删除边后，没有产生孤立顶点；
3. 删除边存在，且删除边后产生了孤立顶点；

对于第一种情况，删除边不存在主要是由于之前的结构剪枝或最近邻判断时将想要删除的这条边已经过滤掉，此时不需要执行任何操作；第二种情况中只需直接确定待删除的边所在具体位置，将边从对应图的边集合中删除即可；对于最后一种情况确定待删除的边所在具体位置并将边从对应图的边集合中删除之后，再将该顶点从对应图的顶点集合中删除。删除边时图数据流更新的具体算法如下：

表3.4 动态维护分区——边的删除

Table 3.4 Dynamic maintenance of partition-Delete edge

|  |
| --- |
| **算法3.4** |
| **输入：**：图数据流的分区  ：查询图  ：删除的边  **输出：**更新后的分区   1. If() 2. 找到边所在图，即 ; 3. If(如果不存在顶点,使得 ) 4. ; 5. ; 6. If(如果不存在顶点, 使得) 7. ; 8. ; 9. Else 10. ; 11. End If 12. End If |
| 13. Return P |

## 3.3生成树集合

由于图数据流是不断更新变化的，若在图数据流上进行处理，则每次更新后都要进行相关的维护，进而增大了子图全匹配过程的代价，因此本文选择为查询图离线创建最小生成树集合，然后再得到最小生成树集合中的每一颗生成树到图数据流的准确全匹配，最后通过这些准确全匹配得到满足编辑距离阈值所有最大相似性匹配子图。本小节主要介绍如何为查询图在离线的情况下创建其最小生成树集合。

### 3.3.1为查询图创建生成树集合

本文通过先为查询图离线创建最小生成树集合，然后通过查询图的最小生成树集合进而得到与查询图满足编辑距离阈值的所有最大相似性匹配子图。但是，很有可能出现这样一种情况，即不同的生成树很可能产生相同的相似性最大匹配结果。例如图3.9所示。数据图和查询图分别如图(a)，(b)，查询图的两个生成树如图(c)和(d)，通过这两个生成树可以在中得到同一个满足的最大相似性匹配，如(e)所示。事实上在这个例子当中，查询图的所有不包含边的生成树都将会得到相同匹配子图。重复的匹配结果不仅在匹配过程中会增加匹配代价，同时还需要额外的操作将其移除，因此需要一种方法避免重复匹配结果的生成。



图3.9重复结果

Fig.3.9 the repeat results

因此，为了保证本文中的匹配方法能够产生不同的相似性最大匹配，本文为中的每一颗生成树引入一个边的集合，使得， ，并且任何由得到的相似性最大匹配一定不包含边集合中的任一条边，这里的边集合被称为的排除边集（exclusion edge set）。例如上面的例子当中，如果让的排除边集，那么通过生成树的一个准确匹配就没有得到满足的相似性匹配结果。该方法的正确性将通过下面的定理3.1得到证明。

对于查询图的每个生成树，均通过一组边的序列来进行表示，即，其中是查询图中顶点的个数，作为第条边或第层。本文使用一种递归的方式以深度优先搜索模式，即从到来生成最小生成树集合(spanning tree)。

为了便于最小生成树集合的生成，本文中为查询图的每一条边分配一个唯一的ID。初始时先按照边的ID号的大小生成一个初始的生成树。遵循这样一种规则，规则3.1：对于，是连通的，并且是中连接中ID号最小的边。产生生成树集合的具体过程如下：

使用双层递归的方式形成生成树集合，其中第一层递归实现向下遍历的过程，对应深度优先遍历；第二层递归实现交替变换并重新排序，从而产生生成树。首先按照上述规则初始化一个查询图的生成树作为初始的生成树加载到中，此时。记录当前被访问的边，其中初始值为1，是当前的生成树，即为生成树的边排除集，记录得到的过程中被替换的边，为编辑距离阈值。然后按照深度优先遍历方式从上到下，即从到，依次进行判断。若不满足条件则跳出第一层递归，执行第二层递归。只要边排除集中边的个数小于，并且在中至少存在一条边能够替换来形成另外一颗生成树，那么就用来记录这些边中ID号最小的边，用它来替换，并且通过按照规则3.1重新排序，从而产生新的生成树；更新这课新生成树的边排除集合和生成树集合。若不满足第二层递归的条件，那么终止第二层递归，执行第一层递归的上一级，以同样的方式继续执行，直到没有符合条件的边存在。具体算法如下：

表3.5 生成查询图的生成树集合QST

Table 3.5 Generating the spanning tree set QST of query graph

|  |
| --- |
| **算法3.5** |
| **输入：**：当前的生成树，初始值按上述规则生成；  ：生成树的边排除集，记录得到的过程中被替换的边；  ：当前所在的层，初始值为1；  ：编辑距离阈值；  ：生成树集合  **输出：**   1. If() 2. **;** 3. If() 4. If(在中至少存在一条边能够替换来形成   另外一颗生成树)   1. ，其中是中ID号最小的边，可以   用它来替换来连接中剩余的边来形成另一颗生成  树；   1. 重新排序 ，其中按照规则3.1重新排   序中的边，同时固定  的顺序不变；   1. ; 2. ; 3. ; 4. End If 5. End If 6. Return ; |

如下图所示是图3.9(b)中查询图对应的生成树集合的产生过程，其中编辑距离阈值。假设边的ID为。首先按照规则初始化生成树，此时。初始时，执行算法的第一层递归，直到截止。然后执行第二层递归的判断，此时，且存在符合条件的边替换，在中，能够替换连接中剩余的边来形成另一个生成树，且ID值最小的边的为，因此用替换。重新排序，因为，所以排序后依然为，因此，更新，。继续递归判断，此时虽然，但是在 中没有符合条件的边能够替换，此时第二层递归结束，返回第一层递归的上一级。此时，，，。在执行第二层递归的判断，此时，也存在符合条件的边替换，在中，能够替换且满足条件的边为，因此用替换。重新排序，因为，所以排序后变为，因此，更新，。此时虽然，且在 中存在符合条件的边替换。所以用替换，同样按上述方法得到生成树，此时，，此时，因此返回第二层递归的上一级，同样在 中存在一条边可以替换边，则用替换。同样得到生成树。没有再符合条件的边存在，因此本次第二层的递归结束，再次返回第一层递归的上一级，即。继续按上述方式递归处理，直到没有符合条件的边存在。最后。



图3.10生成查询图的最小生成树集合

Fig.3.10 generate the minimum spanning tree set of the query graph

**定理3.1** 给定的两个生成子图和，假设通过算法3.5生成了两个不同的生成树和，使得包含，包含，进一步假设，且，那么。

证明。本文中对于每个生成树对应的排除边集，均通过一组边的序列来进行表示，即，其中是排除边集中边的个数，在获得的枚举过程中，在初始的生成树中首先被替换掉，接下来在下一颗生成树中被替换掉，以此类推。在不损失普遍性的情况下，假设。因为，直接可以得到。

情况一：，。此时，因为且。根据算法3.5，一定是通过首先从中替换来枚举得到的。因为包含，而包含，所以包含，然而不包含，所以。

情况二：假设和分别是和中满足的第一条边。那么，通过算法3.5一定存在一个，其中。很明显可以得到和均是由枚举得到的。那么，既包含又包含，从枚举首先替换得到，从枚举首先替换得到。在不损失普遍性的情况下，假设，，并且。根据算法3.5，从枚举迭代地替换层数不小于的边得到。因此，包含，那么包含，又因为不包含，所以。

综上所述定理3.1得证。证毕。

定理3.1显示了只要每一个边排除集在匹配时能够被强制执行，那么产生的所有相似性最大匹配都是不同的，定理3.1的成立保证了本文使用方法的正确性。

**定理3.2** 对于至多少条边的生成子图，存在一颗生成树，使得包含，且不包含中的任何边。

证明。让表示比缺少的边的集合，即。按照初始的生成树中是否包含中的边来分析，存在以下两种情况：

情况一：初始的生成树不包含中的边。此时，定理成立。

情况二：初始的生成树包含中的边。假设在初始的生成树中包含条中的边，记为，其中。作为算法3.5的一部分，我们连续地执行这一部分来到达层，然后再执行迭代替换并重新排序阶段通过边代替来形成。注意，既可能是，也可能不是中的边，现在我们只需关注。由于每次替换边之后会根据边的序列号进行排序，中的边在中的位置可能会改变。然而，因为我们只对层数不低于的边重新进行排序，中层数最低的边一定比低，即。因此，算法3.5向下走到来替换。继续向下执行直到找到一颗生成树，满足不包含中的任何边，且是的一个子集。因此，包含，且不包含中的任一条边。

综上所述定理3.2得证。证毕。

定理3.2保证了所有与满足编辑距离阈值的生成子图均能通过中的一颗生成树得到，保证了本文所使用方法的完整性。

**定理3.3** 通过算法3.5得到的查询图生成树集合是最小的。

证明：反证法。

假设通过算法3.5得到的查询图生成树集合不是最小的。进一步假设即对于生成的两颗生成树和，存在一个查询图的生成子图，使得既包含，也包含。显然地，假设不符合定理3.1，因此假设不成立。那么可以得到通过算法3.5得到的查询图生成树集合是最小的。

综上所述定理3.3得证。证毕。

定理3.3保证了查询图生成树集合的最小性。也就是说，从中移除一颗生成树，由通过边排除集得到的最大相似性匹配不能由另外一个通过边排除集得到。

### 3.3.2有效的存储方式

本文中使用一种树形结构来有效地存储查询图的最小生成树集合，这种树形结构我们称之为深度优先遍历树(，简称)，它不仅能够节省存储所需的存储空间，还可以加快匹配时的速度。主要思想如下：当通过在中替换产生时，本文在中只存储他们的共同前缀一次，和其余的边分别作为两个不同的分枝进行存储。

在一颗中，每一个节点代表生成树的一条边，其中根节点代表最小生成树集合中生成树的头结点。初始的生成树首先被加载为的最左边的路径。在算法3.5中，迭代地，如果一条边代替了中的边来形成另一颗生成树，那么作为临近的右兄弟被加载到中，使得和能够共享前缀。明显地，深度优先遍历树所需空间为。为查询图的最小生成树集合中所包含的生成树的个数，为查询图中顶点个数。

根据图3.10中的查询图，通过算法3.5得到的的最小生成树集合对应的存储结构如图3.11所示。图中箭头既表示一棵生成树中边的顺序，又表示从根到叶子节点给出一颗生成树的路径。首先将加载为的最左边的路径。然后根据生成树集合的形成过程，首先将替换成，形成第二条分支。然后用替换形成分支，再用替换成形成分支。同样形成分支。然后用用替换，并重新排序形成分支。以同样方式一次进行存储，最后形成如下所示。



图3.11最小生成树的存储

Fig.3.11 the storage of minimum spanning tree

## 3.4 子图全匹配

再按上述方法分别为图数据流进行分区，并为查询图创建生成树集合后，接下来执行子图相似性匹配操作。本文的主要思想是初始时按照静态图上的方法进行子图全匹配，当图数据流进行更新之后，在原有的结果上直接进行增量的维护。

### 3.4.1 QI-Sequence

本文使用[20]中对于图的一种编码方式，来实现文中的算法。一个QI-Sequence 表示的一颗生成树，记为，它由一组序列表示，其中每一个被称为的一个入口，同时对应的一个顶点，入口对应于的根顶点。中所有的边由中的生成边（spanning edges）表示，每一个入口有且只有一条生成边，记为，其中。此时称是在生成树中的父亲。中其他的边均称为中的非生成边（not spanning edges），与入口相关的非生成边的集合记为。也就是说，中存放的是与入口相关的生成树中的边，中存放的是与入口相关的中的边。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| S[1](u1) | NULL | Φ |
| S[2](u2) | (S[2],S[1]) | Φ |
| S[3](u3) | (S[3],S[1]) | (S[3],S[2]) |
| S[4](u4) | (S[4],S[1]) | (S[4],S[3]) |



(a)

(b)（（））请在此放置您的文字

图3.12QI-Sequence的实例

Fig.3.12 The instance of QI-Sequence

图3.12为图3.9(b)中查询图的3.9(d)中的生成树所对应的QI-Sequence。图3.12(a)中实现表示生成边，虚线表示非生成边。对应的每个入口的信息如3.12(b)所示。

### 3.4.2子图全匹配

本文进行子图全匹配的主要思想是以深度优先的方式搜索，为每一个生成树在图数据流中找到满足条件的匹配，从而得到和查询图满足编辑距离的所有最大相似性匹配。为了加快子图全匹配的速度，本文不需要枚举所有的生成树，而是采用一种边枚举边判断的方式进行匹配。只要判断出中的一个节点不满足当前映射的扩展，那么就立刻终止扩展到的下一层。具体过程如下：

具体分为两个递归部分，第一个递归部分是递归匹配部分（1-15行），第二个递归部分是递归产生每个生成树所对应的QI-Sequence（16-23行）。其中第二部分的过程与算法3.5相同，不再进行详细阐述。递归匹配时对每个分区中的每个连通图分别进行处理，处理前先通过顶点标签集合进行过滤，当查询图的顶点标签集包含于该连通图的顶点标签集合时才继续向下执行匹配操作，否则将其过滤。因为本文中的匹配要求顶点必须匹配，边可以缺失，所以如果该连通图不包含查询图中的所有顶点标签，那么该连通图中的任何子图都一定不会成为最终的匹配结果，因此将其直接过滤掉。

在进行完上述过滤之后，开始执行匹配操作。初始时，当前所在层数，中所存储的，只有最左边的路径，即只有按照规则3.1所生成的初始的生成树，同时中也只包含初始生成树所对应的QI-Sequence，。为当前的部分映射函数，当前在上的部分映射是一个向量，其中是从入口映射到图数据流中的一个顶点，并且表示从1到入口的前缀，也就是说由入口的所有和组成。

如果当前所在的层，那么就可以从查询图的顶点索引集当中随机选取一个顶点作为初始匹配顶点，否则所选则的顶点不仅要在查询图的顶点索引集当中，还要满足存在一条与该顶点相连的边与生成边相匹配，也就是说该顶点也必须是它在中的父亲对应的映射到G中的顶点的邻居。并且记录当前扩展到该顶点时中所缺失的的条数。如果当前生成树和查询图的编辑距离与扩展到该顶点时中所缺失的的条数之和小于总的编辑距离阈值，那么继续向下执行，否则不再继续扩展。继续判断当前进入的匹配是否包含中的边，如果不包含则可以扩展到该顶点。只要当前层数，继续重复执行上述扩展操作，否则，即是通过生成树引入的与查询图满足编辑距离阈值映射函数为的一个最大相似性匹配。第4-11行的作用主要是判断当前引入的匹配顶点是否满足向下扩展的条件，即是否满足编辑距离阈值，并且不包含当前生成树所对应的边排除集合中的任何边。

处理完当前生成树所引入的所有匹配结果后，按照第16-23行所示算法继续生成另一颗满足条件的生成树，然后再执行匹配的过程，直到处理完所有生成树为止。由此就可以得到所有与查询图满足编辑距离阈值的最大相似性匹配。具体算法如下：

表3.6 子图全匹配算法

Table 3.6 The algorithm of similarity all-matching

|  |
| --- |
| **算法3.6** |
| **输入：**：图数据流当前的分区  ：当前所在的层，初始为1  ：当前的QI-Sequence，初始时是  ：查询图中每个顶点的当前索引集  ：当前的部分映射函数  ：当前的部分映射函数在上缺少的边数  ：编辑距离阈值  ：存储，初始时只有最左边的路径  **输出：**   1. For each 2. For each 3. If() 4. If() 5. 随机选择一个顶点; 6. Else 7. 选择一个顶点，该顶点满足存在一条边能   够与生成边相匹配； |
| 1. 扩展到顶点时中没有匹配的边   的条数；   1. If() |
| 1. If(当前引入的匹配不包括中的任何边) 2. ; 3. If() 4. ; 5. Else 6. Output ; 7. If( && ) 8. If(如果在中至少存在一条边能够替换   来形成另一棵生成树)   1. ，其中 与算法3.5中的意思相同; 2. If(通过用替换得到的一个的生成   树还没有被得到)   1. = reorder ; 2. ; 3. ; 4. ; |

### 3.4.3子图全匹配增量维护

通过上述内容就实现了图数据流上相似性子图全匹配技术，为了实现本文增量更新的思想，当在每个时间戳上有更新到达时，不需要重新进行一遍完整的匹配过程，本文使用如下思想进行增量处理：首先按照最近邻分区的方式为图数据流进行分区，当t时刻图数据流上的改变操作流到达时，按上述方式增量对分区进行维护，并统计图数据流的改变操作流中的所有改变操作分别所属的分区，然后对于没有发生改变的区不需要进行处理，只对存在更新的区重新进行匹配，从而得到最终匹配结果。

注意，由于图数据流更新的局部性，在某一时间间隔内的改变操作往往集中于某个或某几个区中，这些区的个数通常远远小于整个图中分区的个数。因此通过这样对整个图数据流的局部进行处理，并且增量进行更新，大大提高了图数据流上子图全匹配的速度。

## 3.5实验

本节对上述提出的算法进行实验，通过在真实数据集和合成数据集中进行实验，并与传统静态图上子图全匹配的方法进行对比，来证明我们提出方法的正确性和效率。

### 3.5.1实验环境

本节中的所有实验都是在Eclipse JAVA平台下完成的。所有算法均由JAVA语言实现，实验运行环境如下：

处理器：Inter(R) Core(TM) i7-2600 CPU @3.40GHz，3.40GHz。

内存(RAM)：8.00GB。

操作系统：64位Windows操作系统。

### 3.5.2数据集

本章分别使用真实数据集和合成数据集来进行实验：

真实数据集1：DBLP，是一个共引用图，数据来源于<http://www.informatik.uni-trier.de/~ley/db/>。DBLP数据集包含计算机科学领域的自然科学出版物，每个顶点是一个带有属性的文章（如，出版年份），根据文章的你年份，提取一个子图，包括93560条边和13634个顶点。随机在不同时刻产生图数据流上的更新，平均更新大小为N1。

真实数据集2：人类蛋白质交互网络（The Human Protein Interaction Network，简称HPIN），数据来源于<http://www.hprd.org/download>。这个图包括9460个顶点，37081条边，和307个不同的顶点标签。随机在不同时刻产生图数据流上的更新，平均每次更新大小为N2。

合成数据集：我们简称为SYN，合成数据集按如下方法产生。首先随机产生一个生成树，然后在生成树上随机添加边，最后根据幂律分布随机为顶点分配标签。使用幂律分布是因为很多真实数据都符合幂律分布。产生的合成数据集大小为：， （即标签种类为50种）。

根据上述数据集，分别为每个数据集通过使用随机步的方式随机从各个数据集中产生如下查询图集，来研究不同查询图设置对结果的影响：

查询图：随机产生8个查询图集合，记为，，，，，,，，其中每一个查询图中有个顶点，且平均的顶点的度为4。对于每一个，所有查询图均为相应图数据流的子图。

本章中所执行的算法如下：

SimMatch：静态图上的子图全匹配算法，即不带有分区策略的子图全匹配算法；

NNJuegment：带有分区策略的子图全匹配方法，即本文提出的图数据流上的子图全匹配方法。

### 3.5.3实验结果

首先我们检验了最近邻分区阶段结构剪枝的有效性。如图3.13所示，分别在DBLP，HPIN和合成数据集上进行了实验验证，主要测试在不同初始图数据流顶点个数下，剪枝后初始图数据流中所剩顶点个数与初始图数据流中总的顶点个数的比值。可以看到当初始图数据流中顶点个数从1000变化到6000的过程中，DBLP的剪枝效果大约在83%左右，HPIN的剪枝效果大约在74%左右，合成数据集SYN大约在85%左右。

图3.13结构剪枝效果 vs 初始图数据流顶点个数

Fig.3.13 structural pruning power vs the vertex number of original streaming graph

## 3.6总结

本章主要讨论了图数据流上增量子图相似性匹配技术的研究，先提出了一种带有剪枝功效的分区方法——最近邻分区，并实现对分区的增量维护，之后通过为查询图创建生成树集合，通过递归的方式进行子图全匹配，并通过在每次图数据流更新时只对改变的分区重新匹配的方式实现了结果上的增量维护。最后在真实数据集和合成数据集上进行实验，并与传统静态方法进行对比，验证了方法的正确性和有效性。

# 带权图数据流上增量子图近似全匹配

在第三章中主要是对图数据流上增量子图全匹配问题进行了研究，其中研究的图数据流主要是针对顶点带标签的无权图进行的，且研究的匹配主要是相似性匹配(准确匹配视为相似性中编辑距离阈值的一种特殊情况)。不仅如此，第三章中在对图数据流的更新中我们均认为每一条边只能有存在和不存在两种情况，且若存在，只能存在一次。然而，现实世界中各种数据及其相关应用的存在使得这种顶点带标签的无权图，以及这种单一的图数据流的更新方式无法对现实中所有的情形进行模拟。

例如，在社交网络中，图结构往往用于表示用户之间的交互关系，其中顶点表示不同的用户，边表示用户之间的联系。明显地，人与人之间的交互关系在不断发生变化，我们就可以用这种边上带权重的图数据流对这种动态更新的交互关系进行建模。边上的权重用于表示不同用户（顶点）之间联系的频率。当两个用户（顶点）之间进行一次联系时，将这两个用户（顶点）之间边的权重加1。当两个用户（顶点）之间没有任何联系时，我们可以认为两个用户（顶点）之间存在一条边，且这条边的权值为0。研究者往往想要了解不同时间戳上社交网络中存在哪些满足某些边上频率要求的特定的模式，以便进一步了解整个社交网络的结构。

在其他领域中也会有类似的情况出现。比如IP网络和交通路网。在IP网络中，可以用图来表示IP地址之间的联系，其中顶点表示一个IP地址，顶点之间的边表示两个IP地址之间进行一次联系，边上的权重表示联系的次数。随着用户对计算机的不断使用，不同IP地址之间的联系不断增加，可以将一些经常出现的IP网络攻击方式作为一种模式保存下来，并通过实时的对IP网络与这些已知的IP网络攻击模式进行子图全匹配检测，就可以用于实时检测IP网络攻击。在交通路网中，可以将交通情况建模成为对应的图，将不同的地点作为图的顶点，两个地点之间有车通过则作为一条边，图上的权重表示两个地点之间经过车辆的辆数，可以将一些已知的可能的恶意攻击提前被建模为一系列交通图模型，用于检测交通流数据中的恶意攻击。对于这种IP网络攻击和交通流攻击，我们需要实时地报告所有可能的攻击，也就是说，随着数据的更新要实时的找到模式和数据之间的匹配。在这些情况中就对实时性的要求比较高，因为任何多一点的时间都可能导致巨大的损失。

通过上面的实例可知，第三章中无权图数据流上的相似性子图全匹配不能满足上述情况，需要使用带权图进行建模，并且图数据流的更新不拘泥于简单的边的增加和删除。因此提出了本章所研究的问题，集带权图数据流上增量子图全匹配问题，并且本章中图数据流的更新允许同一条边被增加或删除多次，同时为了满足图数据流实时响应的特性，本章采用一种近似匹配的方法来解决我们所提出的问题。

## 4.1问题定义

同样在对本章问题进行具体阐述之前，先对本章中所涉及的一些基本概念和问题进行定义。

### 4.1.1基本概念定义

**定义4.1**（带权图）：一个顶点带标签且边上带权的图是一个五元组，，其中是的顶点集合，是的边集合，是的顶点的标签集合，标签函数定义了映射,是中边的权重集合。

注意：本章中所研究的带权图中边的权重不同于以往的权重，是指两个顶点之间联系的频率，即权重越大，两个顶点之间联系的频率越高；



图4.1带权图

Fig.4.1 the graph with weight

如图4.1所示，是一个边上带权重且顶点带标签的图，其中顶点集合，边标签集合，顶点标签，标签函数定义如下：



边的权重集合

。

**定义4.2**（子图）：给定两个顶点带标签的图和，如果图的顶点集合和边集合分别为图的顶点集合和边集合的子集，（即，且），并且和有相同的标签函数，（即），同时中边上的权值与中对应边上的权重值相等，则图为图的子图。

**定义4.3**（子图同构）：给定两个顶点带标签的图和，子图同构是存在一个单射函数，满足下列条件：

（1），且；（2）；

（3）。

此时就认为图和图是子图同构的，也可以说包含或者说中包含的准确匹配。

**定义4.4**（图改变操作）：给定一个图，对于，边的插入和删除可以通过一个三元组表示，其中表示边的插入或删除操作。由此得到图改变操作的正式定义如下：给定一组图的操作，其中。可以通过转变为，记为。

注意：本章中所研究的图数据流上的更新主要是针对的是复杂更新，即图数据流上的边不仅仅有存在和不存在两种可能，也就是说同一条边能够被插入和删除多次。边上的权重用于表示边存在的次数，如果两个顶点之间不存在边，可以认为这两个顶点之间边的权重值为0.

**定义4.5**（图改变操作流）：给定一个初始图，图的改变操作流是一组图改变操作的序列，其中t代表时间戳，记录改变操作发生的时间。。

以图4.1中为起始图，进行更新操作，如图4.2所示。







图4.2带权图上的图改变操作流实例

Fig.4.2 Illustration of a graph change operation stream on weighted graph

如图4.2所示，是一个带权图数据流上的图改变操作流的实例，即执行操作。更新过程和无权图上相似，即当第一组图改变操作到达时，在初始图上执行相应的更新操作，从而得到图，同样地可以通过更新操作和分别得到和。同样由于边的删除导致可能一些顶点成为孤立顶点，此时我们也并不将孤立顶点删除，以便之后继续进行边的更新操作。与简单更新操作不同的是，本章研究的更新允许边多次进行插入和删除，如在更新操作和中，都有对边的删除操作；不仅如此，对同一条边的更新操作也可以在同一批更新中多次出现，如更新操作中，有两次对边的插入操作。

### 4.1.2 带权图数据流上增量子图近似全匹配

本章所针对个图数据流上的更新主要是复杂跟新，即同一条边可以多次进行插入和删除，因此本章研究的图数据流是带权图数据流。在很多时候想要实时的返回最终的匹配结果，而图数据流的更新速度往往是很快的，因此为了满足这种实时更新的特点，本章采用近似匹配的方式进行处理。

本章假设在查询开始之前存在一个初始的图数据流和一个查询图，然后随着时间推移，图数据流的改变操作从时间戳为1开始依次到来。本研究的期望结果是在每一个时间戳上，可以自动地近似返回该时间戳下对应个静态数据图中有哪几个子图和已有的查询图相匹配。问题的正式定义如下：

**问题4.1**（带权图数据流上增量子图全匹配问题）：给定一个带权图数据流和一个查询图，从时间戳为开始，我们想要连续地报告该时间戳下的所有子图，其中满足在时刻下与查询图是子图同构的。

**定义4.6**（编辑距离）：给定两个边上带权重且顶点带标签的图和，从图转化为图时，两图中各个对应边之间权重的欧氏距离即为图到图的编辑距离，记为，其中若如数据流中两个顶点之间的边不存在，则认为这两个顶点之间边的权重值为0。

和之间的欧式距离公式定义如下：

 （4.1）

其中是指查询图中的第条边和与之相匹配的数据图中的边之间权重的差值，为查询图中边的条数。若查询图中存在的边在图数据流中不存在，则认为在图数据流中这条边的权重为0。与第三章相同，本章考虑编辑距离时也只考虑图数据流比查询图少边的情况，不考虑图数据流比查询图多边的情况，当多边时认为编辑距离为正无穷。



图4.3编辑距离实例

Fig.4.3 the instance of edit distance

如图4.3所示，根据公式4.1可知，查询图与图的编辑距离。可以看到，对于查询图中的边在图中没有一条边与之对应，因此我们假设中与边对应的边存在，但权重为0，如图4.3(a)所示。

为了满足图数据流实时响应的特性，本章中进行子图全匹配时采用近似匹配的方式进行处理，具体定义如下。

**问题4.2**（带权图数据流上增量子图近似全匹配问题）给定一个带权图数据流，一个查询图和一个编辑距离阈值，从时间戳为1开始，在每个时间戳上，我们想要连续地报告该时间戳下的所有可能的子图，其中是在时刻下与查询图是满足阈值为的子图同构关系。

**定义4.7**（最大匹配）查询图在数据图中的一个匹配是最大的当且仅当在与有相同映射函数的任何适当的超图（即是的超图，且），以及与有相同映射函数的任何子图中，均不存在图使得成立，即此时查询图和配之间的编辑距离最小。

**问题4.3**（带权图数据流上增量子图近似最大全匹配问题）给定一个带权图数据流，一个查询图和一个编辑距离阈值，从时间戳为1开始，在每个时间戳上，我们想要连续地报告该时间戳下的所有可能的子图，其中是在时刻下中与查询图满足边集距离阈值的不同的最大相似性匹配。

## 4.2子图全匹配

在本节主要对带权图数据流上增量子图近似全匹配问题的研究进行介绍，由于本文针对的图数据流上的更新是指复杂更新，即允许边的多次插入和删除，使得第三章提出的方法不能直接用于本章所研究的问题当中。接下来本节介绍介绍第三章提出方法的改进方法，使其能够用于解决带权图数据流上增量子图近似全匹配问题。

在本章中同样假设初始的图数据流和查询图都是已知的。从时刻开始图数据流进行更新变化，并在变化过程中对查询图进行全匹配操作，连续地自动返回每个时间戳上的匹配结果。通过对匹配结果的特征的分析，我们的匹配结果同样需要满足以下条件：（1）作为结果的匹配子图必须是连通的，（2）作为结果的匹配子图的顶点个数必须等于查询图中顶点的个数，（3）作为结果的匹配子图的节点标签集合必须等于查询图的节点标签集合，即，（4）作为结果的匹配子图边的个数不多于查询图中边个数，（5）作为结果的匹配子图与查询图之间的编辑距离使用欧氏距离计算。

### 4.2.1基于权重的最近邻分区

本章研究的带权图数据流中的权重指两个顶点之间联系的频率，不同于第三章中的无权图数据流，因此不能直接将第三章中的最近邻分区方法直接用于本章之中，在本节中将最近邻分区方法进行改进使其能够用于带权图数据流上的分区。

基于权重的最近邻分区的步骤如下：

（1）结构剪枝：将图数据流中不包含于查询图的顶点标签集合中的顶点和与该顶点相连的边删除；

（2）基于权重的最近邻判断：根据查询图的结构以及边的权重判断数据图中对应顶点的最近邻，并通过适当增加一些虚边，以尽量减少之后合并两分区的可能性；

（3）分区：通过连通性来判断分区。

与最近邻分区相同，基于权重的最近邻分区也要执行上述三个步骤，其中第一步和第三步与最近邻分区完全相同，因此在本章中不再进行详细叙述，本节集中介绍第二步的具体执行过程。

由于本章中研究的带权图数据流中的权重是指两个顶点之间联系的频率，自然地，可以认为一条边与查询图中对应边之间的编辑距离越大，之后成为匹配结果的可能性越小，一条边与查询图中对应边之间的编辑距离越小，之后成为匹配结果的可能性越大，基于这种思想，对图数据流进行最近邻判断。注意，在查询的过程中人们往往希望查询权重满足适当大小的图的匹配，往往不关心频率过高或者过低的极端情况，因对于那些频率过高的边，虽然可能会被频繁更新，但依然不对其进行维护。

为了记录查询图中各不同节点标签的最近邻信息，我们定义 表示标签为的顶点与标签为的顶点距离最短的距离为，分别表示与距离最近的标签中边权重的最大值。我们使用一种多数据链表存储。具体说明如下图：



图4.4多数据链表结构

Fig.4.4 Multi-data linked list structure

对于头结点，data1用于存储所要查询的标签，每个链表的头结点所对应的标签均不相同，可以作为唯一的标识进行索引。data2用于存储查询图中符合与data1所存标签距离为的顶点标签种类的个数。data3用于存储data2中所有标签对应的顶点与data1中标签对应的顶点之间边权重的最小值和最大值。Next项用于存储指向下一个位置的指针。对于非头结点，data1用于存储在查询图中与头结点的data1所存标签距离为的顶点标签，data2用于存储两标签节点的具体距离，data3用于存储data1中标签对应顶点与data2中标签对应顶点权重的最大值和最小值。Next项用于存储指向下一个位置的指针。对于尾节点，next标记为NULL，表示后面没有符合条件的节点存在。在本文中，只考虑查询图中距离为1的顶点。在本文中使用几组标签不同的多数据链表结构组成NN索引链表，用来检索查询图中符合条件的顶点标签。

图4.5所示是图4.4(b)中查询图对应的NN索引链表的实例。通过索引结构可以得到查询图中各标签的最近邻集合NN，且，，，，所以，，， 。得到各个标签的最近邻的信息后，按照上述所示方式进行存储，结果如图4.5所示。



图4.5多数据链表结构实例

Fig.4.5 the instance of multi-data linked list structure

基于一条边与查询图中对应边之间的编辑距离越大，之后成为匹配结果的可能性越小，反之一条边与查询图中对应边之间的编辑距离越小，之后成为匹配结果的可能性越大这种观点，为图数据流进行分区。基于权重的最近邻分区主要思想如下：在结构剪枝之后，确定分区之前先对每个顶点根据查询图的结构并结合权值进行最近邻判断，通过增加一些辅助边，使一些以后被连接可能性大的顶点尽量在一个分区当中，从而减少每次更新后对图数据流分区进行维护的代价；而且当子图中有一条边与查询图对应的边的编辑距离大于等于给定的阈值，那么这个子图与查询图的编辑距离一定大于等于给定的阈值。具体地说，（1）首先要求根据查询图的结构进行判断，使得图数据流中距离为1的两个顶点，在同一个分区内，对于距离大于等于2小于等于3的两个顶点，为两条顶点之间加一条辅助边，使得满足这种距离要求的两个顶点尽量在同一个分区中；（2）其次根据查询图各边的权重进行判断，对于临近的两个顶点，如果这两个顶点之间的边与对应的查询图中的边的编辑距离大于，则将该边过滤。由于图数据流是不断更新变化的，如果的取值过小，会使得能够成为匹配结果的边被过滤，导致准确性下降；反之如果取值较大，会使得过滤效果较差，导致时间代价增大，因此，一个适当的是非常必要的，的具体取值大小将在本章的实验部分中通过在不同数据集上进行实验来判断，基于权重的最近邻判断就是在基本的最近邻判断的基础上增加了条件（2）的判断，具体操作步骤如下：

首先通过索引结构得到查询图中每个顶点对应的索引集和查询图中各标签的最近邻集合。初始化集合，已访问过的顶点集合和辅助边集合（auxiliary edge set），其中，，，。然后根据基于权重的最近邻集合，根据查询图的结构和权重值对图数据流进行遍历。从查询图中选取候选顶点最少的顶点作为初始，从中选择与相匹配的第一个顶点开始遍历，根据查找与标签中标签相匹配，与距离最近的顶点，且，其中是查询图中与相匹配的点，将加入到集合中，若与距离是1，则将边插入到边集合中，若距离大于等于2小于等于3则将边插入到边集合和中。接下来按照上述方式依次处理其他顶点，当没有其他满足条件的顶点可以处理时，将查询图中的各顶点对应的索引集中所包含的所有中的顶点都删除。若此时查询图各顶点对应的索引集均为空则遍历结束，否则在不为空的集合中同样在候选结点数量最少的索引集合中随机找到一个初始顶点按上述过程继续遍历，直到查询图各顶点对应的索引集均为空时结束。

例如，对于图4.6(a)中的图数据流和4.5(b)中的查询图，首先通过算法3.1结构剪枝后得到如图4.6(b)s所示结果。然后执行基于权重的最近邻判断，通过索引结构可以得到查询图中每个顶点对应的索引集：，， ，以及查询图中各标签的基于权重的最近邻集合：，，，。初始化集合，，和，以及已访问过的顶点集合和辅助边集合，其中，，，，本例中设置，。然后根据基于权重的最近邻集合对图数据流进行遍历。因为中所含的候选阶段最少，因此从中随机选择与相匹配的顶点作为初始顶点开始遍历，如选择。的顶点标签，由可知，在查询图中的邻居为，图数据流中与距离最近的标签为的顶点为，与距离最近的标签为A的顶点为，与距离最近的标签为C的顶点为。分别处理，和。对于，由于，因此将边放入集合中，将的权重加入到中，且将顶点加入到已访问过的顶点集合中，此时，，，。接下来已同样的方式访问顶点和。处理时，对于它的一个最近邻，由于，因此将边过滤，同时将产生的孤立顶点删除。继续处理符合条件的顶点，直至没有符合条件的顶点存在。此时，，，。将查询图中的各顶点对应的索引集中所包含的所有中的顶点都删除。继续选择下一个顶点作为初始顶点按上述方法进行处理，当处理到顶点时，由可知在查询图中的邻居为和，与距离最近标签为的顶点是，距离为2，且，此时在和之间增加辅助边，如图4.5(c)中虚线所示，并将分别加入到集合和中.。按上述方式进行处理直至整个索引集为空。然后根据连通性分区，结果如图4.5(d)所示。









图4.6基于权重的最近邻分区实例

Fig.4.6 The instance of the nearest neighbor partition based on weight

具体算法如下：

表4.1 基于权重的最近邻判断算法

Table 4.1 Nearest Neighbor Judgment Algorithm Based on Weight

|  |
| --- |
| **算法4.1** |
| **输入：**：初始的图数据流  ；结构剪枝后的图数据流  ：查询图中每个顶点的当前索引集  ：查询图中每个顶点的最近邻集  ：编辑距离阈值  ：用来限制权值过滤的上界  **输出：**   1. 初始化，其中,   ，，;   1. 初始化边辅助集合; 2. 初始化已访问顶点集合; 3. 初始化队列; 4. While(只要有一个不为空) 5. 在中选择候选顶点个数最少的一个作为起   始，选择中的第一个顶点作为起始顶点；   1. Push() into ; 2. While(不为空) 3. ; 4. If() 5. ; 6. 找到标签为的列表入口，让   ,…,,  ,…,  其中;  For each   1. 在中找到离它最近的顶点，其中   ；   1. If(&&) 2. Push() into ; 3. If(，其中是中与   相匹配的顶点)   1. If() 2. ; 3. ; 4. If() 5. ; 6. ; 7. End If 8. End If 9. End For 10. End If 11. End While 12. ; 13. End While 14. Return ; |

### 4.2.2基于权重的最近邻分区的动态维护

基于权重的最近邻分区和普通的最近邻分区的主要区别就是它所针对的图数据流的更新是复杂更新，即同一条边可以被插入或删除多次。基于复杂更新的特点，为了避免每次更新后重新对图数据流进行一次完整的分区而导致的过高的时间复杂度，本节进一步介绍当图数据流更新时，在基于权重的最近邻分区上直接进行增量维护。同样，对于图数据流上的更新主要分为边的插入和删除，其中图数据流中顶点的插入和删除都可以看作是边的插入和删除。边插入操作的分类以及每种情况下的增量更新方式如下所示：

（1）插入边已经存在于图数据流当中：此时只需要确定边所在的图边，然后将边的权重值加1即可；

（2）插入边不存在于当前图数据流当中，且其中存在一个顶点的标签不包含于查询图的标签集合中：在这种情况下，不需要将边插入到图数据流中，因为如上所述，本文中要求最终的匹配结果与所给查询图的顶点标签必须保持一致，如果一个顶点的标签不在所给查询图的标签集合当中，那么与该顶点相连的所有边一定不会成为最终的匹配结果，因此不需要将边插入到图数据流中；

（3）插入边不存在于当前图数据流当中，且它的两个顶点的标签和均包含于查询图的标签集合中，但和中只有一个顶点已经存在于当前图数据流当中，假设存在的顶点是；对于这种情况，相当于插入一条边，且当前图数据流中。此时，先找到存在的顶点所在的图，然后将边直接插入，同时将顶点和边分别插入到图的顶点集合和边集合当中，并且将值加1；

（4）插入边不存在于当前图数据流当中，且它的两个顶点的标签均包含于查询图的标签集合中，同时和两个顶点均已存在，并属于同一分区当中，但属于不同的连通图；这种情况下，将两个顶点所属的同一分区的两个不同连通图进行合并，即合并其顶点集合，边集合，顶点标签集合以及边权重集合。然后将边插入到合并之后的边集合当中，并且将的值加1；

（5）插入边不存在于当前图数据流当中，它的两个顶点的标签均包含于查询图的标签集合中，且和两个顶点均已存在，并属于同一分区当中，同时两个顶点属于同一个连通图；这种情况直接将边插入到该边所对应图的边集合中，并更新即可；

（6）插入边不存在于当前图数据流当中，它的的两个顶点的标签均包含于查询图的标签集合中，且和两个顶点均已存在，但属于不同的分区；这种情况是最为复杂的情况，也是更新时代价最高的情况，本章使用的基于权重的最近邻分区方法尽量减少了这种情况出现的可能性。这种情况先将两个分区合并为同一个分区，然后再根据顶点所在位置按上述对应情形中所述方法进行操作。

同样为了加快对图数据流上分区的维护，在插入边之前，本文先通过以下两种方式对边进行过滤：（1）如果在查询图中，与插入边的两个顶点所对应的节点标签和所对应的顶点之间的距离大于1，那么这条边一定不会成为最后的匹配结果中的一部分，因此满足这种条件的边本文不进行插入操作，直接将其过滤掉；（2）如果插入的边是在基于权重的最近邻分区中通过权值判断后被删除的边，即初始时该边与查询图中对应边的编辑距离已大于，则直接将其过滤。

综上所述，插入边时对于图数据流上基于权重的最近邻分区方法动态维护的具体算法如下：

表4.2 基于权重的最近邻分区方法的动态维护——边的插入

Table 4.2 Dynamic maintenance of nearest neighbor partition method based on weight-Insert

|  |
| --- |
| **算法4.2** |
| **输入：**：图数据流的分区  ：查询图  ：插入的边  ：编辑距离阈值  ：用来限制权值过滤的上界  **输出：**更新后的分区   1. If(，其中是查询图中与相匹配的边) 2. If(边已经存在于当前图数据流中)   3. 找到所在的图，并且;  4. else  5. If(&&)  6. If(中和所对应的标签满足)  7. If(&&)  8. 找到所在的图，然后，， ;  9. Else If(&&)  10. 找到所在的图，然后，，;  12. Else  12. If(&&)  13. If(&&，其中)  14. ，，;  15. Else  16. ;  17. Else  18. ，然后找到和的位置，并插入;  19. End If  20. End If  21. End If  22. End If  23. End If  24. End If  25. Return P; |

对于带权图数据流的更新操作，除了边的插入外还包括边的删除，在本文中与对结点的删除操作均视为对边的删除操作，删除与该顶点相连的所有边。同样，对于删除边，存在以下几种情况：

（1）删除边不存在于当前图数据流当中：删除边不存在主要是通过之前的结构剪枝或最近邻判断将这条删除边已经提前进行了过滤，因此，此时不需要执行任何操作；

（2）删除边存在，删除边后，边的权重值不为0，即没有产生孤立顶点：这种情况中只需直接确定待删除的边所在具体位置，更新边的权重值即可；

（3）删除边存在，但删除边后，边的权重值变为0：对于这种情况确定待删除的边所在具体位置并将边从对应图的边集合中删除之后，同时将边的权重值删除，若删除边后产生孤立顶点，则将该顶点从对应图的顶点集合中删除。

同样如果删除的边与查询图中对应边的编辑距离已大于，则直接将其过滤。综上所述，删除边时对于图数据流上基于权重的最近邻分区方法动态维护的具体算法如下：

表4.3基于权重的最近邻分区方法的动态维护——边的删除

Table 4.3Dynamic maintenance of nearest neighbor partition method based on weight-Delete edge

|  |
| --- |
| **算法4.3** |
| **输入：**：图数据流的分区  ：查询图  ：删除的边  ：编辑距离阈值  ：用来限制权值过滤的上界  **输出：**更新后的分区   1. If(，其中是查询图中与相匹配的边) 2. If() 3. 确定边所在图，也就是 4. If(不存在一个顶点，使得) 5. ; 6. ; 7. ; 8. Else 9. ; 10. ; 11. End If 12. End If 13. End If |
| 14. Return ; |

### 4.2.3创建生成树集

同样接下来我们为查询图创建生成树集合（Spanning tree set），创建的方式于算法3.5相似，但由于本章中的查询图的边是带有权重的，因此需要将算法3.5进一步的改进。同样本章也使用深度优先遍历树来存储查询图的生成树集合，具体方法与第三章相同，因此不再重复阐述。

类似地，为了匹配方法能够产生不同的相似性最大匹配，为中的每一颗生成树引入一个边的集合，使得， ，并且任何由得到的相似性最大匹配一定不包含边集合中的任一条边。对于查询图的每个生成树，均通过一组边的序列来进行表示，即，其中是查询图中顶点的个数，作为第条边或第层。同样使用一种递归的方式以深度优先搜索模式，即从到来生成最小生成树集合(spanning tree)。

不同与算法3.5中按照边的ID号大小生成初始的生成树，对于本章中带权重的查询图，我们首先根据边的权重，按照普里姆算法生成初始的生成树。遵循规则4.1：对于，是连通的，并且是中连接中权重最小的边。

注意，对于第三章中定理3.1，定理3.2和定理3.3同样适用于本章中的带权查询图中，因为首先定理3.1和定理3.3在证明的过程中均与查询图上的边是否有权值无关，可以直接适用于带权的查询图；其次本章是通过规则4.1对重新排序，同样只对层数不低于的边重新进行排序，中层数最低的边一定比低，即，所以定理3.2也适用于带权查询图中。因此通过以上三个定理保证了该方法的正确性，完整性和最小性。

为带权查询图产生生成树集合的过程与算法3.5相似，同样适用双层递归的方式形成生成树集合，即第一层递归实现向下遍历的过程，对应深度优先遍历；第二层递归实现交替变换并重新排序，从而产生生成树。同样用来记录当前被访问的边，初始值为1，是当前的生成树，为生成树的边排除集，记录得到的过程中被替换的边，为编辑距离阈值。具体过程如下：首先按规则4.1初始化生成树作为初始的生成树加载到中，此时。然后按照深度优先遍历方式从到依次进行判断。若不满足条件则跳出第一层递归，执行第二层递归。只要边排除集中边的个数小于，并且在中至少存在一条边能够替换来形成另外一颗生成树，那么就用来记录这些边中权值最小的边，用它来替换，并且通过按照规则4.1重新排序，从而产生新的生成树；更新这课新生成树的边排除集合和生成树集合。若不满足第二层递归的条件，那么终止第二层递归，执行第一层递归的上一级，以同样的方式继续执行，直到没有符合条件的边存在。

### 4.2.4子图全匹配

同样使用[20]中对于图的一种编码方式，来实现文中的算法。一个QI-Sequence 表示的一颗生成树，并记为。但由于本章中的查询图是带有权重的，因此在记录生成边（spanning edges）以及非生成边（not spanning edges）时，我们同时记录每条边的权重。

例如对于图4.3(b)中的查询图，按照普里姆算法可以得到它的一棵最小生成树，如图4.7(a)所示。生成树所对应的QI-Sequence如图4.7(b)所示，其中实线表示生成边，虚线表示非生成边，边上的数字表示该边的权重。对应的每个入口的信息如4.7(c)所示。



|  |  |  |
| --- | --- | --- |
|  |  |  |
| S[1](u1) | NULL | Φ |
| S[2](u2) | (S[2],S[1],2) | Φ |
| S[3](u3) | (S[3],S[1],2) | Φ |
| S[4](u4) | (S[4],S[1],1) | (S[4],S[2],3),(S[4],S[3],2) |

(c)的信息

图4.7带权查询图的QI-Sequence的实例

Fig.4.7 The instance of QI-Sequence of graph with weighted

同样我们以深度优先的方式搜索来进行子图全匹配，为每一个生成树在图数据流中找到满足条件的匹配，从而得到和查询图满足编辑距离的所有最大相似性匹配。为了加快子图全匹配的速度，同样不需要枚举所有的生成树，而是采用一种边枚举边判断的方式进行匹配。只要判断出中的一个节点不满足当前映射的扩展，那么就立刻终止扩展到的下一层。

具体的匹配过程与算法3.6类似，分为两个递归部分，递归匹配部分和递归产生每个生成树所对应的QI-Sequence部分。同样递归匹配时对每个分区中的每个连通图分别进行处理，处理前先通过顶点标签集合进行过滤，当查询图的顶点标签集包含于该连通图的顶点标签集合时才继续向下执行匹配操作，否则将其过滤。过滤后，开始执行匹配操作。初始时，当前所在层数，中所存储的，只有最左边的路径，即只有按照规则4.1所生成的初始的生成树，同时中也只包含初始生成树所对应的QI-Sequence，即。

如果当前所在的层，那么就可以从查询图的顶点索引集当中随机选取一个顶点作为初始匹配顶点，否则所选则的顶点不仅要在查询图的顶点索引集当中，还要满足存在一条与该顶点相连的边与生成边相匹配，并且要记录当前扩展到该顶点时中所缺失的的条数。与算法3.6不同的是，我们要判断的并不仅仅只是当前生成树和查询图的编辑距离与扩展到该顶点时中所缺失的的条数之和与总的编辑距离阈值的，而是按照公式4.1计算扩展到该顶点后生成的图与查询图之间的边集距离与给定的总的编辑距离阈值之间的大小，如果此时的编辑距离小于，那么继续向下执行，否则不再继续扩展。然后继续按照算法3.6的方式向下执行，直到所有的生成树都完成匹配。注意，本章中我们输出结果时不仅输出满足编辑距离阈值的所有结果，同时输出每个结果与查询图之间的真实编辑距离的大小，以便于接下来的增量维护。

### 4.2.5子图全匹配的增量维护

接下来，当图数据流进行更新时，为了加快子图全匹配的速度，满足图数据流实时响应的特性，我们使用一种在结果上增量维护的方式进行近似匹配，在介绍具体方法之前，先定义以下公式。

随着时间的推移，时刻当前结果与真实结果之间的近似误差率定义如下：

（4.2）



其中为从一次全匹配之后到当前时刻这段时间之间的带权图数据流上的改变操作流中所包含的总操作个数，为中每个时间戳上与该时间戳上维护的匹配结果候选集中的子图相关的改变操作的总个数，其中，这里的“相关的改变操作”指更新的这条边至少包含于当前匹配结果候选集的一个子图当中。



明显地，随着时间的增加，误差率在不断增加，当不满足一定阈值时，我们重新进行一次完整的匹配操作。



（4.3）



为用户给定的误差率阈值，即当误差率大于用户所给阈值时，重新进行一次完整的匹配操作。



**定理4.1** 近似误差率，一定满足，其中为真实的误差率。



证明：由公式4.2，，其中为从一次完整匹配之后到当前时刻这段时间之间的带权图数据流的改变操作流中所包含的总操作个数，为中每个时间戳上与该时间戳上维护的匹配结果候选集中的子图相关的改变操作的总个数。我们的误差主要来源于图改变操作流中存在这样的一些更新操作，使得原本不在我们结果候选集中的某个子图成为了最终的匹配结果。表示更新操作流中从一次完整匹配之后到当前时刻之间，中每个时间戳上与该时间戳上维护的匹配结果候选集中的子图不相关的改变操作的总个数。 而这一部分的改变操作分为两类，一类是导致提前没有维护的子图成为了匹配结果，即主要误差来源，记总个数为，另一类是与匹配结果完全无关，记总个数为，则。所以，又因为，所以，即。



证毕。

由定理4.1可知只要我们计算的近似误差率，那么真实误差率一定满足，因为，且，所以。



由此本章增量更新的主要思想如下：在时刻首先按上述方式进行子图全匹配操作，返回满足编辑距离阈值的结果，但我们维护所有满足的匹配结果作为匹配结果候选集。当时刻到达一批更新时，先判断到目前为止累计的更新操作和的大小，计算此时的近似误差率是否小于等于用户给定的误差阈值，如果满足要求，则直接判断这批更新操作对候选集合的影响，返回此时候选集合中符合条件的匹配结果，否则按照上述方式重新进行子图全匹配操作，并以该时刻作为新的起始时刻继续向下判断，直至完成所有更新操作。



由观察可知，时刻上结果的近似准确率与近似成泊松分布，如下图所示：



图4.8泊松分布



Fig.4.8 Poisson distribution

由图4.8所示，阴影部分表示时刻下，当我们想要维护的满足的匹配结果作为匹配结果候选集中的时的近似误差率，明显地可知当取值越大时，在该时刻的近似误差率越小，近似准确率越高，则到该时刻为止的累计误差率越小，准确性越高；然而随着的逐渐增加，我们所要维护的匹配结果候选集也随之逐渐增大，维护代价也逐渐增高，因而导致时间代价也随之逐渐增大。因此，适当的选择一个的大小，使其既能有较高的准确性，又能满足图数据流实时响应的特性是非常重要的，本章对大小的选择将通过下文中的实验部分进行验证。



接下来我们详细介绍当图数据流上的一批更新到达时，如何直接在图数据流的已有结果上进行增量维护。本章同样不考虑孤立顶点的插入，并将图数据流上所有的改变均归结为边的插入和删除，也就是说图数据流中顶点的插入和删除均视为相关边的插入和删除，对于每个时间戳上图数据流的更新操作具体可以分为以下几种情形：

（1）更新的这条边的两个节点不均包含于（即至少有一个不包含于）我们所维护的匹配结果候选集中的任何子图当中；

（2）更新的这条边的两个节点均包含于我们所维护的匹配结果候选集中的一个子图当中，且该操作是一个插入操作；

（3）更新的这条边的两个节点均包含于我们所维护的匹配结果候选集中的一个子图当中，且该操作是一个删除操作；

对于第一种情况我们直接将这种更新过滤掉，不对它进行维护，并将这种边的个数记录在当中，随着时间的推移不断进行累加。



对于第二种情况，我们首先确定匹配结果候选集中包含这两个顶点的子图，然后分别在各个子图中插入这条边，如果该边原本不存在，则插入这条边，且权重变为1，如果该边已经存在，则权重直接加1。

对于最后一种情况，同样首先确定匹配结果候选集中包含这两个顶点的子图，然后分别在各个子图中对这条边进行删除操作，即将对应子图中的这条边的权重减1。注意：当因为删除一条边使得该候选子图不再连通时，我们不从候选集合中删除这个子图，因为之后的图改变操作很有可能重新插入这条边使其再次成为一个满足要求的候选结果。我们将其标记为false，表示这个图是非连通的，此时将这类图放入匹配结果候选集的最后，编辑距离记为无穷大，直到重新插入边使其变为连通图后再重新计算编辑距离并排序。

当对时刻下的所有图改变操作均进行处理之后，对于没有变化的候选图，直接将满足编辑距离阈值的候选图放入匹配结果集合中；对于有所该边的候选图，重新计算它与查询图之间的编辑距离，如果他们之间的编辑距离满足编辑距离阈值，则将其放入匹配结果集合，然后将匹配结果集合中的所有匹配结果按照编辑距离进行排序。



具体算法如下：

表4.4 带权图数据流上的增量更新

Table 4.4 Incremental updating over weighted graph

|  |
| --- |
| **算法4.4** |
| **输入：**：通过算法4.5得到的满足编辑距离的匹配结果候选集合，初  始时按的大小排序  ：查询图  ：时刻下的图改变操作  ：编辑距离阈值  ：用户自定义的误差率  **输出：**：满足编辑距离阈值的匹配结果结合，按的大小排序，其中为该匹配结果与查询图的编辑距离   1. For each ，初始化；  1. For each  1. If(&&)  1. 找到包含和的所有  1. For each 满足条件的  1. If()  1. If()  1. If()  1. ;  1. Else 2. ,;  1. Else 2. If()  1. ;  1. Else 2. ，;  1. If(不再连通)  1. ，;  1. End If 2. Else //当时  1. If()  1. If()  1. ;  1. Else 2. ，;  1. If(变为连通)  1. ;  1. 并且重新计算的值;  1. Else 2. If()  1. ;  1. Else 2. ，;  1. End For 2. End If 3. End For 4. End For 5. 按照重新排序;  1. ，且任何满足;  1. Return ; |

## 4.3实验

本节对上述提出的算法进行实验，通过在真实数据集和合成数据集中进行大量实验，并与静态图上的方法进行对比，证明了我们提出方法的正确性和有效性。

### 4.3.1实验环境

本节中的所有实验都是在Eclipse JAVA平台下完成的。所有算法均由JAVA语言实现，实验运行环境如下：

处理器：Inter(R) Core(TM) i7-2600 CPU @3.40GHz，3.40GHz。

内存(RAM)：8.00GB。

操作系统：64位Windows操作系统。

### 4.3.2数据集

本章分别使用真实数据集和合成数据集来进行实验：

真实数据集1：DBLP，本章使用的DBLP与第三章相同，只是在其基础上随机为每条边产生权重，且随机产生的更新是复杂更新。

真实数据集2：同样使用HPIN，并随机为每条边产生一个权重，且随机产生的更新为复杂更新。

合成数据集：按照第三章中的方法产生大小为，，（即标签种类为50种）的合成数据集，并为每条边随机分配权重。

查询图：在第三章中查询图的基础上，为每条边分配权重。

本章中所执行的算法如下：

SimMatch-Weighted：带权静态图上的子图全匹配算法，即不带有分区策略的子图全匹配算法；

NNJuegment- Weighted：带有分区策略的子图全匹配方法，即本文提出的带权图数据流上的子图全匹配方法。

IncMaintain：本章最后提出的近似的图数据流上增量维护算法。

### 4.3.3运行结果

## 4.4总结

本章主要对带权图数据流上增量子图相似性全匹配问题进行了研究，主要针对的是图数据流上的复杂更新。首先在最近邻分区的基础上提出了基于权重的最近邻分区，通过权重的判断对图数据流进一步进行剪枝；然后在上一章算法的基础上进行了改进，提出了适用于带权图上创建生成树以及子图全匹配的方法；接下来通过维护匹配结果候选集，在每次更新时直接在候选集中进行更新，并提出了近似误差率的概念，以保障匹配结果的准确性。最后通过在真实数据集和合成数据集上进行大量的实验，证明了本章提出算法的正确性和有效性。

# 第5章 结论

## 5.1本文的主要贡献和结论

近年来，图数据广泛应用于各种实际应用（如通信网络，模式识别，化学/生物信息等）当中，图数据上的各种问题，如子图匹配，TOP-K查询，最短路径，k近邻等都已得到了广泛而深入的研究。然而随着科学技术的不断发展，图数据不再拘泥于传统的静态形式，而是以一种流的形式不断更新变化。因此越来越多地关注集中到图数据流当中，图数据流上的子图匹配技术作为图数据流上许多其他操作的基石广泛应用与社交网络，生物医学，网络安全等领域当中。然而由于图固有的复杂结构加上图数据流动态更新的特性，使得目前还没有研究集中于图数据流上的增量子图匹配问题上。除此之外，目前对于图数据流上的研究大多集中于简单更新上，而现实当中，图数据流上的复杂更新已成为一个越来越常见的问题。因此，图数据流上增量子图匹配问题有着广泛的应用前景和深远的现实意义，那么，提出一种有效的方式解决图数据流上增量子图匹配问题显得尤为重要。

针对图数据流的现实意义和应用前景，本文基于不同的更新方式提出了图数据流上增量相似性子图匹配问题，并对其进行了研究。

（1）针对于图数据流上的简单更新提出了图数据流上增量子图相似性匹配问题。本文首先提出了一种基于最近邻的分区策略，这种分区策略是一种带有过滤的分区，首先通过查询图的结构对图数据流进行结构剪枝，再通过查询图最近邻的判断适当增加一些虚边，使得之后的更新过程中被合并在一起可能性较大的顶点在同一分区当中，以减少之后对分区进行维护的代价。然后通过为查询图创建生成树集合将原本的图-图匹配转化为树-图匹配，打打提高了子图匹配速度。然后，在每次更新后通过局部更新的方式对原有结果进行增量维护，进一步提高了子图匹配的效率。最后用实验证明了本文提出方法的正确性，并与传统静态方法进行对比，进一步证明方法的有效性。

（2）针对于图数据流上的复杂更新提出了带权图数据流上增量子图相似性匹配问题。为了进一步满足图数据流实时响应的特性，本章提出了是一种近似的方法。首先在最近邻分区的基础上进一步提出了基于权重的最近邻分区，再将初始的图数据流进行结构剪枝进行过滤之后，再通过权重计算编辑距离，进一步进行剪枝。之后将基本的创建生成树的算法以及子图全匹配算法进行改进，使其能够适用于带权图数据流上。然后，通过对的匹配结果候选集进行维护，同时又基于近似误差率，在每次更新到达时直接在匹配结果候选集中进行增量的维护，并当不满足用户所给定的误差大小时重新进行匹配。这样既保证了实时响应的特性，有保证了匹配的准确性。最后通过实验证明了方法的有效性。



## 5.2进一步的工作

随着数据种类的不断发展，图数据流已经得到了越来越多专家学者们的关注，基于其深远的现实意义和广泛的应用前景本文对图数据流上增量子图相似性匹配技术进行了研究，本文主要针对图数据流上的简单更新和复杂更新进行了研究，本文所使用的分区策略、生成树的创建、最大子图全匹配以及增量更新的思想为许多其他研究打下了坚实的基础。在未来的工作中，将主要围绕以下几个方面进行进一步的研究：

（1）本文研究的图数据流上增量子图相似性匹配问题时对于编辑距离的处理只考虑了少边的情况，即匹配结果只能比查询图少边不能多边，同时我们也要求匹配结果的顶点标签必须与查询图完全匹配，然而有时多边的情况以及顶点标签不匹配的情况依然能够满足结果的需要，因此在未来的工作中要对编辑距离和子图匹配的概念进行拓展，使最后的匹配结果更加完善。

（2）本文在对带权图进行研究时，图数据流的更新只包括正常的插入和删除操作，在未来的工作中，将会在更新的过程中加入滑动窗口的概念，使得图数据流的上的权重，还会随着时间的增加而改变，这样我们就可以查询特定时间窗口内的匹配结果。

（3）本文所研究的图数据流都是确定图，然而随着数据多样化的发展，在一些情形中也存在不确定的图数据流，即边上带概率的图数据流，如交通网络，网络安全。因此，下一步工作将研究不确定图数据流上增量子图相似性匹配问题。

（4）本文所有研究的内容是非并行的，在接下来的工作中将会在分区时加入通信代价的考虑，使本文中的模型能够用于并行条件下以处理一下更大的图数据流。

# 参考文献

1. Wu Y, Yang S, Yan X. Ontology-based subgraph querying[C]//Data Engineering (ICDE), 2013 IEEE 29th International Conference on. IEEE, 2013: 697-708.
2. Zhu G, Lin X, Zhu K, et al. TreeSpan: efficiently computing similarity all-matching[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012: 529-540.
3. Zhang S, Yang J, Jin W. SAPPER: subgraph indexing and approximate matching in large graphs [J]. Proceedings of the VLDB Endowment, 2010, 3(1-2): 1185-1194.
4. Wang C, Chen L. Continuous subgraph pattern search over graph streams[C]//Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. IEEE, 2009: 393-404.
5. Chen C, Yan X, Yu P S, et al. Towards graph containment search and indexing[C]//Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007: 926-937.
6. Khan A, Wu Y, Aggarwal C C, et al. NeMa: Fast graph search with label similarity[J]. Proceedings of the VLDB Endowment, 2013, 6(3): 181-192.
7. Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs[C]//Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012: 1222-1230.
8. Angel A, Sarkas N, Koudas N, et al. Dense subgraph maintenance under streaming edge weight updates for real-time story identification[J]. Proceedings of the VLDB Endowment, 2012, 5(6): 574-585.
9. Bahmani B, Kumar R, Vassilvitskii S. Densest subgraph in streaming and mapreduce[J]. Proceedings of the VLDB Endowment, 2012, 5(5): 454-465.
10. Shasha D, Wang J T L, Giugno R. Algorithmics and applications of tree and graph searching[C]//Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2002: 39-52.
11. Yan X, Yu P S, Han J. Graph indexing: a frequent structure-based approach[C]//Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM, 2004: 335-346.
12. Chen C, Yan X, Yu P S, et al. Towards graph containment search and indexing[C]//Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007: 926-937.
13. Zhao P, Aggarwal C C, Wang M. gSketch: on query estimation in graph streams[J]. Proceedings of the VLDB Endowment, 2011, 5(3): 193-204.
14. Bahmani B, Kumar R, Vassilvitskii S. Densest subgraph in streaming and mapreduce[J]. Proceedings of the VLDB Endowment, 2012, 5(5): 454-465.
15. Sarma A D, Gollapudi S, Panigrahy R. Estimating pagerank on graph streams[J]. Journal of the ACM (JACM), 2011, 58(3): 13.
16. Yu W, Lin X, Zhang W. Fast incremental SimRank on link-evolving graphs[C]//Data Engineering (ICDE), 2014 IEEE 30th International Conference on. IEEE, 2014: 304-315.
17. Trißl S, Leser U. Fast and practical indexing and querying of very large graphs[C]//Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007: 845-856.
18. Wang H, He H, Yang J, et al. Dual labeling: Answering graph reachability queries in constant time[C]//Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on. IEEE, 2006: 75-75.
19. J. Cheng, Y. Ke, W. Ng, and A. Lu. Fg-index: towards verification-free query processing on graph databases. In SIGMOD, 2007.
20. Shang H, Zhang Y, Lin X, et al. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism[J]. Proceedings of the VLDB Endowment, 2008, 1(1): 364-375.
21. Zhao P, Yu J X, Yu P S. Graph indexing: tree+ delta<= graph[C]//Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007: 938-949.
22. Shang H, Zhang Y, Lin X, et al. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism[J]. Proceedings of the VLDB Endowment, 2008, 1(1): 364-375.
23. Cormode G, Muthukrishnan S. An improved data stream summary: the count-min sketch and its applications[J]. Journal of Algorithms, 2005, 55(1): 58-75.
24. Tian Y, Patel J M. Tale: A tool for approximate large graph matching[C]//Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE, 2008: 963-972.
25. Zhang S, Li S, Yang J. GADDI: distance index based subgraph matching in biological networks[C]//Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM, 2009: 192-203.
26. H. He and A. K. Singh. Closure-tree: An index structure for graph queries. In ICDE, 2006.
27. Akiba T, Iwata Y, Yoshida Y. Fast exact shortest-path distance queries on large networks by pruned landmark labeling[C]//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013: 349-360.
28. Gubichev A, Bedathur S, Seufert S, et al. Fast and accurate estimation of shortest paths in large graphs[C]//Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010: 499-508.
29. Ghorbani M, Hosseinzadeh M A, Diudea M V. ON TOPOLOGICAL POLYNOMIALS OF WEIGHTED GRAPHS[J]. Studia Universitatis Babes-Bolyai, Chemia, 2012, 57(4). .
30. Shang H, Lin X, Zhang Y, et al. Connected substructure similarity search[C]//Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010: 903-914.
31. Michael R G, David S J. Computers and intractability: a guide to the theory of NP-completeness [J]. WH Freeman & Co., San Francisco, 1979.
32. He H, Wang H, Yang J, et al. BLINKS: ranked keyword searches on graphs[C]//Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007: 305-316.
33. Bhalotia G, Hulgeri A, Nakhe C, et al. Keyword searching and browsing in databases using BANKS[C]//Data Engineering, 2002. Proceedings. 18th International Conference on. IEEE, 2002: 431-440.
34. Zou L, Chen L, Yu J X, et al. A novel spectral coding in a large graph database[C]//Proceedings of the 11th international conference on Extending database technology: Advances in database technology. ACM, 2008: 181-192.
35. Umeyama S. An eigendecomposition approach to weighted graph matching problems[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1988, 10(5): 695-703.
36. Almohamad H A, Duffuaa S O. A linear programming approach for the weighted graph matching problem[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1993, 15(5): 522-525.
37. Gold S, Rangarajan A. A graduated assignment algorithm for graph matching[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1996, 18(4): 377-388.
38. L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. Pattern Anal. Mach. Intell., 26:1367–1372, 2004
39. Biswas S, Morris R. ExOR: opportunistic multi-hop routing for wireless networks[A]. ACM SIGCOMM Computer Communication Review[C], 2005, 35(4): 133-144.
40. Zhang S, Li S, Yang J. GADDI: distance index based subgraph matching in biological networks[C]. Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 2009, 192-203.

# 致谢

时间荏苒，岁月如歌，两年的硕士生活即将画上句点。还记得决定考研时的踌躇满志，还记得考研期间的早出晚归，还记踏入校园时的激动不已，一切的一切都仿佛发生在昨天，依然那么历历在目。两年的时间说长不长，说短也不短，长到让我经历了很多同时也学习到了很多，短到让我觉得仿佛才刚踏入校园。这一路走来，在大家的帮助下我不断成长，不断收获。在此，我要向所有关心帮助过我的老师同学和朋友们送上我最诚挚的谢意！

首先我要感谢的人是我的导师谷峪老师，非常荣幸能够成为您的学生。您在工作时的认真勤奋、精益求精，在小组学习时的激烈讨论、博学多识，在娱乐时的风趣幽默都让我们感受到您严谨的态度、敏捷的思维、高瞻远瞩的学术洞察力和乐观积极的生活态度，这些对我们以后的生活和工作都产生了深远的影响，您不但是我们学习上的导师，更是我们生活上的导师，您教会我们的绝对不仅仅是学习的方法和态度，更多的是做人的道理，人生的感悟，我们将永远的铭记于心并表示由衷的感谢！

接下来衷心的感谢于戈老师、张天成老师、赵志滨老师、李芳芳老师，感谢您们和我们一起度过的每一次学术会议，每一次的激烈讨论都让我们受益匪浅。还有林树宽老师、申德荣老师、王大玲老师、鲍玉斌老师、张伟老师、张莉老师、王斌老师、杨晓春老师、朱靖波老师、董晓梅老师、李晓华老师、张一飞老师、寇月老师、冯时老师、聂铁铮老师、李传文老师等软件所里的每一位老师。感谢您们在学术上热情的指导和帮助以及生活上细致的关心。

我还要感谢李淼师姐、王志刚师兄、王宁师姐、庞俊师兄、马茜师姐、王璐璐师姐、王彪师兄、感谢你们在生活上和学习上都像哥哥姐姐一样对我无微不至的关心和帮助。感谢我的同学张竞宇、刘冠利、胡宇、徐宏斌，徐硕，和你们一起学习和生活的日子里，互相帮助，互相进步，让我感受到了集体的温暖和智慧，使我更加深刻的体会到了研究生生活的快乐。感谢郭艳丹师弟、李东洋师弟、何家驹师弟、李天义师妹、郑建国师弟，虽然相处的时间并不长，但你们刻苦努力的精神让我知道作为师姐要更加严格的要求自己、做你们的好榜样。感谢你们让我在这里感受到亲人般的关怀和帮助，让我在这学习和成长中少了份孤独与忧伤，多了份陪伴与快乐。

感谢和我朝夕相处的室友王佳晗、张春杰、张榜，我们相互的帮助，相互支持，共同度过两年的硕士生活。同时感谢郑换霞、张桂圆、郭彩华同学，和你们的朝夕相处使我们的生活变得更加丰富多彩。希望姐妹们朝着自己的梦想努力，让生活的每一天都充满阳光。

我要深深的感谢我的父母，感谢你们对我的支持与鼓励，感谢你们在我成长的过程中所付出一切，因为您无私的爱和奉献使我更有信心与动力迎接未来的生活，谢谢你们。

最后我要再次向关心帮助支持我的父母、老师和朋友们说一声谢谢，感谢有你们的陪伴，使我拥有美好的生活，谢谢你们。

# 攻读硕士学位期间的项目情况

**攻读硕士学位期间参加的科研项目有：**

国家自然科学基金“云计算环境下基于BSP模型的大规模图数据处理技术”（编号：61272179）项目（2013.01-2016.01）。