分类号：　　单位代码：10140

密　级：公开　　学　　号：4032032345

遼寧大學

硕 士 学 位 论 文

中文题目：基于多特征因子和分布式图划分的个性化 PageRank 算法研究

Research on Personalized PageRank Algorithm based on Multiple Feature Factors and Distributed Graph Partitioning

英文题目：

论文作者：张子扬

指导教师：王嵘冰　副教授

专　　业：计算机软件与理论

完成时间：二〇二三年五月

申请辽宁大学硕士学位论文

基于多特征因子和分布式图划分的个性化 PageRank 算法研究

Research on Personalized PageRank Algorithm based on Multiple Feature Factors and Distributed Graph Partitioning

作　　者：张子扬

指导教师：王嵘冰　副教授

专　　业：计算机软件与理论

答辩日期：2023 年 5 月 16 日

二〇二三年五月·中国辽宁

摘 要

随着互联网技术时代的全面快速发展，各类新兴网络应用平台（社交网络、个性化推荐）涌入人们的日常生活中，使得网络数据量暴增，这些海量数据通常来说都蕴含着重要且有价值的信息。数据带给人们的价值无法估量，针对大型网络数据结构图计算问题的研究已经成为一块热门的研究领域，而互联网的信息越来越丰富，数据量也随指数上升。因此，在现今海量而复杂的信息数据中，高效准确地获取有价值的信息变得尤为重要。针对这一问题，本文从面向大规模网络图个性化 PageRank 算法的排序查询精确度问题与计算效率问题两个方面展开研究，具体包括如下两个部分：

（1）针对个性化 PageRank 算法在大规模网络图排序查询中存在查询结果精确度及个性化搜索时效性不高的问题，本文提出了一种基于多特征因子的个性化 PageRank 查询优化算法（Improved Enhanced-RatioRank，IER），该算法主要通过主题漂移优化、内容相关度增强和内容时效性平衡三个模块对个性化 PageRank 算法进行查询优化，在主题漂移优化模块中本文首先引入了权重因子，权重因子可以解决主题漂移问题，使其用户根据主题关键词查询时尽可能减少垃圾网页的情况。其次针对权重因子进行了优化，通过设置α和β权重比调节因子，利用调节权重比因子α和β来调控链入权重及链出权重的影响度，更好的解决 PPR 值下沉问题；在内容相关度增强模块中本文利用频数因子改善用户查准率，使用户搜索目标需求与反馈结果的匹配度更高，提高查询内容相关度；内容时效性平衡模块本文主要利用时间有效性因子来提高查询内容的时效性，使其查询时效性高的网页节点上升，时效性低的网页节点下沉。最后本文采用了五个大小不同数据集将 IER 算法与 Enhanced-RatioRank、Time-PageRank 等同类算法进行实验对比，实验结果表明 IER 算法不仅可以使搜索结果更偏向用户查询需求，而且还提高了查询内容的精确度，极大地提高了算法的查准率。

（2）针对现在主流的分布式图划分算法在进行图划分过程中产生的子图之间的通信频率高导致计算效率不高的问题，本文提出了一种基于分布式图划分的个性化 PageRank 改进算法（Boruvka Adjacent Edge Structure DSP，BAESD），本文首先提出了一种基于 Boruvka 算法的分布式图划分算法，完成局部子图的构建与平衡化，将大规模网络图划分为多个局部平衡子图，使个性化 PageRank 算法的迭代计算转移到局部子图数据中；其次提出了一种相邻边缘结构（Adjacent Edge Structure，AES），AES 结构主要从两个方面对算法进行优化：一方面是提高图划分效率，尽可能减少图数据处理的时间；另一方面是该结构可以减小局部

分区子图与子图之间的依赖关系密度；最后，本文利用 DSP 模型（delta-stepping synchronous parallel）来加速个性化 PageRank 算法的迭代计算，从而进一步提高个性化 PageRank 算法在大规模图上的计算效率。本文算法在四个不同大小数据集上与 PM、RA 和 TDA 等主流算法进行实验对比，实验结果表明本文所提算法极大提高了个性化 PageRank 算法在大规模网络图上的计算效率。

关键词：个性化 PageRank 算法，多特征因子，分布式图划分，相邻边缘结构

ABSTRACT

With the comprehensive and rapid development of the Internet technology era, various emerging network application platforms (social networks, personalized recommendations) have flooded into people's daily lives, making the amount of network data skyrocket, and these massive data usually contain important and valuable information. The value of data to people is immeasurable, and the research on large network data structure graph computation problem has become a popular research area, while the information on the Internet is becoming more and more abundant and the amount of data is increasing exponentially. Therefore, in today's massive and complex information data, it becomes especially important to obtain valuable information efficiently and accurately. To address this problem, this paper starts the research from two aspects, namely, the accuracy problem of ranking query and the computational efficiency problem of personalized PageRank algorithm for large-scale network graphs, which includes the following two parts:

(1) For the personalized PageRank algorithm in large-scale network graph sorting query has the problems of query result accuracy and personalized search timeliness, this paper proposes a personalized PageRank query optimization algorithm (Improved Enhanced-RatioRank, IER) based on multiple feature factors, which mainly optimizes the personalized PageRank algorithm through three modules: topic drift optimization, content relevance enhancement and content timeliness balance, the algorithm is mainly optimized by three modules of personalized PageRank algorithm: topic drift optimization, content relevance enhancement and content timeliness balance. In the topic drift optimization module, this paper first introduces the weight factor, which can solve the topic drift problem and minimize the spam web pages when users query according to the topic keywords. Secondly, we optimize the weight factor by setting alpha and beta weight ratio adjustment factors, and use the adjustment weight ratio factors alpha and beta to regulate the influence degree of chain-in weight and chain-out weight, so as to better solve the problem of sinking PPR value; in the content relevance enhancement module, this paper uses the frequency factor to improve the user search accuracy rate, so that the user search target demand and feedback In the content relevance enhancement module, this paper uses the frequency factor to improve the user search accuracy, so that the user search target needs and feedback results match

better and improve the relevance of the query content; the content timeliness balance module, this paper mainly uses the time validity factor to improve the timeliness of the query content, so that the web nodes with high timeliness rise and those with low timeliness sink. Finally, this paper adopts five data sets of different sizes to compare the IER algorithm with similar algorithms such as Enhanced-RatioRank and Time-PageRank. The experimental results show that the IER algorithm can not only make the search results more biased to the user's query needs, but also improve the accuracy of the query content, which greatly improves the accuracy of the algorithm.

(2) Aiming at the problem of high frequency of communication between subgraphs generated in the process of graph partitioning by the mainstream distributed graph partitioning algorithms nowadays leading to low computational efficiency, this paper proposes a personalized PageRank improvement algorithm based on distributed graph partitioning (Boruvka Adjacent Edge Structure DSP, BAESD). This paper first proposes a distributed graph partitioning algorithm based on Boruvka algorithm, completing the construction and balancing of local subgraphs, dividing the large-scale network graph into multiple local balancing subgraphs, so that the iterative computation of the personalized PageRank algorithm is transferred to the local subgraph data; and secondly, proposing an Adjacent Edge Structure (AES), AES structure The AES structure is mainly optimized from two aspects of the algorithm: on the one hand, it improves the efficiency of graph partitioning and reduces the time of graph data processing as much as possible; on the other hand, the structure can reduce the density of dependencies between locally partitioned subgraphs and subgraphs; and finally, this paper uses the DSP model (delta-stepping synchronous parallel) to accelerate the iterative computation of the personalized PageRank algorithm, so as to further improve the personalized PageRank algorithm on large-scale This paper uses the DSP model to accelerate the iterative computation of the personalized PageRank algorithm, thus further improving the computational efficiency of the personalized PageRank algorithm on large-scale graphs. The algorithms in this paper are experimentally compared with mainstream algorithms such as PM, RA and TDA on four different size datasets. The experimental results show that the proposed algorithm greatly improves the computational efficiency of the personalized PageRank algorithm on large-scale network graphs.

Key Words: personalized PageRank algorithm, multiple feature factors, distributed graph partitioning, adjacent edge structure

目 录

图 表 目 录

图目录

表目录

第 1 章 绪论

1.1 研究的背景及意义

在当今互联网时代，随着科学技术的快速发展，各种现代网络应用平台如雨后春笋般涌入到人们的生活中。这些平台赋予了人们可以通过多渠道获取信息的能力，让信息成为了人们日常生活中不可或缺的一部分。根据中国互联网络信息中心最新发布的一份数据报告显示：中国互联网基础资源日益丰富，互联网用户规模及普及率在不断地增长，国家网民人数已经超过 10 亿人。随着人们生活节奏的不断加快，互联网给人们带来了很多便利，可以体现在工作、交流、学习、以及娱乐等多个方面。然而，随着互联网信息的日益丰富，网络数据量也急剧上升，这会导致用户利用网络获取有价值的信息变得异常困难，甚至效率也很低下。事实上，人们已经处于信息过载的时代，用户在众多的搜索结果中往往会遇到信息的重复、冗余或者不相关的问题。因此，从纷繁复杂的互联网数据库中高效准确的获取有价值信息，这是一个尤为关键的问题。

搜索引擎[1]是一种信息检索工具，它可以提高用户在互联网搜索及获取信息数据的效率，拓宽了用户获取知识的渠道，也提高了用户的知识面。它应运而生的初衷是为了让用户能够获取有价值的信息，并在互联网中心获取更加全面的信息数据。通过使用搜索引擎，用户可以更加及时、全面地获取信息，从而避免信息失去其有效性。搜索引擎的工作流程可以分为三个步骤，分别是搜集网页数据、建立索引库、搜索与排序。首先，在搜集网页数据方面，搜索引擎从互联网中心搜集海量网页信息数据，并对这些信息进行获取和组织。其次，搜索引擎建立索引库，将这些信息进行归类和存储。最后，当用户输入关键词进行搜索时，搜索引擎会根据这些关键词定位目标网页，然后根据匹配的算法对目标结果排序完成后将查询结果反馈给用户。由于排序是搜索引擎的关键要素，因为它能够确保搜索结果的准确性和权威性。PageRank 算法是当前使用最为广泛的搜索引擎网页排序算法[2]。它是由 Larry Page 和 Sergey Brin 提出的一种基于网页链接结构的网页排序算法。该算法首先通过迭代计算各个网页节点的 PageRank 值，而某个网页节点的 PageRank 值可以用来评价该网页节点相对于其他节点的重要性[3]，并根据一定的权重比，分配好每个网页节点的 PageRank 值后传递到目标网页。这一过程能够帮助搜索引擎确定搜索页面的 PageRank 值，从而确保搜索结果的准确性和排名权威性。总之，搜索引擎的出现和发展为用户提供了更加方便、高效、准确的信息检索和获取服务，同时也推动了网络与信息技术等相关领域的不断创

新和发展。PageRank 算法作为一种重要的网页排序算法，对于搜索引擎的准确性、权威性与用户体验都有着不可忽视的作用。

然而，随着互联网技术的更新迭代，网络数据量急剧增加，用户通过搜索引擎在网上搜索信息时，由于用户想要通过互联网获取更丰富复杂的信息数据时，搜索引擎返回的查询结果也存在冗余，而且查询的内容也不是很准确。因此，传统的搜索引擎已经不能够为用户提供高效、精准的查询结果，它主要存在以下方面的问题：

（1）当用户的查询频率在不断上升且获取的信息数据成指数增长时，加上查询结果与用户目标数据信息不符合。尽管传统搜索引擎的结果非常依赖于用户给定的关键字，但是早期的排序算法只可以在静态网页结构中获取数据，这样会严重降低最终返回用户的结果内容和主题相关度。

（2）由于网络数据量急剧增加，用户的搜索频次和搜索信息数据量也会暴增，这样会严重加重搜索引擎的响应负担，使其用户的搜索查询效率低下，因此，本文需要对搜索引擎提出更高的要求，即改善查询效率。

（3）网页信息数据时效性不高，由于现代互联网信息数据更新迭代快，用户想要在第一时间内了解并关注到最新的热点信息，所以对实时信息数据的追求也越来越高，因此也需要针对网页信息数据时效性做进一步的优化。

综上所述，搜索引擎确实存在一些问题，因此排序算法的改进[4]是提高搜索引擎性能的核心要素，这对于 PageRank 算法的改进研究具有不可或缺的现实意义。

近年来，大量的国内外学者对大型网络数据结构图计算问题进行了深入研究。在网络搜索引擎中，PageRank 算法被广泛应用，其主要作用是评价某些网页相对于其他网页的重要程度。该算法最大的优势在于可以在大型网络中进行高效计算。随着现代社交网络应用的个性化发展，需要一种类似的个性化网络结构度量方法。因此，个性化 PageRank（Personalized PageRank，PPR）算法应运而生，它是 PageRank 算法的特殊应用[5]。个性化 PageRank 算法可以被认为是以自我为中心的个性化网络结构度量算法。个性化 PageRank 可以看作是一组特定节点集根据一定的随机游走准则来度量访问其他节点的概率，图中节点的概率值（PPR 值）可以用来评价该网页节点与其他节点的相对亲密度。例如，在社交网络系统中，某个社交用户的 PPR 值是相对的，该用户相对于亲密度一般的其他用户 PPR 值比较小，但相对于亲密度高的好友用户 PPR 值会比较大，个性化 PageRank 算法可以帮助搜索引擎实现个性化搜索,该算法主要根据两个指标来评估用户的搜索意图，其中一个指标是用户的知识背景，另一个指标是用户自身属性，最后还将排序算法得到的结果重新筛选排序，进一步提高查询结果的准确度。自 PPR 提

出以来，PPR 在网络中有着广泛的应用。如好友推荐[6]，网页搜索[7]，社区检测[8]，链路分析[9]等。在现代互联网应用中，个性化 PageRank 算法不仅具有实际的应用价值，而且还具有广阔的发展前景，与传统的 PageRank 算法不同，个性化 PageRank 算法不仅考虑网页节点与节点之间的相对亲密度，而且还要考虑用户的查询偏好，所以个性化 PageRank 算法具有内容查询与用户需求高度匹配，当用户根据自己的需求查询相关内容时，该算法不仅极大地提高了查询结果准确度，而且更加符合用户实际需求。

1.2 国内外研究现状

在现代信息化网络时代中，个性化 PageRank 算法不仅是一种主流的图计算算法，而且在应用方面也十分广泛。该算法通过网络图节点之间的链接关系来计算出节点之间的关联度，相当于该节点的个性化 PageRank 值。个性化 PageRank 值可以定义为由一组受约束条件的节点集合根据随机游走模型来度量访问其他节点的概率。由于该算法是一种个性化网络结构度量算法，因此它的应用场景很多，特别是在可以用网络图结构表示的领域。由于传统的 PageRank 算法计算单一网页的排名时需要用到所有节点的信息，在大规模图中进行计算会导致计算时间成本巨大，而个性化 PageRank 算法在自身的计算之外还需要考虑一些受约束条件的特殊节点。因此，如何实现高效的个性化 PageRank 算法计算是学者们的一个研究焦点。另外，随着用户量的增加以及个性化服务的普及，当用户根据自己的偏好查询相关内容时，用户网络节点的 PPR 值会随着用户的查询偏好不断变化。在这种背景下，如何提高个性化 PageRank 算法的查询准确度，则属于另一个需要攻克的难题。当网络规模图比较大时，直接在大规模网络图上进行 PPR 的迭代计算，这样不仅时间成本高，而且可操作性也很差。

近年来，国内外研究人员对个性化 PageRank 算法的研究主要分为两类：一类是建立在蒙特卡洛方法的近似算法的基础上，该类算法基于一个随机游走模型，首先设置一个阈值，对节点的 PPR 值进行近似估计，尽可能的降低每个节点 PPR 值的计算时间复杂度，从而提高 PPR 的计算效率。在近似算法中，Fujiwara Y 等人[10]提出了一种 Castanet 方法，该算法利用迭代计算出两个阈值，一个上阈值，另一个阈值，之后根据夹逼准则估算出结果值，然后动态修剪冗余的链接网络结构，从而降低节点迭代计算的时间复杂度，提高了搜索效率，但由于一个合理准确的界限值难以确定，所以也会严重影响算法效率。Lofgren 等人[11]提出了一种基于双向 PPR 估计器的算法，该算法不需要迭代通过所有候选结果的集合，只需要对候选对象的 PPR 进行抽样来识别最相关的结果，然而抽样过程很有可能

会降低样本值的精确度，也会对 PPR 值的精确计算产生一定的影响。Wang Sibo 等人[12]利用一种索引技术来提高 PPR 的计算效率，该算法的主要思想是对经常参与 PPR 处理的选定枢纽节点进行预计算和索引辅助信息，从而提高查询效率和查询准确度。李兰英等人[13]提出了一种在 MapReduce 平台上 topK-Rank 算法，该算法通过在迭代计算中清洗掉冗余的网络节点结构动态构建子图，在子图上近似估算边界值，该算法在一定程度上提高了算法效率，但图规模大小容易受限于主存容量。贾瑞娜等人[4]提出了一种可达子图 RA 算法，该算法通过给定源节点，利用在线搜索方法预先求出源节点的可达子图，有效降低了 PPR 在大图上的计算规模。

另一类是精确算法，刘齐等人[14]提出了一种改进的 PageRank 算法，该算法基于主题相似度原则，首先对相似度进行计算得出每个节点的分值，然后根据每个节点的评分来赋予不同的影响权重，从而有效减少主题漂移现象，提高查询结果的准确度，但如果数据规模较大时，算法的效率也会受到影响。Nykl 等人[15]在个性化 PageRank 算法添加了学术论文的显著性指标，对科学论文作者的影响力进行排名。曹姗姗等人[16]在网页节点链接结构模型的基础上，对 Bias PageRank 算法进行改进得到优化算法，该算法可以根据用户反馈的信息数据对用户查询需求进行综合分析，在一定程度上提高了算法的排序质量以及用户信息查询的满意度。臧思思等人[17]提出的 Time-PageRank 算法在学术社交网络中通过计算作者的 PPR 值对作者的学术贡献度进行排名，从而得到作者的影响力。杨红果等人[18]提出一种连通分量子图结构的迭代算法，该算法不需要直接计算在原图规模的 PPR 值，而是通过简单的信息通信使其在子图集合上计算 PPR。然而该算法在强连通子图数量过多时，子图与子图之间的通信量会迅速增加，也可能会导致资源分布不均匀的情况。陈星玎等人[19]提出了一种多步幂法修正的广义二级分裂迭代法，该算法在外迭代之前先利用幂法进行修正得到多步的广义二级分裂迭代法，然后通过引入新参数来加速迭代收敛，该算法在一定程度上可以降低计算成本，然而选取合适的参数是十分困难的。在精确算法中，PM 算法[20-22]利用多次迭代来计算 PPR 向量，在每次迭代计算过程中，都会更新一个 PPR 向量值,在一定约束条件收敛后确定最后的 PPR 值。研究人员针对该算法进行了优化，主要分为两类：串行算法[23-24]和分布式处理算法[25-29]。串行算法是基于原图数据的一种迭代计算方法，当原图的数据规模体系很庞大时，内存只能够计算机一小部分的操作结果和数据，这样会使 I/O 负载过重，算法的复杂度也会变得很高。而分布式处理算法主要是将大规模网络图划分为多个独立的子图，然后在每个计算结点上迭代计算出各个子图的 PPR 值，直至所有结果收敛。分布式图划分算法的主流算法有 TDA 算法与 DA 算法[18]，TDA 算法是一种传统的分布式算法，该

算法划分精确度比较高，但是划分时间也是随着迭代次数成正比；DA 算法首先将 PPR 在大规模图中的计算利用分布式计算模型划分为多个子图，然后并行化处理PPR在每个子图的计算。虽然分布式算法可以在多个计算结点上并行计算，在很大程度上可以提高计算效率，但是在进行分布式图划分的过程中产生了大量的子图，子图和子图之间依赖性错综复杂，导致子图之间的需要高频次的通信，严重影响了算法的计算效率。

综上所述，目前针对个性化 PageRank 算法的研究中或多或少存在一些问题，比如在基于蒙特卡洛的近似估计算法中难以确定一个合理的上下界限值和参数，当图规模数据比较大时，用户连接节点的数量相对较少，而且这些连接也分散在整个网络中，这就导致了个性化 PageRank 算法需要访问大量的稀疏数据，而这在实际应用中会导致计算资源的浪费和运行速度的缓慢。算法的精确度也会受到一定的损失。在精确算法中，当图规模数据大到一定量级的时候，内存容量受限以及在大规模图划分子图之间的通信量问题，这将会严重影响算法的计算效率。

1.3 论文的主要研究内容

本文从面向大规模网络图中个性化 PageRank 算法的排序查询精确度问题与计算效率问题两个方面展开深入研究，因为算法的查询精确度和计算效率是个性化 PageRank 算法的两个核心部分，它们是一个并列关系，在数据处理的过程中，提高算法的查询精确度可以更好的利用数据特征进行优化，从而优化算法的计算操作，进一步提高算法的计算效率，它们对于 PPR 算法的优化作用是相辅相成的。

在个性化 PageRank 算法的排序查询问题研究中，仍然面临着一些问题，比如随着现代的社交网络应用向个性化发展，网络规模图数量急剧增加，个性化 PageRank 算法的查询结果精确度及内容相关度不够高，且个性化搜索的时效性也不够好。综上所述，针对个性化 PageRank 算法在大规模网络图排序查询中存在查询结果精确度及个性化搜索时效性不高的问题，本文提出了一种基于多特征因子的个性化 PageRank 查询优化算法（Improved Enhanced-RatioRank，IER），该算法主要通过主题漂移优化、内容相关度增强和内容时效性平衡三个模块对个性化 PageRank 算法进行查询优化，在主题漂移优化模块中本文首先引入了权重因子，权重因子可以解决主题漂移问题，使其用户根据主题关键词查询时尽可能减少垃圾网页的情况。其次针对权重因子进行了优化，通过设置α和β权重比调节因子，利用调节权重比因子α和β来调控链入权重及链出权重的影响度，更好的解决 PPR 值下沉问题；在内容相关度增强模块中本文利用频数因子改善用户查准

率，使用户搜索目标需求与反馈结果的匹配度更高，提高查询内容相关度；内容时效性平衡模块本文主要利用时间有效性因子来提高查询内容的时效性，使其查询时效性高的网页节点上升，时效性低的网页节点下沉。最后本文采用了五个大小不同数据集将 IER 算法与 Enhanced-RatioRank、Time-PageRank 等同类算法进行实验对比，实验结果表明 IER 算法不仅可以使搜索结果更偏向用户查询需求，而且还提高了查询内容的精确度，极大地提高了算法的查准率。

针对现在主流的分布式图划分算法在进行图划分过程中产生的子图之间的通信频率高导致计算效率不高的问题，本文提出了一种基于分布式图划分的个性化 PageRank 改进算法（Boruvka Adjacent Edge Structure DSP, BAESD），本文首先提出了一种基于 Boruvka 算法的分布式图划分算法，完成局部子图的构建与平衡化，将大规模网络图划分为多个局部平衡子图，使个性化 PageRank 算法的迭代计算转移到局部子图数据中，其次提出了一种相邻边缘结构（Adjacent Edge Structure, AES），AES 结构主要从两个方面对算法进行优化：一方面是提高图划分效率，尽可能减少图数据处理的时间；另一方面是该结构可以减小局部分区子图与子图之间的依赖关系密度，最后，本文利用 DSP 模型来加速个性化 PageRank 算法的迭代计算，从而进一步提高个性化 PageRank 算法在大规模图上的计算效率。本文算法在四个不同大小数据集上与 PM、RA 和 TDA 等主流算法进行实验对比，实验结果表明本文所提算法极大提高了个性化 PageRank 算法在大规模网络图上的计算效率。

1.4 论文的组织架构

本文共分为5章，具体的组织结构安排如下：

第1章：绪论。在本章中，首先对本文算法的研究背景及核心意义进行了概述与总结，其次在国内外研究现状中，本文对近年来很多学者针对个性化 PageRank 算法的研究进行了详细描述，并且在后面进行了优缺点分析和总结。最后，本文介绍了研究的主要内容及其组织结构。

第2章：相关工作。主要详细概述了本文研究涉及的基础理论知识与算法，本章首先介绍了图的基本概念知识，涵盖图的定义、顶点的度、存储以及子图基本概念等，其次阐述图划分相关的经典算法，然后概述了经典 PageRank 算法的核心思想，由于该算法不仅可以利用幂迭代计算模型，而且可以利用蒙特卡洛近似估算模型，因此本文对此展开了详细介绍，同时总结分析了算法的优势与不足之处，针对个性化 PageRank 算法的核心概念进行了重点阐述，最后概述了 Boruvka 算法与 DSP 计算模型的核心思想。

第3章：基于多特征因子的个性化PageRank 算法查询优化。本章一开始介绍了个性化 PageRank 算法广泛应用背景及一些基本概念知识，并且对该算法的优势与不足展开了总结分析，从而开始引出本文的主要研究工作，本章首先首先介绍了个性化 PageRank 算法（简称 PPR 算法）在大规模网络图中排序查询优化的一些主流算法，其次描述了该类算法所存在的一些问题，然后对本文提出的 IER（Improved Enhanced-RatioRank）算法进行了详细的介绍，最后利用 RFM 模型和两个指标（查准率与鲁棒性）对本文所提算法进行实验分析。

第4章：基于分布式图划分的个性化 PageRank 改进算法。本章首先介绍了分布式图划分算法的基本概念知识，其次介绍了个性化 PageRank 算法（简称 PPR 算法）在大规模网络图中高效计算问题的一些主流算法，其次描述了该类算法所存在的一些问题，然后对本文提出的 BAESD 算法进行了详细的介绍，最后利用三个指标（算法准确度、算法的计算效率和存储空间）对本文算法进行实验对比分析。

第5章：结论与展望。首先对本文研究内容进行总结分析，其次指出了实验中存在的不足之处，最后提出了对下一步研究工作的展望。

第 2 章 相关工作

2.1 图论基础知识

图论[30]是一种研究图形和节点之间关系的理论，在工程学、信息技术等领域中有着广泛的应用。在图中，每个节点代表一个对象，节点之间可以形成某种特定的关系。图论的核心思想是将现实世界的复杂问题抽象为图形，通过数学模型和图形表示来更好地理解和描述这些问题。因此，图论已经成为探究各种现象的重要工具之一。随着互联网技术的不断更新迭代，图论的应用范围也在不断扩大。例如，人工智能语言、大数据技术、电路系统分析等领域都在广泛地应用图论。在人工智能中，图论可以用于构建知识图谱和分析语义关系；在大数据技术中，图论可以用于分析网络拓扑和社交网络；在电路系统分析中，图论可以用于分析电路网络。通过图论的研究，可以更好地理解和描述现实世界中的复杂问题，并为人类做出更好的决策提供基础。因此，图论在各个领域中的应用越来越广泛，已成为解决现实问题的重要工具之一。

（1）图的定义

在计算机科学和数学中，图是由节点（也称为顶点）和它们之间的边（也称为连线或弧）组成的抽象数据结构。图是许多实际问题的自然模型，包括社交网络、交通网络、通信网络等等。图是由有限个顶点集V和边集E组成，具体如公式（2.1）所示：

$$G = (V,\ E) \tag{2.1}$$

用G表示图，其中$V(G)$可以表示图G中的顶点集，该集合包含有限个元素且非空；$E(G)$表示图G中所有链接边构成的边集合。图可以分为有向图和无向图，由若干的链接边在一定的约束条件下将若干个节点相互连接起来形成一个图结构，如果从任何一个节点出发的链接边有指定的方向到其他的节点，那么图G表示有向图，如图 2-1 所示，如果节点与节点之间的链接边不存在指定方向，则图G表示无向图，如图 2-2 所示。

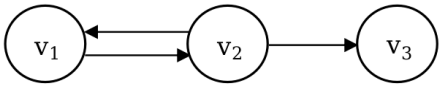


图 2-1 有向图G_1

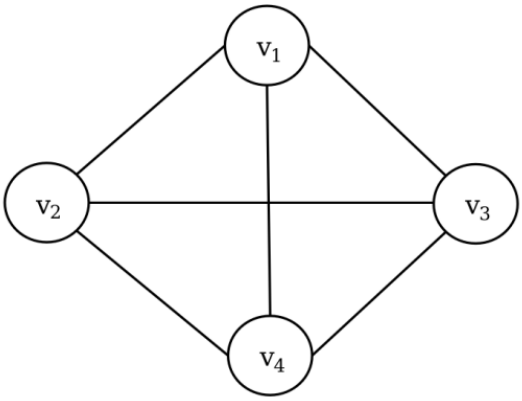


图 2-2 无向图G_2

（2）顶点的度

在一个图中，一个顶点的度是指该顶点所关联的边的数目。顶点的度包括入度与出度，对于无向图，它是由一组节点和一组连接这些节点的无序边构成的，其中节点的度数是指与该节点相连的边的数目，连通性描述了图的整体结构是否连通，路径是节点之间的边按照顺序连接而成的序列，而环则是一条起点和终点相同的闭合路径。顶点v的度数是指该顶点关联边的总条数，记为$TD(v)$。在具有n个顶点e条边的无向图中，如公式（2.2）所示：

$$\sum_{i=1}^{n} TD(v_i) = 2e \tag{2.2}$$

有向图是由若干个顶点及连接它们的边构成的一种图形结构。每个顶点都有一个对应的度数，度数分为入度和出度。入度表示有多少条边从其他顶点指向该顶点，而出度则表示从该顶点出发有多少条边指向其他顶点。如果一个有向图中的所有顶点的入度和出度都相等，那么这个有向图就是一个平衡图。在一般情况下，一个有向图中顶点的入度和出度并不一定相等。例如，某个顶点的出度可能比入度大，也可能比入度小，还可能相等。具体来说，对于一个顶点v，它的入度可以计算为所有指向该顶点v的边的数量之和，记作$ID(v)$，而它的出度可以计算为从该顶点v出发，指向其他顶点的边的数量之和，记作$OD(v)$。具体如计算公式（2.3）所示：

$$\sum_{i=1}^{n} ID(v_i) = \sum_{i=1}^{n} OD(v_i) = e \tag{2.3}$$

（3）图的存储

图是由节点和它们之间的边组成的一种数据结构。常见的图的存储概念包括

邻接矩阵、邻接表、关联矩阵和十字链表。图是一种抽象数据类型，常用来表示不同元素之间的关系和连接。在计算机科学中，图通常由节点和边组成，其中节点表示元素，边表示元素之间的关系。图可以有两种表现形式：无向图和有向图。无向图中的边是没有方向的，而有向图中的边则有方向。对于有向图，可以把边看作有起点和终点的有向边。图的存储方式主要有两种：邻接矩阵存储和邻接表存储。当图中的节点数量相对于边的数量大很多时，可以称这种图为稠密图，稠密图的存储一般用邻接矩阵表示。而当图中的节点数量相对于边数量小很多时，可以称这种图为稀疏图，稀疏图是指存储元素个数相对不多，该图的存储一般用邻接表表示。邻接矩阵存储是利用一个二维数组来存储图数据。对于无向图，邻接矩阵是一个方矩阵，而且还是一个对称矩阵。如果图节点之间存在关联边，则可以在对应的矩阵坐标位置上用 1 标记，反之，可以用∞标记。对于有向图，因为图中任意两个节点之间的链接边的数目不一定是两条，因此邻接矩阵是一个非对称矩阵。邻接矩阵存储可以很清晰地表达出图顶点之间是否存在关联边。邻接表存储是一种常见的图存储方式。在图中，每个节点对应一个链表，链表中存储的是该节点所连接的边的信息。具体而言，每个节点的链表中存储的是指向该节点的边的信息，包括边的起点、终点以及权重等。邻接表存储方式可以节省空间，因为它只需要存储有关联边的节点，而不需要为没有边的节点分配额外的空间。总的来说，邻接矩阵存储和邻接表存储各有优缺点，应根据具体情况选择合适的存储方式。稠密图适合使用邻接矩阵存储，而稀疏图则适合使用邻接表存储。

如果图$G = (V，E)$，则该邻接矩阵A是是一个n维方矩阵，且G的顶点编号依次为$v_1，v_2，\cdots，v_n$。若$(v_i，v_j) \in E$，否则$A[i][j] = 0$。例如，图 2-3 是一个由五个顶点构成的无向图 G。

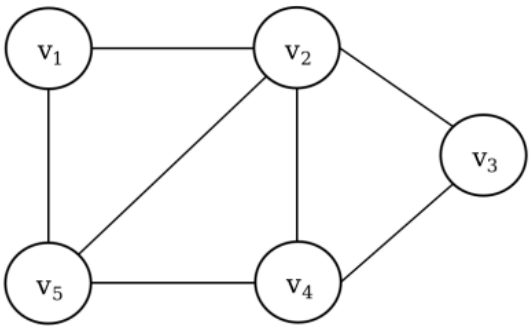
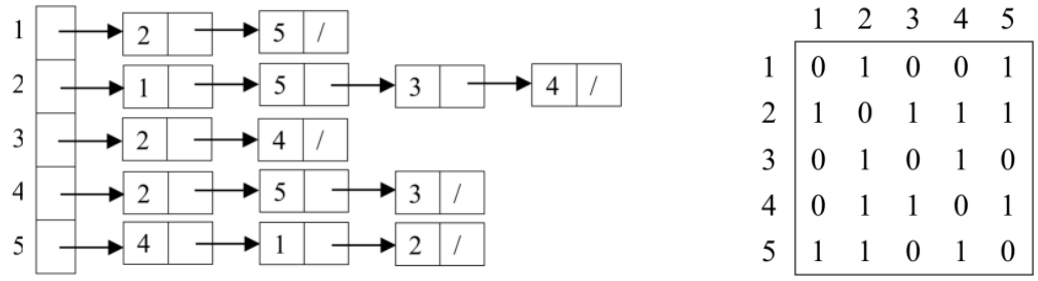


图 2-3 无向图 G

邻接表存储的定义：在图G中如果存在若干个顶点，且某个顶点可以对应一个单向链接表，且可以用单向链接表中的结点代表与该顶点相关联的边，该单链表称为该顶点的边表。图 2-4 为无向图的两种表现形式。



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

图 2-4 无向图的两种表示

顶点表结构可以存储顶点数据域和边表指针域，邻接点域和指针域可以用边表结构表示。因此在邻接表中存在两种结构：顶点表结构和边表结构，如图 2-5 所示：

图 2-5 顶点表与边表节点结构图

（4）子图概念

在图论中，子图指的是一个图的一部分，它包含了原图的一些顶点和边，但不一定包含所有的顶点和边。如果一个图G的子图也是一个图，则称其为G的真子图。如果存在两个图G和G'，且两个图都是若干个顶点与若干个边构成的，如果图G'中的顶点集是属于图G的子集，图G'中的边构成的集合是属于图G的子集且图G和G'两者都是存在且非空的，那么它就是G的生成子图。

2.2 图划分算法

2.2.1 传统式图划分算法

传统的图划分算法目的是将一个大规模的图划分成多个具有相似性质的子图，通常用于优化问题中，例如在计算机科学领域中的并行处理。这类算法采用贪心策略，并包含以下步骤：首先将输入图表示成一个数据结构，对于每个节点给定一个初始划分；其次，通过迭代地移动节点，以达到最优划分质量；最后，在预设的迭代次数或算法运行时间超时时终止算法，并输出最终的划分结果。传统式图划分算法主要是一些早期比较经典的图划分算法，这些算法一般只适合小

规模图，而且大部分算法仅用于图的二分操作，但是通过递归过程可以引申为多个子图的划分。谱二分法早期是由 Donath[31]提出，该算法利用拉普拉斯矩阵迭代计算出对应的特征向量，然后特征向量构成一个划分约束条件，如果子区数大于 2，则可以通过递归来达到划分目标。如矩阵公式（2.4）所示：

$$l_{i,j}=\begin{cases}-1, & \text{如果 } e_{i,j}\in E\\ deg(v_i), & \text{如果 } i=j\\ 0, & \text{否则}\end{cases} \tag{2.4}$$

几何划分法以顶点坐标信息为基准来划分图。坐标二分法[32]是比较有代表性的方法。该方法根据顶点在某一方向上的坐标信息，递归地将图划分成两部分。然而，该方法的划分结果容易受到坐标信息的影响，因此同一图在不同坐标系下的划分结果可能会有所不同。惯性法[33]是一种改进方法，它以划分基准线为标准，使所有顶点到基准线的平方和最小。Kernighan 和 Lin 提出了 KL 算法[34]，该算法首先对图进行分布式划分，之后在子集中根据一定的约束条件迭代生成新的划分子区，虽然该方法具有较高的划分精度，但划分时间随着迭代次数成正比，因此只适用于顶点数较少的小规模图。

2.2.2 流式图划分算法

流式图划分算法是一种能够处理动态变化的图数据并将计算任务划分到多个处理器中并行处理的算法。它的主要目的是提高计算效率、实现负载均衡和优化划分方案。在流式图划分算法中，流图是一个有向无环图，用于表示计算任务之间的依赖关系。其中，节点表示计算任务，边表示任务之间的数据依赖关系。划分是指将流图中的节点划分到不同的处理器上并行执行计算任务的过程。划分的过程可以分为三个阶段：初始化阶段、优化阶段和终止阶段。在初始化阶段，算法会根据流图的特性和任务的特性，将任务初始化到不同的处理器上。优化阶段采用贪心策略，即根据当前状态，选择最优的节点移动方式，以优化划分方案。贪心策略的核心在于每次迭代都会选择最佳的节点移动方案，以期望达到最优解。为了实现流式划分，算法需要考虑任务的优先级、任务的执行时间等因素，动态地更新划分方案。这就要求算法能够快速响应图数据的动态变化，并在此基础上进行划分。为了应对这种动态变化，流式图划分算法采用流式划分的方式，即动态地更新划分方案。在流式划分中，算法会在运行过程中不断地调整任务的分配方案，以保证每个处理器上执行的计算任务的负载尽量均衡。负载平衡是流式图划分算法的核心，它是保证每个处理器上执行的计算任务的负载尽量均衡的关键。为了实现负载平衡，流式图划分算法会根据不同的负载情况，动态地将任务分配到不同的处理器上，并对每个处理器上的任务进行调度。在负载平衡过程中，算

法会根据任务的特性、任务之间的依赖关系等因素进行分析，以达到负载均衡的目的。最后，流式图划分算法会在终止阶段根据预设的条件，终止划分过程，并输出最终的划分结果。流式图划分算法的应用场景广泛，可以应用于分布式计算、高性能计算等领域，提高计算效率，实现负载均衡，优化划分方案，实现更高效的计算。

在流式划分图算法中，HASH 划分算法是最基本的,它以图中顶点或边的编号作为输入，通过构建的散列映射函数将输入映射到不同的分区，该算法的缺点是会导致较高的割边率且划分质量较差。DBH 算法[35]基于 HASH 算法进行了改进，会优先切割度数较高的顶点。流式图划分算法[36-40]主要是针对大规模流式数据图进行划分的，该算法首先将原图转化成若干个由顶点及边组成的集合序列，其次在一定约束条件下划分对应的序列集，所以在整个图划分过程中只需要对图遍历一次，不需要进行多次迭代计算，提高了划分效率，但是该算法在划分过程中是依据当前部分数据来进行划分的，所以划分精度不是很高。

2.2.3 分布式图划分算法

分布式图划分算法是一种用于将大规模的图数据分布式地划分到多个处理节点上的算法，旨在提高大规模图数据的处理效率和可扩展性。在该算法中，图数据通常以稀疏矩阵的形式表示，节点和边分布在不同的处理节点上。该算法的核心思想是通过优化划分方案，将图数据均衡地分配到不同的处理节点上，以实现负载均衡和减少通信开销。分布式图划分算法通常包含以下步骤：首先将输入的图数据分配到不同的处理节点上，并进行初始化；其次，通过迭代的方式不断优化图的划分方案，以实现负载均衡和减少通信开销。在迭代过程中，算法会动态地调整节点和边的分布，以期望达到最优的划分结果。最后，当达到预设的迭代次数或者算法运行时间超过预设时间时，算法终止，并输出最终的划分结果。在分布式图划分算法中，负载均衡和通信开销是两个关键问题。负载均衡是指在多个处理节点上均衡地分配计算任务，以提高计算效率和减少计算时间。通信开销是指在处理节点之间传输图数据时所需要的时间和带宽开销，它对算法的性能和可扩展性有着重要影响。因此，在分布式图划分算法中，需要考虑图的拓扑结构、计算任务的特性和处理节点的带宽等因素，以优化划分方案并减少通信开销。Kappa 算法[41]和 JA-BE-JA 算法[42]是分布式图划分比较经典的算法，Kappa 算法在粗糙化阶段加入了匹配机制，并且使用 FM 算法来细化操作，而 JA-BE-JA 算法利用局部搜索机制和模拟退火技术对图进行划分，通过顶点的交换来达到最优划分。分布式计算可以处理海量数据，因此分布式图划分非常适用于大规模图数据的划分。分布式框架的普及也使得分布式资源更容易获取，如谷歌提出的

MapReduce 并行计算模型[43-44]，它将并行化、容错、局部优化、负载均衡等细节都放在底层，极大地降低了并行编程的难度，如图 2-6 所示，详细描述了利用 MapReduce 并行计算的过程。

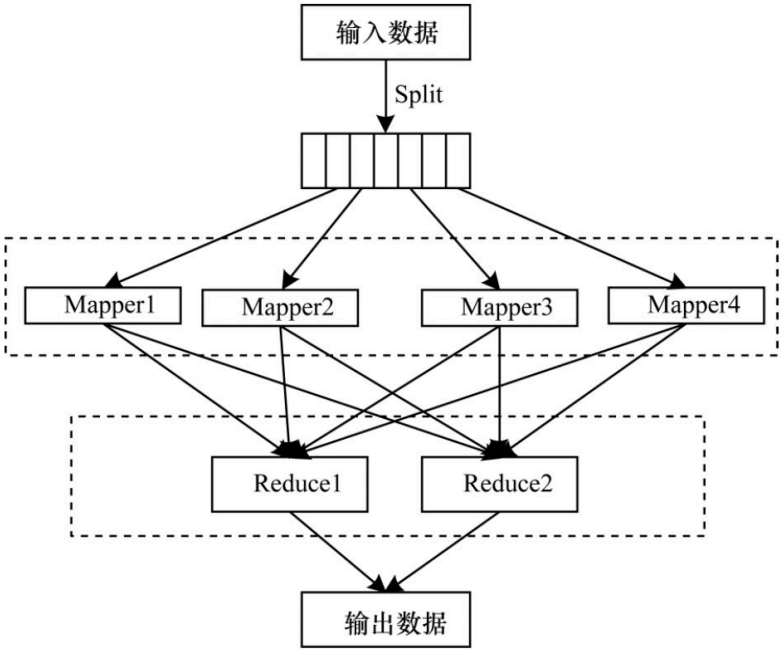


图 2-6 MapReduce 并行计算流程图

2.3 马尔科夫链

数学家马尔科夫提出了马尔科夫链（马氏链）[48-49]，马尔科夫链是一种常用于不同领域的数学模型，包括自然语言处理、金融建模和生物信息学等。在自然语言处理中，文本可以看作是一个马尔科夫链，其中每个状态表示一个单词或短语，当前状态表示当前单词或短语，下一个可能的状态表示下一个单词或短语。这个模型可以应用于文本生成和语言模型等任务。在金融领域，马尔科夫链通常被用于构建股票价格模型和风险管理模型。在生物信息学中，马尔科夫链可以被用于序列分析和蛋白质结构预测等任务。马尔科夫链具有“无记忆性”的特点，也就是说，每个状态的转移只与其前一个状态有关，而与更早的状态无关。因此，马尔科夫链是由状态空间和状态转移概率矩阵组成。马尔科夫链可以定义为一类比较特殊的随机过程，这里将系统在不同的时间点所处的状态空间记为$R = X(t), t \in T$。如果一个系统在时刻$t = n$前所处的状态与它在时刻n后所处的状态不存在依赖性，则可以称这个随机过程具有马尔科夫属性[4]。系统状态的转移是指系统根据一个特定概率值从一个状态转移到另一个状态，不同的状态转移间的概率值构成的矩阵称为系统的状态转移矩阵。马尔科夫性使得马尔科夫链可以通

过简单的概率计算方法来预测未来状态的分布，或者从过去状态的分布中推断出其它信息。

如果存在网页j链入网页i，那么$(i,j)\in e$,有向图g的邻接矩阵可以表示为$G\in N^{n\times n}$，它可以表示网页节点之间的链接关系图，如果用户以一个重复概率值点击内容网页中的每个链接时，相同的概率点击它所在网页中每个超链接，那么网页之间的转移概率矩阵$A\in R^{n\times n}$如公式（2.5）所示：

$$\mathrm{A}(i,j)=\begin{cases}\frac{1}{\sum_{k=1}^{n}G(k,j)} & G(i,j)=1;\\ 0 & G(i,j)=0;\end{cases} \tag{2.5}$$

综上所述：矩阵A与网页邻接矩阵G结构十分相似，两者的矩阵结构都比较稀疏。由于网页链接图中的网页在一般情况下不包含超链接，所以矩阵A中每一列元素将全部为 0。在该约束条件下，矩阵A将不满足随机矩阵的性质，即对应的马尔科夫链不存在平稳分布的特征。矩阵A公式如（2.6）所示：

$$\hat{A}=A+vm^{T} \tag{2.6}$$

其中$m\in N^{n\times 1}$是列向量全为 0 或 1 的向量，$\mathrm{m}(i)=1(1\le i\le n)$当且仅当$A$中第$i$列元素全为 0，$v\in N^{n\times 1}$是$n$维列向量且服从概率分布，即$0<v<1$且$\|v\|_1=1$。向量v表示当用户根据输入网页跳转到不包含超链接网页的访问概率，由 Perron-Frobenius 定理[50]可知，$\hat{A}$可以表示为一个列随机矩阵。因为$\hat{A}$服从唯一的稳定分布状态，所以$\hat{A}$必须保证具有不可约性。当然，用户也可以通过其他方式来搜索包含超链接的网页节点，比如根据用户的兴趣偏好来访问页面，概率向量v是指个性化向量，该向量描述了用户基于个人兴趣偏好的情况下访问每个网页的概率，综上所述，网页排序转移概率矩阵$\hat{P}$如公式（2.7）所示：

$$\hat{P}=d\hat{A}+(1-d)ve^{T} \tag{2.7}$$

式中，阻尼系数$0<d<1$描述的是用户根据超链接方式每次访问一个新网页时的概率值，也可以看作网页节点构成网络有向图的权重，$e\in N^{n\times 1}$是元素全为 1 的n维列向量。因此，马尔科夫链问题的求解即可以转化为线性系统的求解，如公式（2.8）所示：

$$Ax=0,A=I-\hat{P},\ x>0,\|x\|_1=1 \tag{2.8}$$

其中稳定分布x表示网页排序向量，分量大小可以代表网页节点的重要程度。

2.4 个性化 PageRank 算法介绍

2.4.1 个性化 PageRank 算法

PageRank，即网页排名算法，该算法属于搜索引擎的核心算法，搜索引擎可

以利用网页节点与节点之间 PageRank 值在网络链接结构图中的传递来计算结果。搜索引擎在一开始是利用手动方式对目录结果进行分类，随着互联网的信息越来越丰富，用户在互联网上的信息数据需求已经严重匹配不上搜索引擎的优化速度，为了提高搜索引擎的效率及搜索结果的质量，在 1998 年，谷歌创始人拉里·佩奇（Larry Page）和谢尔盖·布林(Sergey Brin)提出了 PageRank 算法，该算法根据搜索引擎中页面节点与其他页面节点之间的链接结构的重要性进行评级。该算法根据合适的权重比，对页面的重要性进行划分，经过数次迭代计算后传递给被搜索的网页，从而确定搜索页面的 PageRank 值。

个性化 PageRank 算法是在 PageRank 算法的基础上进行了改进，以考虑用户查询的相关性，提高搜索引擎反馈结果与用户搜索意图的匹配度。这种算法可以有效地提高查询内容的准确度，使得搜索引擎更能够满足用户的需求。在 PageRank 算法中，节点的权重值是由该节点被其他节点所链接的数量决定的。在个性化 PageRank 算法中，节点的权重值除了考虑链接数量外，还考虑了用户的兴趣偏好，从而使得搜索结果更符合用户的需求。具体地说，个性化 PageRank 算法可以看作是一组特定节点集根据一定的随机游走准则来度量访问其他节点的概率。通过这种方式，算法可以计算每个节点与其他节点的相对亲密度，从而得到每个节点的概率值（PPR 值）。在计算过程中，用户可以根据自己的兴趣偏好来跳转到某个网页，从而进一步优化搜索结果。在计算结果值稳定之后，迭代过程会停止，这样就可以得到每个节点的最终权重值。个性化 PageRank 算法公式可以由 PageRank 算法公式推导而来。PageRank 算法公式是基于概率的，用于计算每个节点的权重值。个性化 PageRank 算法公式在此基础上增加了用户的偏好因素，从而可以更好地反映用户的需求。个性化 PageRank 算法公式可由 PageRank 算法公式推导如（2.9）所示：

$$PR(u) = \gamma(1-d) + d\sum_{v\in B(u)}\frac{PR(v)}{N_v} \tag{2.9}$$

γ是一个用户兴趣偏好节点的概率分布，偏好节点概率满足$\sum_{i\in u}\gamma_i=1$，d 是一个跳转概率系数。从公式可以看出，个性化 PageRank 算法考虑了与节点相关的所有节点，而不仅仅是其链接的节点。这种算法可以更好地反映节点与其他节点之间的相对亲密度，从而得到更准确的权重值。此外，个性化 PageRank 算法还可以根据用户的兴趣偏好来跳转到某个节点，从而得到更符合用户需求的搜索结果。在网页节点结构图中，如果一些网页节点与其他网页节点不存在任何关联，那么这些网页称为孤立网页，使得下一个访问网页可以是任意网页，此时就会出现等级下沉(Rank Sink)现象，由于网页节点在随机游走的过程中会根据一个跳转

概率 d 链接到下一个网页节点，因此可以减弱等级下沉现象，这样可以成功的解决 Rank Sink 问题。

2.4.2 个性化 PageRank 算法的计算方法

目前，研究人员针对个性化 PageRank 算法的研究主要分为两类：基于幂迭代的精确方法[51]与基于蒙特卡洛思想[52]的近似估计方法。

（1）幂迭代方法

幂迭代方法是一种基于矩阵运算的迭代算法，在一定约束条件下，它通过邻接矩阵初始PPR值向量进行迭代计算，直到PPR值收敛为止。尽管幂迭代方法的计算量较大，但该方法能够得到精确的PPR值。因为个性化PageRank可以看作是一组特定结点集根据一定的随机游走准则来度量访问其他结点的概率，则求解PPR向量的计算公式如（2.10）所示：

$$\varphi_s = \alpha * u + (1-d)v * \varphi_s \tag{2.10}$$

式中：u表示个性化向量，表示用户偏好节点的概率，起始节点集$U_0 \in U$，其中包含k个节点，记作$u_i \in U_0$,$\sum_{i=1}^{k} u_i = 1$,而其余节点则取值为 0；PPR 迭代计算的具体步骤如下：首先，针对每个网页节点需要初始化其初始概率值，并将其带入公式中进行迭代计算。其次，需要计算出连续两次迭代后 PPR 向量的差值，即$\left|\varphi_s^i - \varphi_s^{i-1}\right|$。当误差范围$\left|\varphi_s^i - \varphi_s^{i-1}\right| <$ ε 小于预设的阈值 ε 时，迭代会停止，那么第i次迭代计算的 PPR 向量值即为最终结果，根据误差阈值 ε 可以得出幂迭代计算的时间复杂度是$O\left(mlog(1/\varepsilon)\right)$,每次迭代计算需要进行m次计算。幂迭代算法的伪代码如算法 2-1 所示：

算法2-1：幂迭代算法

Input：跳转系数d，误差阈值ε，转移概率矩阵M；

Output：φ_s

1. Initialization: personalized vector u，i = 0；
2. while$\left(\left|\varphi_s^i - \varphi_s^{i-1}\right| < \varepsilon\right)$ do
3. i++;
4. $\varphi_s = \alpha * u + (1-d)v * \varphi_s$
5. end if

（2）蒙特卡洛方法

蒙特卡洛方法是一种基于概率现象的数值模拟方法，也称为统计模拟法或近似估计方法。具体来说，蒙特卡洛方法从图中随机选择一个节点开始，然后按照一定的概率规则，沿着图中的链接随机游走。通过多次随机游走，记录每个节点被访问的次数，并将其归一化，得到最终的PPR值。蒙特卡洛方法的计算量较小，

但估计值可能存在一定的误差。该方法的核心思想是依据随机采样的结果值来估计PPR值，因此，随机采样次数直接影响到最终结果的准确性。针对个性化PageRank算法基于随机游走模型这一特点，可以将蒙特卡洛思想与PPR向量的求解相结合。借助随机游走模拟算法，根据随机采样结果来近似估计PPR值，从而达到有效求解PPR值的目的。以下是蒙特卡洛算法步骤的描述：

（1）初始化 PPR 值：将每个网页的 PPR 值初始化为$1/n$，n是总网页数。

（2）构建链接图：根据网页之间的链接关系，构建链接图，其中每个节点表示一个网页，每个有向边表示一个链接。如果网页 A 链接到网页 B，则在 A 到 B 之间连接一条有向边。

（3）设置随机游走概率：为了使个性化 PageRank 算法具有收敛性，需要设置一个随机游走概率d，表示用户有d的概率按链接跳转到下一个网页，有$1-d$的概率随机跳转到任意一个网页。

（4）进行多次随机游走：对于每个网页，进行多次随机游走，模拟用户随机访问网页的行为，记录每个网页被访问的次数。

（5）计算 PPR 值：根据蒙特卡洛方法的原理，每个网页的 PPR 值可以被估计为其被访问的次数除以总的访问次数，然后乘以一个缩放因子$1-d$，加上一个平均值d/n，其中n是总网页数。这个平均值表示如果没有链接跳转的随机游走，每个网页都会有相等的 PPR 值。

（6）重复进行第 4 和第 5 步，直到 PPR 值收敛。收敛的判断可以通过设置一个阈值，当 PPR 值的变化小于该阈值时，算法停止。

Fujiwara 等人[10]提出了一种有效的方法 Castanet,该算法通过精确的网页节点排序来查询前k个节点，为了提高搜索效率，该算法采用迭代方式估计上下相似度边界值，且在每次迭代中动态的删除不必要的网页节点以进行 top-k 搜索，如果获取了 top-k 节点的精确排序，则 Castanet 终止迭代计算。Castanet 方法与最初的迭代方法相比较，通过构造用于相似性计算的子图来提高搜索效率。

2.4.3 随机冲浪模型

随机冲浪模型[45-47]是一种概率模型，用于搜索引擎排名算法中。该模型将网页与随机游走者建立联系，根据网页之间的链接结构和网页自身的重要性计算每个网页的排名。谷歌公司最初提出了该模型，并在其搜索引擎中广泛使用。在随机冲浪模型中，每个网页都被视为一个节点，链接则被视为从一个节点到另一个节点的有向边。在随机游走的过程中，用户以一定概率在当前页面停留，并以一定概率随机跳转到其他页面，跳转的目标页面由当前页面的链接结构和页面的排名等因素决定。多次随机游走可以得到一个概率分布向量，其中每个元素表示游

走结束时游走者在相应网页上的概率。通过计算随机游走者在不同网页上停留的概率，可以计算每个网页的 PPR 值，用于衡量网页的重要性和排名。在实际应用中，随机冲浪模型需要考虑诸多因素，如链接权重、随机游走概率等，并通过迭代算法进行计算。该模型的计算表达式如（2.11）所示：

$$PPR = \alpha u + (1-d)M * PPR \tag{2.11}$$

其中, PPR 值代表一个n维概率分布向量，该向量的结果值会在随机游走过程结束后趋于稳定。其中，d为随机跳转概率,M是有向图的邻接矩阵,该矩阵经过归一化处理后得到，M中的元素$m(\beta_i,\beta_j)$代表节点β_i将计算结果值转移到节点β_j的概率，u是一个$n(n=|V|)$维列向量，u可以表示初始化值，具体参数如公式（2.12）-（2.13）所示：

$$PPR = \begin{bmatrix} PPR(\beta_1) \\ PPR(\beta_2) \\ \cdots \\ PPR(\beta_n) \end{bmatrix}$$

$$u = \begin{bmatrix} 1/n \\ 1/n \\ \cdots \\ 1/n \end{bmatrix}$$

$$M = \begin{bmatrix} m(\beta_1,\beta_1) & m(\beta_1,\beta_2) & \cdots & m(\beta_1,\beta_n) \\ m(\beta_2,\beta_1) & \ddots & m(\beta_i,\beta_j) & \\ \vdots & & & \\ m(\beta_n,\beta_1) & & & m(\beta_n,\beta_n) \end{bmatrix} \tag{2.12}$$

若网页j到i存在链接关系，那么$m(\beta_i,\beta_j) = 1/N^{out}(\beta_j)$，否则$m(\beta_i,\beta_j) = 0$。因此，最终个性化 PageRank 算法的向量矩阵方法具体可以表示为：

$$PPR = \alpha \begin{bmatrix} 1/n \\ 1/n \\ \cdots \\ 1/n \end{bmatrix} + (1-\alpha) \begin{bmatrix} m(\beta_1,\beta_1) & m(\beta_1,\beta_2) & \cdots & m(\beta_1,\beta_n) \\ m(\beta_2,\beta_1) & \ddots & m(\beta_i,\beta_j) & \\ \vdots & & & \\ m(\beta_n,\beta_1) & & & m(\beta_n,\beta_n) \end{bmatrix} * PPR \tag{2.13}$$

2.5 Boruvka 算法

Boruvka算法[56]是一种解决最小生成树问题的算法，最小生成树（Minimum Spanning Tree）是一个连通无向图中所有边的权值和最小的生成树。因为该算法是一种分阶段的贪心算法，所以它可以在稠密图上高效运行，该算法通过多次扫描图中的边，每次选择能够连接两个不同连通分量、权值最小的边，直到整个图成为一个连通分量为止。其主要工作流程如下：

（1）初始化：将每个顶点都看作一个连通分量。

（2）扫描所有连通分量的边，选择每个连通分量的最小边，并将这些边存储在一个集合中。

（3）根据上一步中得到的边集合，将所有被选择的边连接的两个连通分量合并为一个连通分量。

（4）重复执行步骤（2）和（3），直到图变为一个连通分量为止。

Boruvka 算法在每一次扫描中都会找到每个连通分量的最小边，然后将这些边连接的连通分量合并为一个连通分量。因此，该算法在每一次迭代中都能够将图的连通分量数量至少减半，最终形成一个连通分量。由于 Boruvka 算法对于每个连通分量都选择了一条最小边，因此可以保证生成的最小生成树的权值是最小的。具体算法的核心代码如算法 2-2 所示，Boruvka 算法输入为图 graph，顶点数 vertices，输出为最小生成树分量 mst。

算法 2-2：Boruvka 算法

```
Input: graph，vertices
Output: 最小生成树分量mst
1: def find_min_edge(graph, vertices): // 寻找连接连通分量到其他连通分量的最小边
2: min_edge = (None, float('inf'))
3: return min_edge
4: def boruvka_algorithm(graph): // 主函数，实现 Boruvka 算法
5:mst = set() // 初始化 mst 和森林 forest
6: forest = [set([v]) for v in graph.keys()]
7: while len(forest) > 1: // 当森林 F 中有两个及以上的连通分量时
8: new_forest = []
9: for components in forest: // 找到连接连通分量到其他连通分量的最小边
10: u, v, w = find_min_edge(graph, components) // 将连接的连通分量合并为一个连通分量
11: for tree in new_forest:
12: if u in tree:
13: tree.update(components)
14: else:
15: new_forest.append(components | set([v])) // 将连接的边添加到 mst 中
16: mst.add((u, v, w))
17: forest = new_forest // 更新森林 forest
18: return mst
```

2.6 DSP 模型

1990 年，Leslie Valiant 在牛津大学提出了大同步并行 BSP（bulk synchronous parallel）模型，它的核心思想是将一个大规模计算任务划分成若干个小的计算子任务，并行地执行这些子任务，最后将结果合并起来。BSP 模型由局部计算、通信和阻塞同步三部分组成。在 BSP 模型的基础上，DSP（delta-stepping synchronous parallel）并行计算模型[59]被引入，该模型也称为多步前进同步并行模型。与 BSP 模型相比，DSP 模型更进一步地将每一步的局部计算变成了多步的局部计算，而不涉及节点之间的通信，它只进行一些局部数据的更新，如图 2-7 所示：

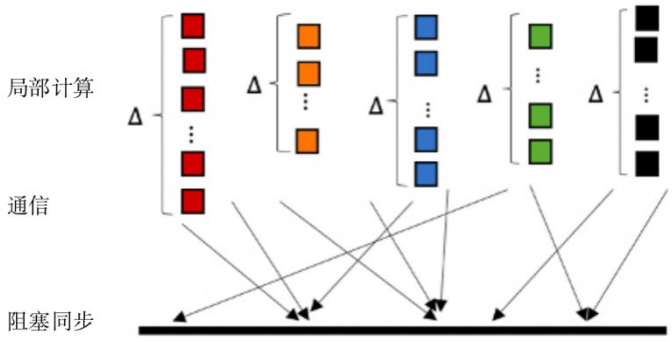


图 2-7 DSP 计算模型

如下图 2-8 所示：这里详细描述了 DSP 模型加速图节点计算过程，每个虚线方格表示一个计算节点，每个计算节点负责计算图中的每个局部节点集，在 DSP 模型中，由于可以将每个计算操作步可以分解成若干个局部计算步，在局部节点集中对数据进行局部更新操作，在第一轮同步的计算过程中，存在一个局部子集在一个计算节点中计算；在第二轮同步计算过程中，两个局部子集对应两个计算节点；第三轮的同步计算中，利用三个计算节点来计算三个局部子集；在第四轮的同步计算中，两个局部子集对应两个计算节点；在最后一轮的同步计算中，只存在一个局部子集在一个计算节点中。因此，利用 DSP 模型完成该图计算过程只需要 5 轮的同步计算。

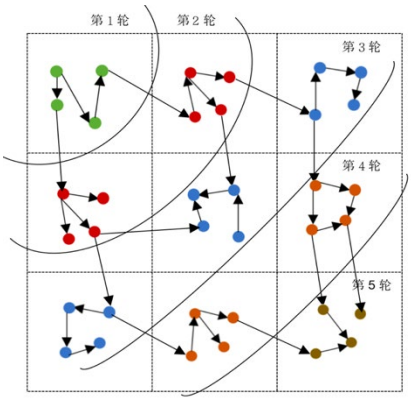


图 2-8 DSP 模型加速图节点计算过程

DSP 模型的收敛速度主要依赖三个因素：Δ，γ 和 β，Δ是指局部计算的步数，当数据集比较稀疏或数据集划分效果比较好时，可以选择大一些的Δ，反之，较小的Δ加速效果会更好。而β在一定程度上可以用数据分区间的依赖关系强度来解释，γ可以指代数据分区内的依赖关系强度，如果增强数据分区内的依赖关系强度（即增大γ），或者减弱数据分区间的依赖关系强度（即减小β），那么可以有效加速算法的收敛性。由于本文在上文中提出了一种融合相邻边缘结构的分布式图划分算法，它将图划分为规模相等的多个子集，在图划分过程中极大程度的减少了割边数，降低了割边率，也就是减弱了数据分区间的依赖关系密度（即减小β），这就为利用 DSP 模型加速迭代 PPR 的计算创造了条件。

2.7 本章小结

本文章节首先介绍了图论的基本概念，包括图的基本定义、存储和一些图划分算法等，并且介绍了分布式图划分算法的经典计算模型（MapReduce 并行计算模型），详细描述了该模型的计算流程，其次介绍了马尔科夫属性以及该算法的核心思想，然后详细概述了个性化 PageRank 算法的核心思想与计算方法，并且对随机冲浪模型、Boruvka 算法及 DSP 计算模型的核心思想做了详细介绍，最后从个性化 PageRank 算法的查询精确度和计算效率两个方面进行了优化提升。本文章节不仅为本文的展开提供了依据，而且也为接下来的实验验证提供了支持和衡量指标。

第 3 章 基于多特征因子的个性化 PageRank 算法查询优化

本章主要围绕本文研究内容所涉及的基于多特征因子融合的个性化 PageRank 算法查询优化进行详细阐述，首先介绍了个性化 PageRank（Personalized PageRank，PPR）算法在大规模网络图中排序查询优化的一些主流算法，其次描述了该类算法所存在的一些问题，然后对本文提出的 IER（Improved Enhanced-RatioRank）算法进行了详细的介绍，最后利用 RFM 模型和两个指标（查准率与鲁棒性）对本文所提算法进行实验分析。

3.1 问题提出

个性化 PageRank 算法的出现为搜索引擎的个性化搜索提供了重要支持，个性化 PageRank 算法可以帮助搜索引擎实现个性化搜索,该算法主要根据两个指标（用户的知识背景及自身属性）来评估用户的搜索意图，自 PPR 算法提出以来，PPR 算法在网络中有着广泛的应用，如好友推荐，网页搜索，社区检测，链路分析等。因此，提高个性化 PageRank 算法在大规模网络图中排序查询的查询精确度是一件至关重要的事情。

现有提高个性化 PageRank 算法在大规模网络图中排序查询的查询精确度的主流算法可以分为两类：一类是基于蒙特卡洛的近似估计算法，另一类是精确算法。近似算法以 Fujiwara Y 等人[10]提出的 Castanet 方法为主流，该算法的局限性就是难以确定一个合理准确的界限值，需要近似估计 PPR 值，严重影响算法的查询精确度。精确算法以 Singhand R 等人[53]提出的 Enhanced-RatioRank 算法为主流, Enhanced-RatioRank 算法公式如（3.1）所示：

$$ER(u) = (1-d) + d\sum_{v \in B(u)} \frac{\alpha V_u W^{in}_{(v,u)} + \beta W^{out}_{(v,u)} ER(v)}{TL(v)} \qquad (3.1)$$

式中，V_u表示网页v到网页u的链接访问点击量；$TL(v)$表示与网页相关所有链接的访问点击量；d是跳转概率；$W^{in}_{(v,u)}$为链入权重因子，$W^{out}_{(v,u)}$为链出权重因子；$B(u)$表示链入网页u的集合；α和β表示权重因子调节系数。

然而 Enhanced-RatioRank 算法仍然存在着一些问题，比如该算法考虑了用户的兴趣偏好，用户网络节点的 PPR 值会随着其查询偏好的变化而改变，而且会出现 PPR 值下沉到旧网页的问题，当网络规模图较大时，会导致查询内容匹配度较低，从而影响算法的查准率和查询结果精确度。由于本文是为了提高个性化 PageRank 算法在大规模网络图中排序查询的查询精确度，所以近似算法相对于

精确算法会损失算法的查询精确度，因此本文基于 Enhanced-RatioRank 算法展开算法研究及改进。

3.2 算法描述

针对 Enhanced-RatioRank 算法存在 PPR 值下沉、查询内容相关性低和搜索内容时效性低的问题，本文基于 Enhanced-RatioRank 算法提出了一种多特征因子的个性化 PageRank 查询优化算法，简称 IER（Improved Enhanced-RatioRank）算法，该算法主要通过主题漂移优化模块、内容相关度增强模块和时间有效性平衡三个模块对个性化 PageRank 算法进行查询优化，解决 PPR 值下沉、查询内容相关性低和搜索内容时效性低的问题，从而改善查询内容的精确度。IER 算法主要有三大核心组成部分：（1）主题漂移优化模块，本文在该模块中首先引入了权重因子，权重因子可以解决主题漂移问题，使其用户根据主题关键词查询时尽可能减少垃圾网页的情况。其次针对权重因子进行了优化，通过设置α和β权重比调节因子，通过调节权重比因子α和β来调控链入权重及链出权重的影响度，这样可以更好的解决 PPR 值下沉问题，避免主题漂移现象；（2）内容相关度增强模块，本文在该模块中提出了一种频数因子，它可以改善用户查准率，使用户搜索目标需求与反馈结果的匹配度更高，提高查询内容相关度；（3）内容时效性平衡模块，本文在该模块中提出了一种时效性平衡函数，该函数主要利用时间有效性因子来提高查询内容的时效性，使其查询时效性高的网页节点上升，时效性低的网页节点下沉。实验结果证明：该算法在内容相关性和查准率方面较其他同类算法有明显提高，极大的提高了算法的查询精确度。IER 算法公式如（3.2）所示：

$$IER(u) = (\gamma(1-d)) + d\sum_{v\in B(u)}\frac{\alpha W_{(v,u)}^{out} IER(v)+\beta V_u}{TL(v)})$$
$$IER(i) = IER(i) * T_{\mathrm{i}} \quad i \in A \tag{3.2}$$

式中，d是跳转概率，γ是一个用户偏好节点的概率分布，偏好节点概率满足$\sum_{\mathrm{i}\in\mathrm{u}}\gamma_{\mathrm{i}}$=1，$W_{(v,u)}^{out}$表示链出权重，$V_u/TL(v)$表示频数因子，α和β分别表示链出权重因子和频数因子所占权重比。T_i代表第i个网页节点的时间有效性因子，A指代所有网页节点构成的集合。例如：在查询的过程中，从偏好网页节点v出发，在用户想要访问搜索下一个网页节点时会生成一个局部的网页节点集合，这个局部网页节点集合可以划分为两种状态：其中一类网页节点集是一些与用户查找目标网页节点差距较大的节点，即从偏好节点v出发到该类局部节点集的概率远小于参照值，由于另一类局部节点与用户查找目标网页节点基本不存在关联性，属于无效网页节点，这些网页节点可以被忽略，因为处于第一种状态的局部节点是距离偏好网页节点v比较近的一些节点，该类节点是可以由 PPR 算法在离线计算中

得出。IER 算法框架如图 3-1 所示，本文将在下文对 IER 算法的三个核心部分展开详细介绍。

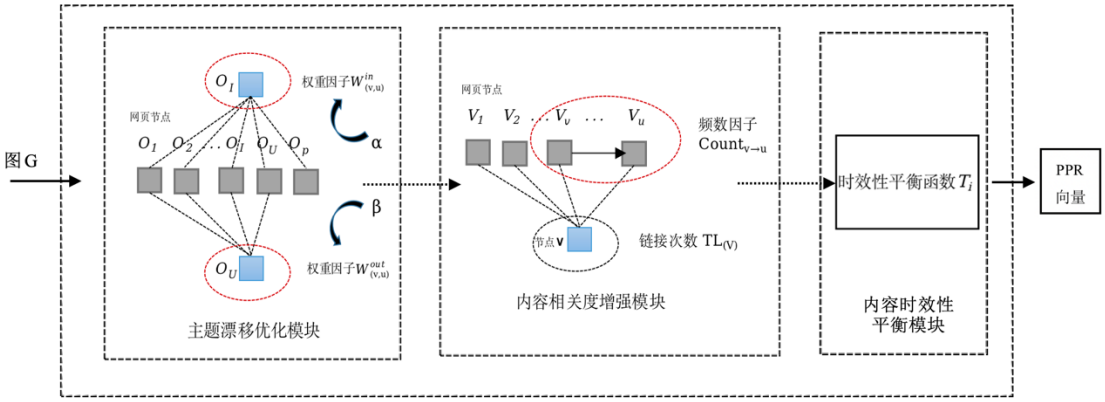


图 3-1 IER 算法框架图

3.2.1 主题漂移优化

加权页面排名算法是基于标准页面排名算法的一个扩展，在页面排名算法中，权重被均匀地传递到网页的所有链入链接，如果该网页的链入链接的权重总和较高，那么该网页排名会被赋予一个高权重的值，但是在加权页面排名算法的情况下，网页的排名是基于内链接和外链接的影响力给出的，而不是在内链接之间平均分配权重。内部链接和外部链接的数量在很大程度上能够影响网页节点的受欢迎程度，就针对任何用户查询返回的相关页面数量而言，由于加权页面排名算法是基于网页内容挖掘，所以该算法在网页方面提供了最高的相关性。

在大型网络图中，因为原始的 PageRank 在计算结果值的时候对网页链接结构图节点的度依赖性很高，这样很容易使 PPR 值沉淀到一些无效网页节点之中，出现 PPR 值沉淀情况，最后使用户查询结果出现主题漂移的情况。因此，本文首先引入权重因子，权重因子的主要作用就是可以平衡网页的重要程度，且可以分为链入权重因子和链出权重因子，权重因子可以根据网页节点与其他节点之间的链接数以及重要程度来分配不同的链出与链入权重值，在网页节点构成的网络结构图中，链入权重与链出权重相对于每个网页节点的的影响程度是不均衡的，因此本文引入两个权重比调节因子α和β，通过调节权重比因子α和β来调控链入权重及链出权重的影响度。某个网页节点的链入权重是指以该网页节点为中心，从其他网页节点指向该网页节点的链接边数目，因此链入权重在一定程度上可以表示该网页节点相对于其他网页节点的亲密度，也叫被动关联度。链入权重因子 $W^{in}_{(v,u)}$ 计算公式如（3.3）所示：

$$W^{in}_{(v,u)}=\frac{O_I}{\sum_{p=B(v)} O_p} \tag{3.3}$$

某个网页节点的链出权重是指从该网页节点出发，主动链接其他的网页节点的链接边数目，在一定程度上代表了该网页节点的主动意愿度，但随着网页节点 PageRank 值的迭代过程中，该网页节点的 PageRank 值可能会转移到其他网页节点上，从其他网页节点指向该网页节点的链接边数目，因此链出权重在一定程度上可以表示该网页节点相对于其他网页节点的主动关联度。链出权重因子$W_{(v,u)}^{out}$计算公式如（3.4）所示：

$$W_{(v,u)}^{out}=\frac{O_u}{\sum_{p=B(v)} O_p} \tag{3.4}$$

式中，O_I与O_u表示网页节点I和u的用户偏好概率值，O_p表示网页节点v与相关网页节点的用户偏好概率值，$B(v)$是网页节点v相关联的网页集合。为了更加直观的显示权重因子对于主题漂移优化的效果，本文通过举例来说明。如图 3-2 所示：Rank Sink 图由五个顶点构成的网络节点链接结构图，顶点与顶点之间存在的链接关系构成有向边。在个性化 PageRank 算法的迭代计算的过程中，由于在迭代计算过程中没有加入权重因子，如表 3-1 所示，图中顶点的 PPR 值会产生沉淀，表明主题漂移现象严重，如表 3-2 所示，迭代计算过程中加入权重因子后，PPR 值没有沉淀，说明主题漂移问题得到了解决。

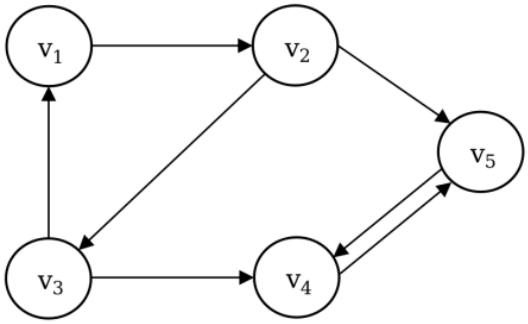


图 3-2 Rank Sink 图

表 3-1 未引入权重因子迭代过程中网页的 PPR 值

迭代次数	PR(v_1)	PR(v_2)	PR(v_3)	PR(v_4)	PR(v_5)
初始值	0.5	0.5	0.5	0.5	0.5
1	0.4	0.4	0.5	0.6	0.6
2	0.4	0.2	0.25	0.65	0.7
3	0.2	0.2	0.125	0.75	0.7
4	0.1	0.1	0.125	0.75	0.725
5	0.05	0.05	0.025	0.815	0.875
…					
n	0.0001	0.0002	0.0002	0.8125	0.8725

表 3-2 引入权重因子迭代过程中网页的 PPR 值

迭代次数	$PR(v_1)$	$PR(v_2)$	$PR(v_3)$	$PR(v_4)$	$PR(v_5)$
初始值	0.5	0.5	0.5	0.5	0.5
1	0.4	0.4	0.515	0.535	0.63
2	0.325	0.325	0.255	0.543	0.659
3	0.225	0.275	0.145	0.568	0.718
4	0.185	0.225	0.125	0.576	0.725
5	0.125	0.175	0.115	0.581	0.754
…					
n	0.1275	0.1955	0.111	0.5815	0.7569

3.2.2 内容相关度增强

网页排序算法计算结果值的时候对网页链接结构图节点的度及网页中所包含的东西依赖性很高，但在大多数情况下忽视了用户反馈，导致用户查询内容的相关度不够高。因此，用户反馈也是重中之重，其中包括显性反馈和隐性反馈。显性反馈是指用户通过主动参与的行为来表达对搜索内容的评价和反馈，例如点击、评分、评论、收藏等行为。它直接反映了用户对搜索结果的满意度和需求，是搜索引擎查询优化的重要指标之一。隐性反馈是一种通过用户个性化搜索网页留下的痕迹信息来推断查询结果的准确率，而没有用户的明确参与行为[53]。它主要利用用户搜索历史、点击记录、停留时间、滚动位置等信息，来分析用户的搜索行为和偏好，从而改善搜索结果的精准度和个性化程度。隐性反馈的优势在于不需要用户的主动参与，减轻了用户反馈的负担，同时也能够提供更加细致、精准的反馈信息。由于显性反馈的关系在一定条件下相当于网页节点之间的链接数量关系，所以可以看作是频数因子，频数因子通过网页浏览者对链接的关联次数表示，为了进一步提高用户查准率和内容相关度，在所有网页节点构成的一个大型网络图中，网页节点之间的相关链接次数是影响网页节点 PPR 值的一个重要因素，因此，为了增强查询内容的相关度，本文提出了频数因子，频数因子$Count_{v\to u}$计算公式如（3.5）所示：

$$Count_{v\to u}=\frac{V_u}{TL(v)} \tag{3.5}$$

式中，$TL(v)$表示网页节点v与其他网页节点的相关链接总次数，V_u表示网页节点v与网页节点u之间的相关联边数,即$V(v\to u)$值越大表示网页节点v与网页节点u的相关度越高，网页节点u应该获得更高的 PPR 值。为了验证加入频数因子的有效性，利用三个网页节点构成了一个网络有向图，且有向图之间的关联边是赋予权重的，如图 3-3 所示：

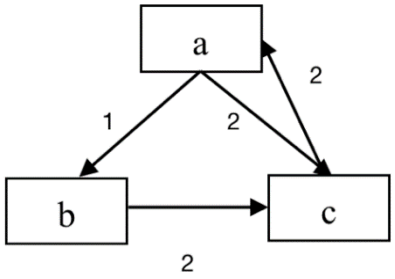


图 3-3 基于链接结构的 Web 图

网页 a、b、c 的 PPR 值计算表达式如（3.6）所示：

$$PPR(a) = (1-d) + d\left(PPR(c) \times \frac{2}{2}\right)$$

$$PPR(\mathrm{b}) = (1-d) + d\left(PPR(\mathrm{a}) \times \frac{1}{3}\right)$$

$$PPR(c) = (1-d) + d\left((PPR(b) \times \frac{2}{2}) + \left(PPR(a) \times \frac{1}{3}\right)\right) \quad (3.6)$$

由计算公式（3.6）可知：PPR(a)=1.10, PPR(b)=0.68, PPR(c)=1.21

而引入频数因子后算法计算所得PPR(a) = 1.217，PPR(b)=0.512，PPR(c) = 1.41，在算法引入频数因子后，网页 a 与网页 c 的 PPR 值在提高，由于网页 a 和网页 b 的出链都指向了网页 c，所以网页 c 的 PPR 值在增大，网页 a 包含了由网页 c 链出权重为 2 的入链，网页 b 包含了由网页 a 链出权重为 1 的入链，说明网页 a 比网页 b 重要程度更高，验证了算法加入频数因子后内容相关度增强。

3.2.3 内容时效性平衡

网页的链接数目是影响 PPR 值的重要因素之一。然而，许多网页的内容都是有时效性的，随着时间的推移，它们的价值不断降低。因此，本文提出了一种时效性平衡函数，该函数的核心思想利用时间有效性因子来提高内容时效性高的网页节点的排序，降低内容时效性低的网页节点排序，保持网页节点内容时效性的动态平衡，从而提高了查询排序的准确性，使得搜索结果更加准确、实用。时间有效性因子T_i计算公式如（3.7）所示：

$$T_i = \frac{1}{1 + \frac{X_i - min}{max - min}} \quad (3.7)$$

式中，X_i表示第i个网页节点与基准时间的差值（单位：天），i属于所有网页节点构成的集合，max表示集合i中的网页节点与基准时间的最大差值，min表示集合i中的网页节点与基准时间的最小差值。令 $\theta = 1 + \frac{X_i - min}{max - min}$，$T_i$随着 θ 不断增大的变化曲线如图 3-4 所示：

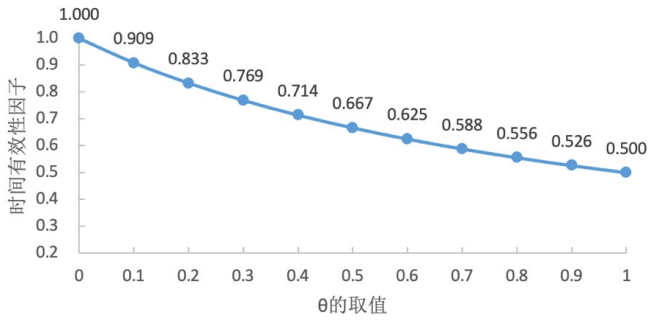


图 3-4 时间有效性因子T_i随 θ 取值的变化曲线

3.2.4 算法实现

如算法 3-1 所示，IER 算法输入为图 G，参数包括偏好概率γ，跳转概率d,迭代终止阈值 g，最大迭代次数 max_iterations，输出为PPR 向量值。

算法 3-1：IER 算法

Input: $G=(V, E)$，偏好概率γ，跳转概率d,迭代终止阈值g，最大迭代次数max_iterations

Output: PPR向量值

```
n = length(graph)  // 获取图的节点数
W_in(v,u)=ones(n)，W_out(v,u)=ones(n) // 初始化权重因子
Count_v→u = ones(n)  // 初始化频数因子
T_i = ones(n)  // 初始化时间因子
out_degrees = zeros(n)，in_degrees= zeros(n) //初始化入度与出度值
for i = 0 to  n-1:  // 计算每个节点的出度
  out_degrees[i] = sum(G[i])
  PPR = ones(n) / n  // 初始化 PageRank 值
 for iterations = 1 to max_iterations:
  old_PPR = copy(PPR)
 for i = 0 to n-1:
  IF out_degrees[i] == 0:  //处理没有出边的节点
  PPR[i]=0
 for j = 0 to n-1:
 if G[j][i] != 0:
 IER  ←W_in(v,u)，W_out(v,u)，Count_v→u，T_i
 PPR[i] = (1 – d) / n + IER  //计算新的 PageRank 值
 diff = abs(PPR - old_PPR).sum()  //计算 PageRank 值的差值
 if diff < g:  // 判断是否停止迭代
 return PPR  // 返回 PPR 向量
```

3.3 实验分析

为了验证本文所提出的算法的有效性，本文采用了真实的银行零售业务系统数据集进行了实验分析。这类数据集展示了每个客户节点之间的链接关系，它们会以一个巨大复杂的社交网络存在，每个客户节点对应一个 ID，节点与节点之间的链接关系可以构成一个大规模网络图。该类数据集的选取主要从两个方面考虑：一方面数据集设置不同的数据量大小来验证本文算法的鲁棒性，另一方面每个数据集上节点与节点之间都存在链接关系，可以构成一个大规模网络图，进一步提高了数据集与本文算法的匹配度，从而更加有效的验证算法的查询精确度。因为每个数据集具有对应的属性标签，例如交易金额、交易频次及交易时间等。本文首先根据交易金额利用权重因子计算公式得出权重因子，因为主动交易可表明客户对银行业务营销具有更高的接受度和价值。其次，本文根据交易次数利用频数因子计算公式得出频数因子，相当于网页节点链接量，衡量了客户在整个网络中的交易情况。然后，本文根据交易时间利用时间有效性因子计算公式得出时间有效性因子。由于本文算法还考虑到每个特征因子对节点的影响程度不同，因此本文引入了平衡系数α和β来控制权重因子和频数因子的权重比例。综上所述，针对个性化 PageRank 算法在大规模网络图排序查询中存在查询结果精确度及个性化搜索时效性不高的问题，本文提出的 IER 算法通过计算出每个客户节点的新 PPR 值，并根据 PPR 值排名，越高的 PPR 值代表节点的潜在价值和影响力越高，从大规模网络图中精准高效地挖掘出重要发展客户。最后本文采用了五个大小不同数据集将 IER 算法与 Enhanced-RatioRank、Time-PageRank 等同类算法进行实验对比，具体的实验分析与实验结果将在下文展开详细描述，实验流程图见图 3-5。

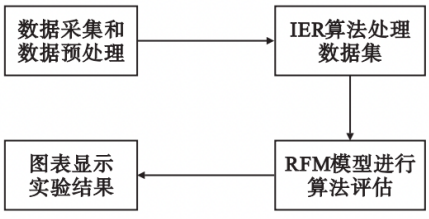


图 3-5 基于 IER 算法的实验流程

3.3.1 实验环境与数据集

本文实验环境：Win10 操作系统，处理器为 Intel(R) Core(TM) i7-5500U，实验模型使用 Python3.6 进行构建，对改进的 IER 算法进行实验仿真，本文的数据集来自银行零售业务系统的真实交易数据，一共有五条数据集，其具体信息如表 3-3 所示：

表 3-3 实验数据

数据集	节点 ID 数	链接数
Bank_tracsactions1	6390	76889
Bank_tracsactions2	9775	106785
Bank_tracsactions3	11087	125779
Bank_tracsactions4	20996	191096
Bank_tracsactions5	32880	342097

3.3.2 算法的关键步骤

(1) 数据预处理

本文实验的数据集经过数据处理后，数据交易记录信息包括客户节点 ID、客户交易金额、客户交易时间，每一个客户 ID 代表一个客户，所有的客户数据交易记录都处于 2010 年 1 月 1 日至 2020 年 8 月 1 日之间，设置一个基准时间为 2021 年 9 月 9 日。

(2) 特征因子计算

根据客户主动交易数据集，由公式 3-3 和 3-4 可以计算出权重因子，根据客户的交易次数，由公式 3-5 计算出频数因子，通过调节权重比因子α和β来调控链入权重及链出权重的影响度，经多次实验论证，取α=0.55，β=0.45，本文算法在查准率与鲁棒性方面均可以达到一个良好的效果。

(3) 数据归一化处理

X_i表示第i个客户交易时间与当前时间的差值（单位：天），采用$max-min$离差标准化方法对X_i进行数据归一化处理，再由公式 3-7 计算出时间有效性因子，利用时效性平衡函数对节点的时效性保持动态平衡。

(4) 算法评估

以 RFM 模型，以客户交易时间与标准对照时间的差值作为 Recency,客户交易次数作为 Frequency,以客户单位交易次数的交易金额作为 Monetary，通过主题漂移优化模块、内容相关度增强模块和时间有效性平衡三个模块对个性化 PageRank 算法进行查询优化，最后将实验结果进行分类，从而验证算法的有效性。

(5) 实验结果计算

根据 IER 算法可知，将权重因子、频数因子和时间有效性因子与算法融合得到一个新的 PPR 值，客户的 PPR 值越高，说明客户的潜在价值和交易影响力越高，越有可能成为重要发展客户，取出排名最高的前十名和排名最低的后十名的值，计算结果见表 3-4 和表 3-5。

表 3-4 前十名客户对应的 IER 值

客户 ID	IER 值
243	0.007781241
537	0.00726986
263	0.007105935
965	0.006919674
126	0.006838479
747	0.006740278
502	0.006697831
285	0.00663286
146	0.006526682
16	0.006508354

表 3-5 后十名客户对应的 IER 值

客户 ID	IER 值
920	0.000744652
318	0.000737197
424	0.000727598
759	0.000701681
135	0.000672079
62	0.000593919
923	0.000589047
558	0.000544394
408	0.000391721
93	0.000375249

3.3.3 RFM 模型分析与验证

RFM 模型的核心思想是将客户的行为数据转化为 RFM 评分，然后根据不同的评分将客户划分为不同的类别。为了计算 RFM 评分，通常需要对数据进行一些预处理和归一化处理，例如将购买时间转化为天数或月数，将购买金额转化为对数等。然后，可以将 RFM 评分进行综合计算，得出每个客户的总评分，根据评分的高低将客户分为不同的类别。利用 RFM 模型，本文可以快速的对客户进行按照价值需求进行划分，以便于针对客户定制个性化服务，这个模型通常采用最近一次消费距离现在的时间（Recency）、消费频率（Frequency）和消费金额（Monetary）三个条件来评价客户的价值。如果客户的最近一次消费距离现在时间短、消费频率高、消费金额大，那么这个客户的潜在价值就会更高。

本文以 RFM 模型为基础，以客户交易时间与标准对照时间的差值作为 Recency,客户交易次数作为 Frequency,以客户单位交易次数的交易金额作为 Monetary,通过客户的 RFM 行为特征来衡量分析客户的影响力和潜在价值，从而验证算法实验结果的有效性。本文使用了三个指标（Recency, Frequency, Monetary）的平均值作为参照值，来比较客户的不同表现情况，因为总共有三个指标，每种指标有两种状态：非 1 即 0。因此，总共有 8 种不同的客户类别[54]，具体的计算结果可以参见表 3-6。

表 3-6 客户类别划分标准

R	F	M	客户类别
1	1	1	重要发展客户
1	1	0	重要保持客户
1	0	1	重要保持客户
1	0	0	一般客户
0	1	1	重要挽留客户
0	1	0	一般客户
0	0	1	一般客户
0	0	0	无价值客户

3.3.4 实验结果分析

将实验结果进行聚类分析。首先分别计算二十名客户的 Recency、Frequency 和 Monetary，其次分别把 Recency、Frequency 和 Monetary 的均值作为三个指标的参考值，然后将二十名客户的 RFM 值与总 RFM 均值比较，如果每个客户的 Frequency 和 Monetary 值大于参照值，则用数字 1 分别在 F 和 M 下方标记该指标，反之则用数字 0 标记，因为客户交易时间越接近标准对照时间，代表客户的交易活跃度越高，所以 Recency 越小，代表客户的交易活跃度越高，说明客户的潜在价值越高，因此这里规定每个客户的 Recency 值小于参照值，则用数字 1 在 R 下方标记该指标，反之则用数字 0 标记。最后由表 3-6 可以得出客户类别分类结果，见表 3-7。例如 ID 为 243 的客户，Recency 值为 694 小于参照值 1163，Frequency 值为 26 大于参照值 16.4，Monetary 值为 733 大于参照值 445，则该客户的 RFM 分别用 1，1，1 来标记，由表 3-7 可以确定该客户级别为重要发展客户。

表 3-7 客户类别分类结果

客户 ID	Recency	Frequency	Monetary	R	F	M	客户类别
243	694	26	733	1	1	1	重要发展客户
537	770	23	527	1	1	1	重要发展客户
263	1024	24	661	1	1	1	重要发展客户
965	848	24	618	1	1	1	重要发展客户
126	566	22	509	1	1	1	重要发展客户
747	469	21	611	1	1	1	重要发展客户
502	675	23	618	1	1	1	重要发展客户
285	1029	26	687	1	1	1	重要发展客户
146	772	25	822	1	1	1	重要发展客户
16	1121	28	624	1	1	1	重要发展客户
920	1756	12	346	0	0	0	无价值客户
318	1578	8	183	0	0	0	无价值客户
424	1271	12	265	0	0	0	无价值客户
759	1610	9	281	0	0	0	无价值客户
135	1883	7	212	0	0	0	无价值客户
62	1649	10	329	0	0	0	无价值客户
923	1878	8	217	0	0	0	无价值客户
558	1129	9	296	1	0	0	一般客户
408	834	5	138	1	0	0	一般客户
93	1700	6	213	0	0	0	无价值客户
参照值	1163	16.4	445				

为了验证 IER 算法的可行性，本文采用 RFM 模型进行实验验证，并对比了 Enhanced-RatioRank 和 Time-PageRank 等同类算法，Enhanced-RatioRank 算法与 Time-PageRank 算法属于大规模网络图中排序查询的主流个性化 PageRank 算法，前者可以在一定程度上提高算法的查询准确度，但没有考虑内容节点时效性因素，导致查询结果时效性不高；后者由于基于时间序列机制，在查询内容时效性及查准率方面效果不错。本文实验分析主要关注算法的查准率[55]和鲁棒性。查准率越高，代表用户的目标需求与反馈结果匹配度越高，本文通过计算 IER 算法的查准率，验证了 IER 算法在客户交易网络中寻找重要发展客户方面的有效性。除了查准率，本文还关注了算法的鲁棒性。鲁棒性测试算法在数据急剧变化下的稳定性和对数据处理的有效性。本文通过模拟数据波动变化，比较 IER 算法与其他同类算法的效果，得出 IER 算法具有较高的鲁棒性和有效性。综上所述，本文通过实验验证，IER 算法不仅可以使搜索结果更偏向用户查询需求，而且还提高了查询内容的精确度，极大地提高了算法的查准率。查准率算法公式如（3.8）所示：

$$Precision = \frac{\sum_{i=1}^{n}\left(\frac{sum_s}{sum}\right)_i}{n} \tag{3.8}$$

式中，sum表示挖掘出来的总客户记录数，sum_s表示挖掘结果中的为重要客户价值的客户记录数，n为数据集的个数，为了使本文算法更具有说服力，本文取了五个大小不同的数据集，因此取$n = 5$。实验过程是由五个数据集分别对 IER 算法，Enhanced-RatioRank 算法和 Time-PageRank 算法进行测试，实验一共进行了 300 次数据挖掘，并且获取了客户价值排名最高的前十名客户和客户价值排名最低的后十名客户记录，将这些记录用 RFM 模型来验证分析，利用查准率计算公式得到挖掘结果的查准率，见图 3-6。

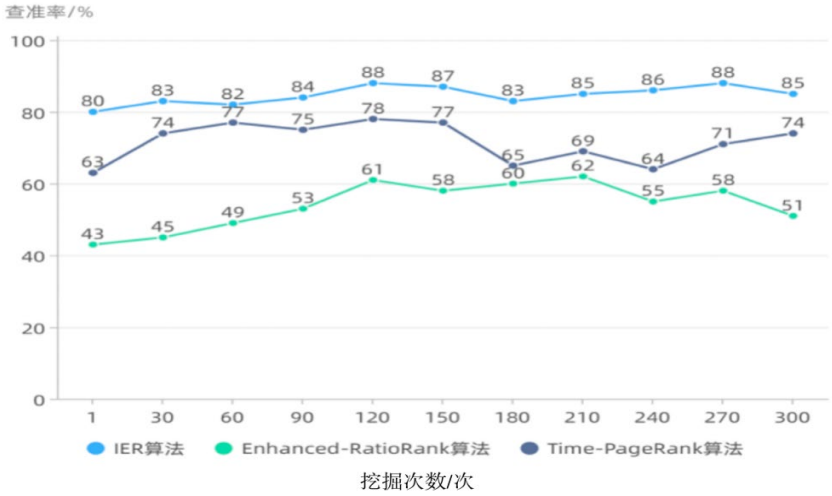


图 3-6 各算法挖掘结果的查准率

本文算法的鲁棒性测试：通过对数据集的数据量波动变化来分别对 IER 算法，Enhanced-RatioRank 算法和 Time-PageRank 算法进行测试，从而可以研究出

在不同数据集记录数对算法的查准率的影响情况，结果见图 3-7。

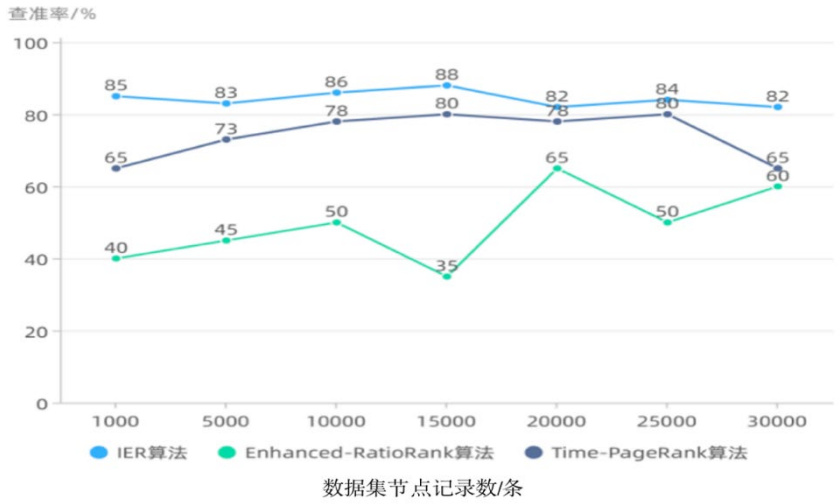


图 3-7 各算法在不同数据集记录数下的查准率

从图 3-7 实验结果可以得出：随着数据集记录数的不断增加，从 1000 条数据记录到 30000 条数据记录（单位：条），Enhanced-RatioRank 算法与 Time-PageRank 算法的查准率不高，而且随着数据集的不断增大，同类算法的曲线波动幅度比较大，代表同类算法查准率的波动性比较大，而本文算法查准率的平均值保持在 84%左右，且本文的 IER 算法对于挖掘结果的查准率一直是处于一个稳定的状态，则一定程度上也说明了 IER 算法较其他算法的鲁棒性更好。

为了使本文提出的算法更具有说服力，本文采用实验数据中的 Bank_tracsactions3、Bank_tracsactions4 和 Bank_tracsactions5 三个数据集对 IER 算法进行消融实验对比分析，且采用算法的查准率来评估算询优化效果。首先，本文提出的 IER 算法的两种变体：ER@weight 与 ER@weight-freq 算法，将 ER 算法和两种变体算法与本文 IER 算法在不同的数据集上进行实验对比分析，如图 3-8 至图 3-10 所示，从实验结果可以分析出：ER、ER@weight、ER@weight-freq 及本文 IER 算法在三个不同数据集里的算法性能是以阶梯形式提高的。这也进一步验证了本文提出的 IER 算法的有效性。

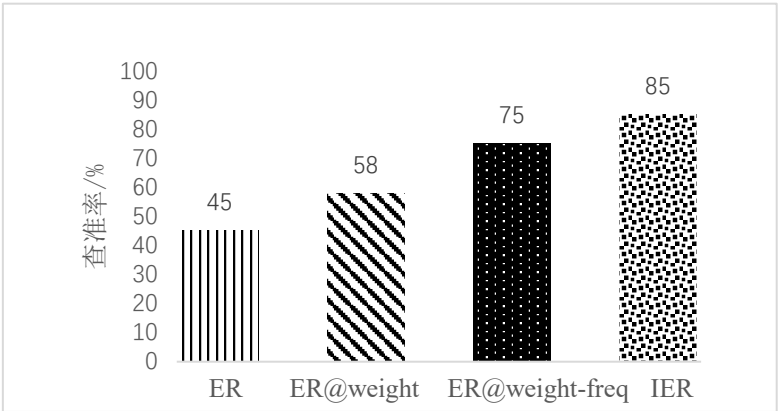


图 3-8 IER 算法与其他变体算法在 Bank_tracsactions3 上的查准率

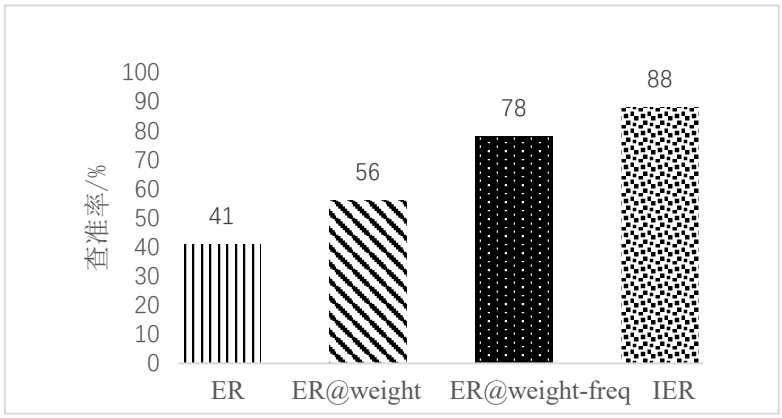


图 3-9 IER 算法与其他变体算法在 Bank_tracsactions4 上的查准率

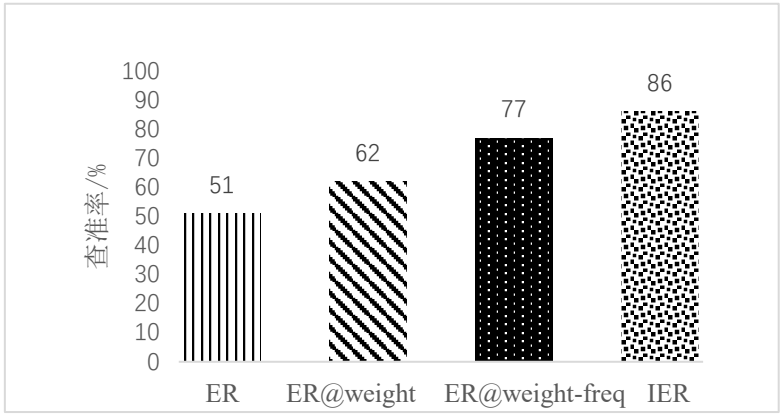


图 3-10 IER 算法与其他变体算法在 Bank_tracsactions5 上的查准率

3.4 本章小结

针对个性化 PageRank 算法在大规模网络图排序查询中存在查询结果精确度及个性化搜索时效性不高的问题，本文提出了一种基于多特征因子的个性化 PageRank 查询优化算法（Improved Enhanced-RatioRank，IER），该算法主要通过主题漂移优化、内容相关度增强和内容时效性平衡三个模块对个性化 PageRank 算法进行查询优化，在主题漂移优化模块中本文首先引入了权重因子，权重因子可以解决主题漂移问题，使其用户根据主题关键词查询时尽可能减少垃圾网页的情况。其次针对权重因子进行了优化，通过设置α和β权重比调节因子，利用调节权重比因子α和β来调控链入权重及链出权重的影响度，更好的解决 PPR 值下沉问题；在内容相关度增强模块中本文利用频数因子改善用户查准率，使用户搜索目标需求与反馈结果的匹配度更高，提高查询内容相关度；内容时效性平衡模块本文主要利用时间有效性因子来提高查询内容的时效性，使其查询时效性高的网页节点上升，时效性低的网页节点下沉。最后本文采用了五个大小不同数据集将 IER 算法与 Enhanced-RatioRank、Time-PageRank 等同类算法进行实验对

比，实验结果表明 IER 算法不仅可以使搜索结果更偏向用户查询需求，而且还提高了查询内容的精确度，极大地提高了算法的查准率。

第 4 章 基于分布式图划分的个性化 PageRank 高效计算

本章主要围绕本文研究内容所涉及的基于分布式图划分的个性化 PageRank 高效计算进行详细阐述，首先介绍了个性化 PageRank（Personalized PageRank，PPR）算法在大规模网络图中高效计算问题的一些主流算法，其次描述了该类算法所存在的一些问题，然后对本文提出的 BAESD 算法进行了详细的介绍，最后利用三个指标（算法准确度、算法的计算效率和存储空间）对本文算法进行实验对比分析。

4.1 问题提出

个性化 PageRank 算法在图计算领域热度很高，随着网络节点图规模的不断增加，网络图中的数据链接关系也变得错综复杂，这样会使图计算的工作量变得复杂多样，因为数据的规模性和复杂性不仅给数据预处理带来了较大麻烦，而且也会显著影响算法的计算效率，同时还要求硬件设备具有极高的计算能力，从而大大增加了计算成本。为了解决这个问题，现在主流优化算法的思路均是将大规模图上的计算分割到多个子图上进行计算，也就是通过图划分算法来对数据进行处理。图划分算法主要可以分为三类：传统图划分算法[31-34]、流式图划分算法[35-36]和分布式图划分算法[18,41,42]。

传统图划分算法只能适应小规模图数据，流式图划分算法需要在一定约束条件下才能进行子集划分，这样会导致划分质量不高以及较高的割边率，由于分布式计算可以处理海量数据，因此分布式图划分算法非常适用于大规模图数据的划分，然而该算法也存在一些问题，比如在进行分布式图划分的过程中产生了大量的子图，子图与子图之间依赖性十分复杂，导致子图之间的通信频率高，严重影响算法的计算效率，所以本文基于分布式图划分算法展开算法研究及改进。

4.2 算法描述

针对现在主流的分布式图划分算法在进行图划分过程中产生的子图之间的通信频率高导致计算效率不高的问题，本文提出了一种基于分布式图划分的个性化 PageRank 改进算法（简称 BAESD 算法），本文首先提出了一种基于 Boruvka 算法的分布式图划分算法，完成局部子图的构建与平衡化，将大规模网络图划分为多个局部平衡子图，使个性化 PageRank 算法的迭代计算转移到局部子图数据

中，其次提出了一种相邻边缘结构（Adjacent Edge Structure，AES），AES 结构主要从两个方面对算法进行优化：一方面是提高图划分效率，尽可能减少图数据处理的时间；另一方面是该结构可以减小局部分区子图与子图之间的依赖关系密度，最后，本文利用 DSP 模型来加速个性化 PageRank 算法的迭代计算，从而进一步提高个性化 PageRank 算法在大规模图上的计算效率。将本文算法在四个不同大小数据集上与 PM、RA 和 TDA 等主流算法[18]进行实验对比，实验结果表明本文所提算法极大提高了个性化 PageRank 算法在大规模网络图上的计算效率。BAESD 算法主要有两大核心部分组成：（1）局部子图构建与平衡化模块；（2）相邻边缘结构（Adjacent Edge Structure，AES）。本文将在下文对这两个核心部分展开详细介绍，BAESD 算法框架图如图 4-1 所示：

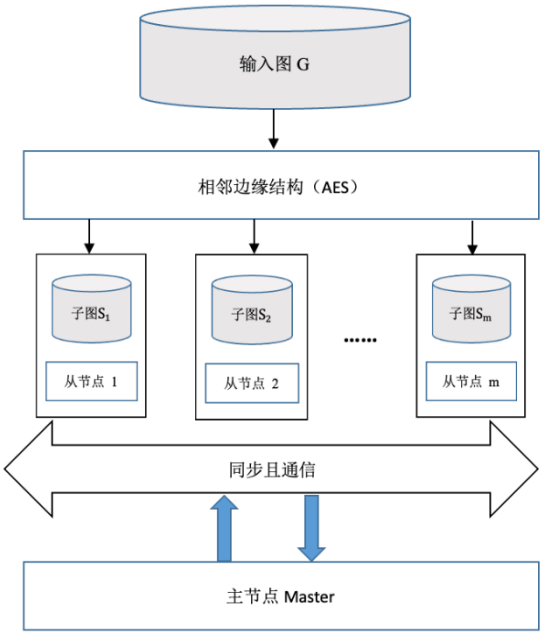


图 4-1 BAESD 算法框架图

4.2.1 局部子图构建与平衡化

本文提出了一种基于 Boruvka 算法的分布式图划分算法，该算法首先把大规模图分割成若干个局部子图，其主要的思路就是将 PPR 算法在大规模图上的计算转移到多个局部子图上计算，这样可以首先将大量无关图节点数据和冗余数据处理掉，其次本文对构建的局部子图根据一定的约束条件进行平衡化处理，尽可能减少局部子图之间的通信量，从而优化算法的计算效率。下面本文将对局部子图的构建与平衡化展开详细描述。

（1）局部子图构建

受 Boruvka 算法[56]最小生成树思想的启发，本文首先对大规模图中的每个

顶点进行初始化工作，目的是使大规模图在分布式图划分的过程中可以形成每个独立的子图集，方便后续对独立子图的约束性操作；其次初始化后的点可以看做一个独立子集，找到该子集存在最短路径且相关联的另一个子集，将两个子集合并。重复上述一个过程，构建最终若干个局部子图。这里使用使用边的权值来度量两个点之间的距离，由于无权图不存在具体的权重值，无法进行距离度量，因此，本文将有连接的两点的距离视作 1，而没有连接的两点的距离视作∞。但是，如果一个点有多个相邻点，则这些相邻点之间的距离都是 1，这可能会导致错误的划分结果。为了解决这个问题，本文需要使用其他方法来度量两点之间的距离。本文采用共同邻点比例方法[57]，即使用两个顶点的共同邻点数与它们邻点数的乘积的比值来表示它们的距离。u和v两点之间的距离表达公式如（4.1）所示：

$$\mathrm{W(u,v)}=\frac{\mathrm{commonNeihbours_{uv}}+2}{\sqrt{(\mathrm{D_u}+1)\times(\mathrm{D_v}+1)}} \tag{4.1}$$

式中，$\mathrm{commonNeihbours_{uv}}$表示顶点$v$和$u$的公共邻点数，$D_u$和$D_v$表示顶点$u$和$v$的度数，两点之间的共有邻点数可以通过计算边的三角形个数来得到。

（2）局部子图平衡化

构建局部子图的主要思路就是将 PPR 算法在大规模图上的计算转移到多个局部子图上计算，由于在分布式图划分中局部子图与子图之间需要一定的数据通信量，本文将构建的局部子图进行平衡化处理，生成若干个规模相等的平衡局部子图，尽可能减少局部子图之间的通信量，从而优化算法的计算效率。在一定约束条件下，算法经过多次迭代可以产生若干个规模一样的独立子集，每个独立子集代表一个局部子图，本文假设独立子集个数是m,独立子集的个数达到阈值后停止迭代，如果独立子集的个数没有达到阈值或者无限逼近于这个阈值，那么需要对独立子集进行再平衡化，这里引入邻近子图和邻点相似度和值的定义[58]。本文算法执行后会生成的m个树称为邻近子图，这里用符号φ表示，其中，邻近子图的顶点数量为$|V_\varphi|$，边数量为$|E_\varphi|$，当邻近子图的数量达到$|V_\varphi|=n/m$,称之为最大邻近子图，用符号φ_{max}表示。如果在$|V_\varphi|$集合中，一个顶点v有n个邻点，每个顶点分别为$v_1 \in |V_\varphi|, v_2 \in |V_\varphi|, \ldots, v_n \in V_\varphi|$，这时，点$v$的邻点相似度和值具体公式如（4.2）所示：

$$T_v=\sum_{i=1}^{n} w(v,v_i) \tag{4.2}$$

式中，T_v表示点v的邻点相似度和值，$w(v,v_i)$表示局部子图中的顶点v与顶点v_i之间的距离。如图 4-2 所示：这里描述了一个局部子图构建与平衡化的一个过程，大规模图G_1首先经过分布式图划分成两个局部子图S_1和S_2，将 PPR 算法在大规模图上的计算转移到多个局部子图上计算，$P_k^{S_1}$表示 PPR 算法在子图S_1迭代 k 次后的 PPR 值，两个局部子图S_1和S_2在图划分的过程需要一定的同步数据通信，图G_1在分布式划分子图S_1和S_2的过程中会根据邻电相似度准则来约束局部

子图的划分，这样可以在一定程度上减少局部子图之间的通信量，PPR 算法的迭代计算会在收敛后停止迭代，最后形成的两个 PPR 向量会传递给分布式计算节点M_3。

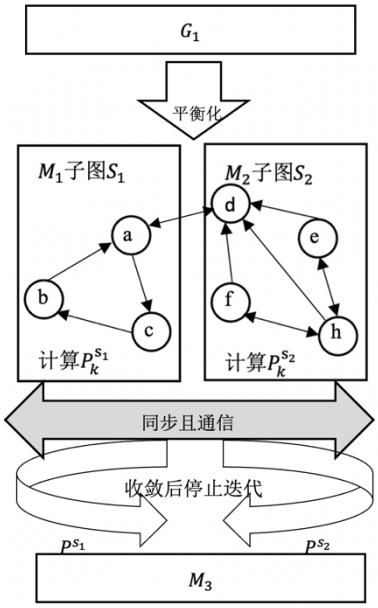


图 4-2 局部子图的构建与平衡化

本文提出了分布式图划分算法，该算法受 Boruvka 算法最小生成树思想的启发，首先将大规模图分割成若干个局部子图，之后对构建子图进行平衡化处理。如算法 4-1 所示，面是局部子图构建与平衡化算法的具体描述：

算法 4-1：局部子图构建与平衡化算法

Input: 网络$G=(V,E)$，平衡约束条件 balance,子图数目 m,每个子图的分片数目n

Output: m个子图$\pi_1,\pi_2,\dots,\pi_m$

1. for all$(u,v)\in E$ do
2. $w(u,v)$←计算顶点 u 和 v 的相似度
3. φ←Boruvka (G,w,k)
4. G_1←φ｛将φ中的点映射为顶点序列｝
5. for i=0 to n do
6. $L_{(i)}$←G_1｛ 将每个局部子图的顶点序列划分为 m 个子片｝
7. $L'_{(i)}$←balance (G, G_1, m, n)｛局部子图平衡化｝
8. for all i=0 to r do
9. else
10. φ'←$L'_{(i)}$
11. return $\{\pi_1,\pi_2,\dots,\pi_m\}$

4.2.2 相邻边缘结构

在分布式计算中，图划分是一项常见任务，但其计算复杂度高，耗费时间和资源较多。由于PPR算法在大规模图中是通过不断的迭代计算来求解PPR向量，本文提出了一种相邻边缘结构（Adjacent Edge Structure，AES），该结构主要从两个方面对算法进行优化：一方面是提高图划分效率，尽可能减少图数据处理的时间；另一方面是该结构可以减小局部分区子图与子图之间的依赖关系密度，从而减少算法的迭代计算次数，提高计算效率。

对于融入 AES 的分布式图划分算法，在每次分配一个顶点v后，该顶点v的邻点$N(v)$会被设置为键，而它的子集信息$P(v)$是保存在相邻边缘结构中的值（$N(v)$: $S(v)$）。这里的计算过程可以参考表 4-1 和图 4-3 进行理解。在时间 T 内，顶点v_1被分配给子集S_1，其子集信息（S_1）被设为值，并且v_1的邻点（v_2和v_3）被设置为键，然后将它们保存到相邻边缘结构中（v_3: S_1，v_3: S_1）。在时间 T+1 内，与顶点v_1距离更近的顶点v_3加入，v_3的子集信息被添加到相邻边缘结构中（v_3: S_1），因为v_3邻点的子集信息已经保存在相邻边缘结构中（v_3: S_1），为了遍历S_1中距离最近的顶点v_6和v_2，在S_1中加入与S_1距离更近的顶点v_2，然后重复之前的过程，直到分配完所有的顶点。

表 4-1 从时间 T 到时间 T+1 在划分图 G 的过程中缓存数据的内容

时间	顶点 ID	缓存数据的内容
T	1	{v2: S1; v3: S1}
T+1	3	{v6: S1; v2: S1}
T+2	2	{v5: S2; v4: S3}
T+3	5	{v5: S2; v2：S2;v4: S2,S3}
T+4	4	{v4: S3}
T+5	6	Null

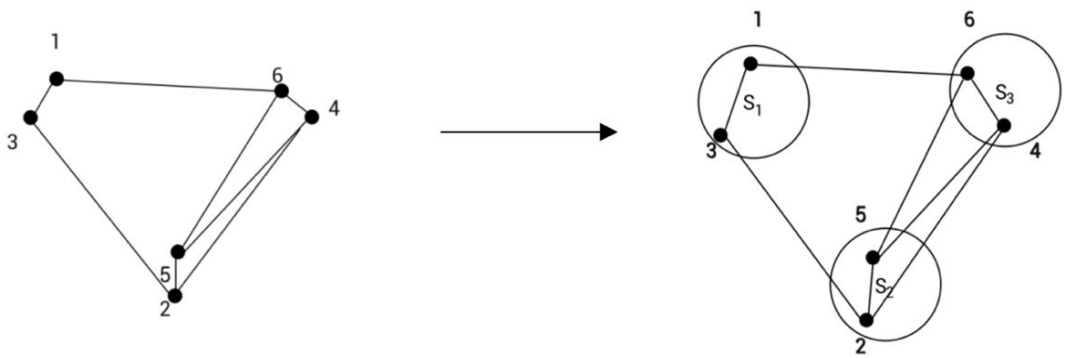


图 4-3 示例图 G 划分为三个子图过程

本文使用三个操作来管理本文的缓存数据：

更新操作（updated（$N(v)$，$S(v)$））：每次分配一个顶点v时，本文将执行更新操作（updated($N(v)$，$S(v)$））来更新该顶点的邻点信息。在该过程中，邻点$N(v)$会被设置为键，而v的子集信息则被设置为值。以顶点v_1为例，假设v_1的邻点是v_2，算法首先会在缓存数据中查找是否有v_2的条目作为键，如果有的话，会被直接附加到子集信息$S(v)$中。

删除操作：间接删除操作。$item(v)$表示以v为键的条目，顶点v被分配后，$item(v)$作为冗余数据需要被删除，如果缓存数据大小超过了内存容量，那么应该删除值较小的条目，因为值小的条目对划分质量的影响比较小。以顶点v_2为例，假设v_2的邻点是v_3，算法首先会根据数据缓存表的键值对信息查询到顶点v_2的具体位置，然后再根据顶点v_2所在的子集信息将该顶点删除，最后再平衡化缓存数据中所有顶点。

添加操作：当需要在缓存数据表中添加一个局部子图顶点$S(v)$，首先会根据数据缓存表的键值对信息($N(v)$，$S(v)$来索引顶点添加的位置$S(v)'$，其次根据顶点相似度准则直接附加到子集信息$S(v)'$中。如算法 4-2 所示，以下是本文提出的 AES 算法的具体描述：

算法4-2：AES算法

Input: Graph G, number of subsets k

Output: A partition of V into k parts $S(v)$

1. FOR v in V_{random} //遍历图节点
2. update($N(v), P(v)$) //更新操作
3. $S(v) \leftarrow$ update$(item(v))$
4. IF $item(v)$ in $dynamiccache()$
5. Delete($item(v)$) //删除操作
6. $S(v) \leftarrow$ Delete$(item(v))$
7. IF $item(v)$ in $dynamiccache()$
8. Add($item(v)$) //添加操作
9. $S(v) \leftarrow Add(item(v))$
10. END IF
11. IF the dynamic cache data size exceeds the memory capacity //缓存溢出判断
12. $item(v)' \leftarrow item(v)$
13. Delete($item(v)'$)
14. RETURN $S(v)$

在处理大规模图数据时，内存和时间的限制往往是本文需要解决的问题。为了优化图数据的处理，提高算法的的计算效率，本文采用邻接矩阵存储输入图G，这样可以在数据预处理中减少冗余及不必要的图节点，首先将下三角邻接矩阵信息保存在其中，其次本文只需要输入上三角邻接矩阵信息，输入图数据相比原始数据减少了一半。这种方式不仅大幅度提高了内存利用率，同时能够有效提高图划分的效率。邻接矩阵存储方式是一种经典的数据结构，在图论中具有广泛的应用。通过采用邻接矩阵存储的方式，本文能够快速地实现图数据结构，并进行高效的图数据运算。如图 4-4 所示：

$$G(V,E)=\begin{bmatrix}0&1&1&0&0&0\\1&0&0&1&1&0\\1&0&0&0&0&1\\0&1&0&0&1&1\\0&1&0&1&0&1\\0&0&1&1&1&0\end{bmatrix}\longrightarrow G(V,E)=\begin{bmatrix}0&1&1&0&0&0\\0&0&0&1&1&0\\0&0&0&0&0&1\\0&0&0&0&1&1\\0&0&0&0&0&1\\0&0&0&0&0&0\end{bmatrix}$$

图 4-4 图 G 的邻接矩阵和上邻接矩阵

同时，本文提出的相邻边缘结构主要从两个方面对算法进行优化：一方面是提高图划分效率，尽可能减少图数据处理的时间；另一方面是该结构可以减小局部分区子图与子图之间的依赖关系密度。由于 PPR 算法在大规模图中是通过不断的迭代计算来求解 PPR 向量，因此算法的计算效率主要依赖于算法的迭代次数，本文在最后引入了一个 DSP（delta-stepping synchronous parallel）多步前进同步并行计算模型[59]，该模型将每一步的局部计算变成了多步的局部计算，而不涉及节点之间的通信，它只进行一些局部数据的更新，因此本文最后利用 DSP 模型来加速 PPR 算法，尽可能减少算法的迭代次数，从而进一步提高 PPR 算法的计算效率。

4.2.3 算法实现

本文提出了一种基于分布式图划分的个性化 PageRank 高效计算算法（简称 BAESD 算法），该算法首先完成局部子图的构建，将算法在大规模图上的计算转移到多个局部子图上计算，由于在分布式图划分中局部子图与子图之间需要一定的数据通信量，本文将构建的局部子图进行平衡化处理；其次提出了一种相邻边缘结构（AES），本文提出的相邻边缘结构主要从两个方面对算法进行优化：一方面是提高图划分效率，另一方面是该结构可以减小局部分区子图与子图之间的依赖关系密度；最后本文利用 DSP 模型来加速 PPR 算法，尽可能减少算法的迭代次数，从而进一步提高 PPR 算法的计算效率。如算法 4-3 所示，以下是 BAESD 算法的具体描述；

算法 4-3：BAESD算法

Input: $G=(V，E)$，子图数目k

Output: k个子区$\pi 1,\pi 2,\ldots,\pi k$

1. for all$(u,v)\in E$ do
2. Map$(key,value)$
3. 计算顶点度(u,v)
4. $key'\leftarrow(u,v)$
5. if $value\supseteq$ 度数 do
6. $value'\leftarrow(D_u,D_v)$//顶点$u$和$v$的度数赋值给$value'$
7. else
8. $value'\leftarrow w(u,v)$
9. Reduce $(key',values')$
10. for $value\in values'$
11. if $value\supseteq$ 度数 do
12. $\text{commonNeighbors}_{uv}\leftarrow$ 顶点u和v边上三角形的个数
13. return $w(u,v)$
14. for v in V_{random}
15. update$\big(N(v),P(v)\big)$
16. if $item(v)$ in $dynamiccache()$
17. Delete $(item(v))$
18. if the dynamic cache data size exceeds the memory capacity
19. return $S(v)$
20. $C_n\leftarrow S(v)$
21. 选择距离最近的两个子图S_i,S_j
22. $S_i\leftarrow S_i\cup C_v$,$C_n\leftarrow C_n\backslash S_j$， n=n-1
23. if $|v_{S_i}|>|v_{\psi_{max}}|$ do
24. repeat
25. 选择S_i中T_v值最小的的顶点v_i
26. $S_i\leftarrow S_i\backslash v_i$， n=n+1， $C_n\leftarrow v_i$
27. until $|v_{S_i}|>|v_{\psi_{max}}|$
28. DSP（$S_i,S_2,\ldots,S_i$）
29. return $\pi 1,\pi 2,\ldots,\pi k$

4.3 实验分析

为了评估本文算法的性能，实验主要是从三个指标进行评估：算法的准确度、算法的计算效率和算法的存储空间。这三个指标可以从多个维度对本文算法进行性能分析，在本次实验中，本文需要利用一个合理的临界值来终止 PPR 的迭代计算，如果连续两次迭代计算结果的差值的 F 范数[60]小于临界值 0.001，则停止每个子图上的迭代并输出最终结果向量。

4.3.1 实验环境与数据集

实验硬件环境：Intel CPU，16G 内存，500GB 的硬盘，本文算法是在 Hadoop 平台上搭建的 4 台服务器构成的集成环境中进行的，其中有 1 个 master 和 3 个 slaver。本文实验数据均从斯坦福数据网站下载的数据集，其具体信息如表 4-2 所示：

表 4-2 实验数据

数据集	节点数	边数
Slashdot	82168	948464
Gemsec_facebook	134833	1380293
web_Stanford	281903	2312487
web_BerkStan	685230	7600595

4.3.2 实验结果分析

与本文算法作对比实验的主流算法有 PM、RA 和 TDA 算法，PM 算法是在原图上计算的精确算法，该算法在小规模图上性能良好；RA 算法是一种可达子图的算法，该算法是对网络图数据进行预处理，依照某种规则删减冗余节点与边，从而缩小数据规模，该算法的局限性是无法在动态网络图和实时数据中实现；TDA 算法是一种传统的分布式算法，该算法划分精确度比较高，但是划分时间也是随着迭代次数成正比。

（1）算法准确度

在实验过程中，因为实验机器精确度计算的差异和迭代阈值的设定，会导致算法在各个子图中的计算结果存在一定的误差。由于 PM 算法是在原图中进行 PPR 计算，该算法不会损失精确度，因此 PM 算法不参与实验比较，本文利用余弦相似度算法来计算本文所提出的算法及同类算法与 PM 算法计算结果向量的相似度[18]，如图 4-5 和图 4-6 所示，本文所提算法的计算结果存在的误差对最终结果的影响是很小的，且该算法的准确度在同类算法中也是占有优势的。

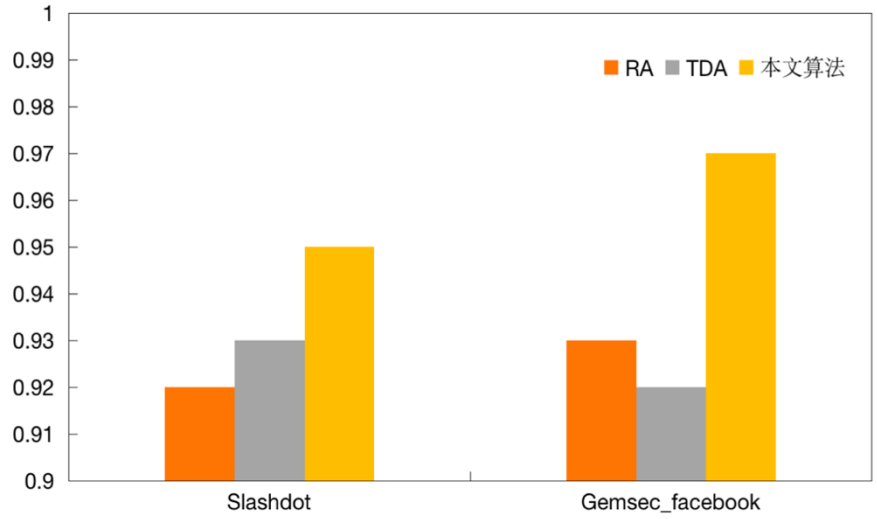


图 4-5 每种算法在 Slashdot 和 Gemsec_facebook 数据集上的准确度

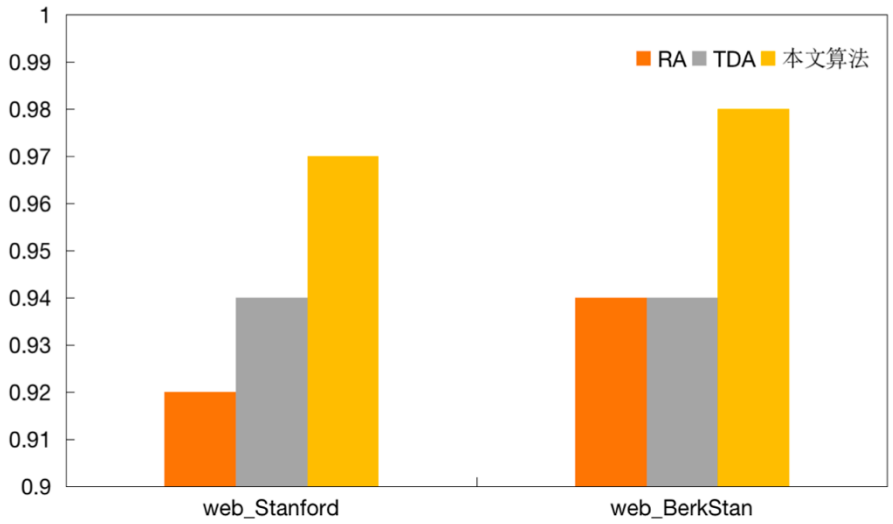


图 4-6 每种算法在 web_Stanford 和 web_BerkStan 数据集上的准确度

（2）算法加速比与平均执行时间

本文首先对 DSP 加速效果与Δ、γ和β之间的关系进行了实验验证，因为本文主要验证β（分区间的依赖关系密度）对 DSP 加速效果的影响，所以本文固定γ和Δ，变化β，横坐标是β，纵坐标是本文算法分别与 PM 算法、RA 算法和 TDA 算法的加速比$speedup$（本文 A 与 B 加速比的定义为 B 的迭代次数与 A 的迭代次数的比值），实验结果如图 4-7 至图 4-9 所示：在四个不同大小的数据集里，$speedup$随着β的增大而下降，说明子图间的依赖关系程度增加后，本文算法与 PM 算法、RA 算法和 TDA 算法的加速比在下降，但加速比仍然大于 1，且在β较小时，也就是子图间的依赖关系程度较小时，本文算法与其他算法的加速比能够达到 3 到 5 倍，也就是说，在控制变量的情况下，同类算法需要迭代 3 到 5 次完成计算，本文算法只需要迭代 1 次。实验结果进一步验证了减弱数据分区间的

依赖关系密度（即减小β），本文算法的收敛性可以得到有效加速。

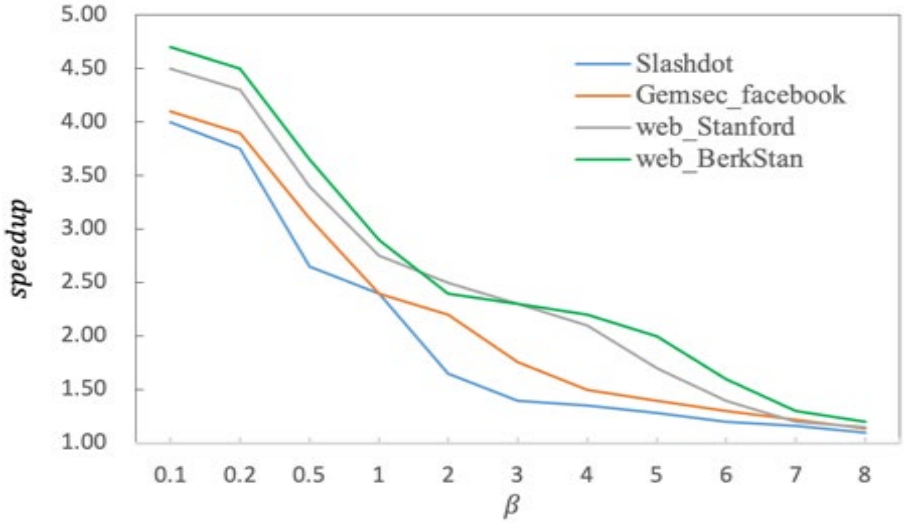


图 4-7 本文算法与 PM 算法在每个数据集上的加速比

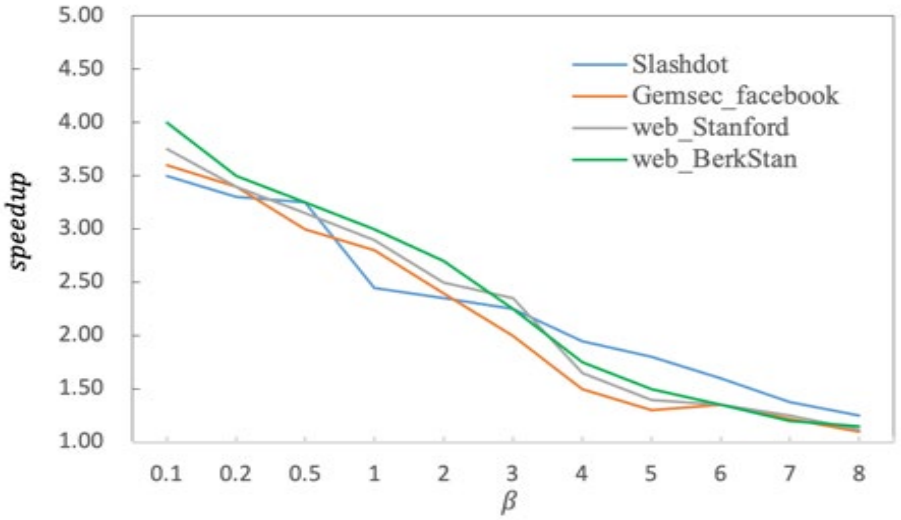


图 4-8 本文算法与 RA 算法在每个数据集上的加速比

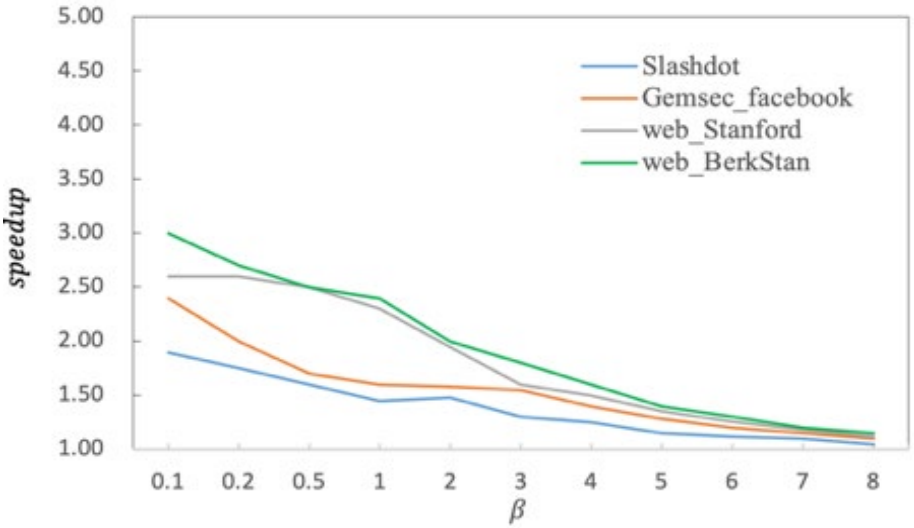


图 4-9 本文算法与 TDA 算法在每个数据集上的加速比

本文选取了 100 个样本结点，本文只需要利用各算法计算出 100 个样本结点的 PPR 的平均花费时间，将本文提出的算法与 PM 算法、RA 算法和 TDA 算法进行比较，见图 4-10 和图 4-11，在图规模不是很大时，PM 算法优于 RA 算法和 TDA 算法，因为数据集所占用存储空间不大，可以直接放入内存中计算，而当图规模增大时，RA 算法和 TDA 算法是优于 PM 算法的，因为内存空间不足，而分布式计算和可达子图计算的优势就显现出来了，本文算法在四个不同数据集中平均执行时间是最少的，因为本文算法需要占用一定的分布式计算资源，所以在小图规模中优势不是很明显，但总体上本文算法的计算效率是最高的。见图 4-12 和图 4-13，当图规模比较大时，PM 算法由于内存空间严重不足导致计算效率低，RA 算法因为可达子图的规模过大，从而在一定程度上影响了算法效率，TDA 算法由于没有考虑数据分区间的依赖关系密度，从而会使得子图之间的通信量很大，而本文算法通过减小计算规模，加速迭代计算的优势就显现出来了。

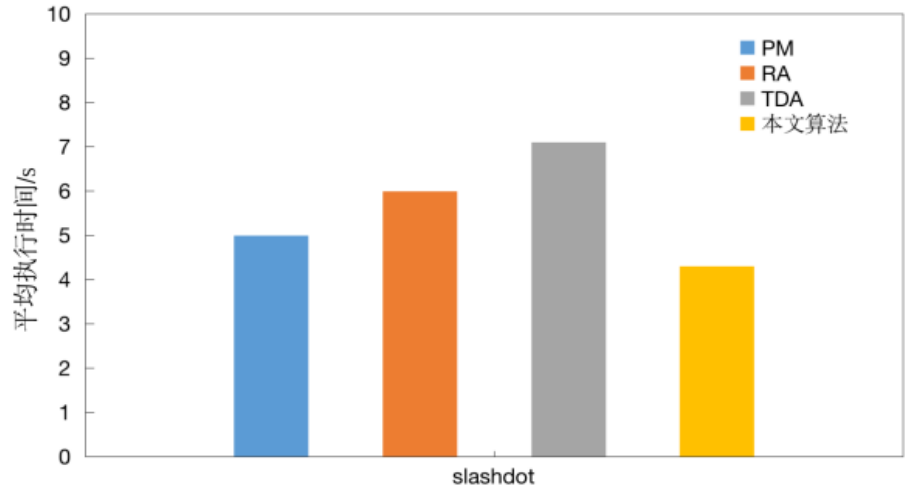


图 4-10 各算法在 Slashdot 数据集上的平均执行时间

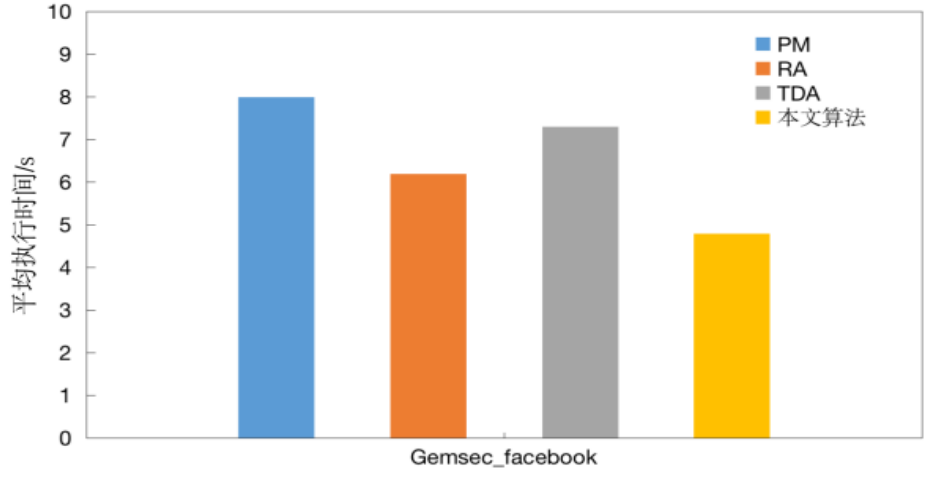


图 4-11 每种算法在 Gemsec_facebook 数据集上的平均执行时间

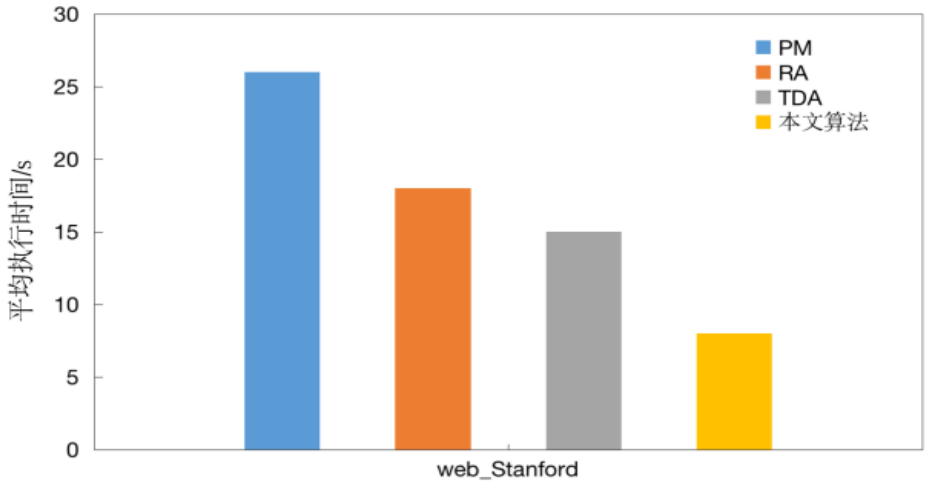


图 4-12 每种算法在 web_Stanford 数据集上的平均执行时间

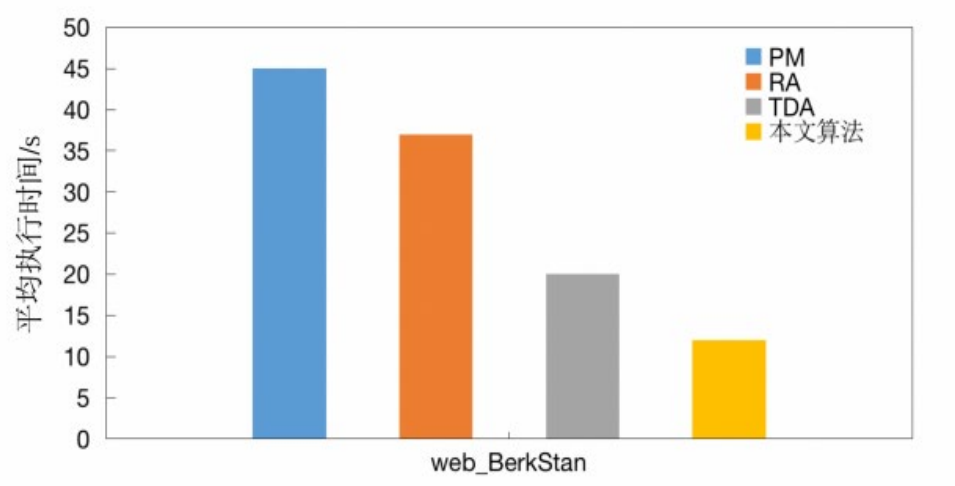


图 4-13 每种算法在 web_BerkStan 数据集上的平均执行时间

（3）算法存储空间

下面通过实验，使用 web_Stanford 数据集进一步测试 PM 算法、RA 算法、TDA 算法和本文算法在计算过程中所占用空间大小，这里的空间主要由两个部分组成：原图数据集所占用的空间和预处理所需要的额外空间。实验结果如表 4-3 所示：因为 TDA 算法是传统分布式计算方法，该算法本质上是属于用分布式计算资源来换取时间的算法，所以该算法所需要更多的额外空间，PM 算法因为是在原图迭代计算的方法，该算法需要的额外空间较少，RA 算法是基于随机游走准则，从而缩减了数据规模，所以在一定的程度上减少了原图数据集空间，但是可达子图规模大到一定程度时，该算法所占用的空间也会很大。本文提出了一种基于 Boruvka 算法的分布式图划分算法，并且提出了一种新型缓存数据结构，称为相邻边缘结构（AES），根据 AES 的特点，对原始数据的处理，输入图数据相比原始图数据减少了一半，所以该算法所占用的空间是最少的。

表 4-3 每种算法所占空间的大小

算法	原始图数据占用的空间/MB	用于预处理的额外空间/MB	总空间/MB
PM	37.96	65.12	103.08
RA	37.96	43.01	80.97
TDA	37.96	121.33	159.29
本文算法	37.96	22.24	60.2

为了使本文提出的算法更具有说服力，本文采用实验数据中的三个数据集对 BAESD 算法进行消融实验对比分析，且采取了算法执行 100 个节点的平均执行时间来评估算法效率。首先，本文提出的 BAESD 算法的两种变体：DA-AES 与 DA 算法，两种变体算法与本文 BAESD 算法在 web_Stanford 和 web_BerkStan 数据集上的性能表现最为显著，而在另外一个数据集 Slashdot 上的性能提升较少。这是因为在 web_Stanford 和 web_BerkStan 两个数据集上的图节点规模比较大，分布式计算的优势显现出来了。如图 4-14 至图 4-16 所示，从实验结果可以分析出：DA、DA-AES、BAESD 算法在三个不同数据集里的算法性能是以阶梯形式提高的。这也进一步验证了本文提出的 BAESD 算法的有效性。

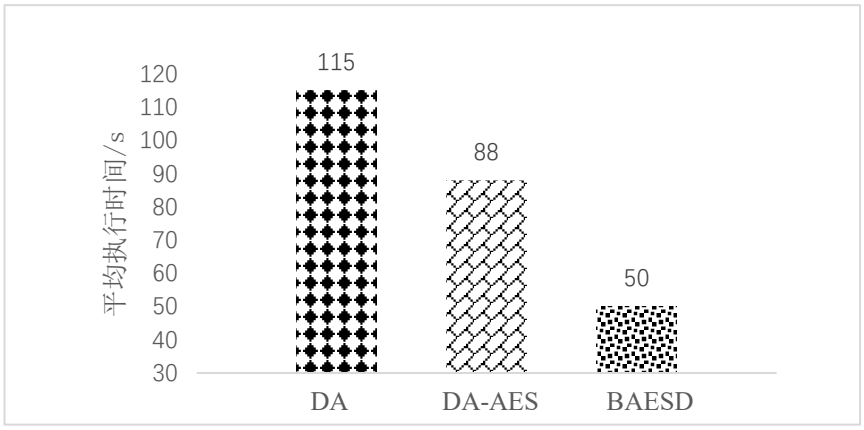


图 4-14 BAESD 算法与 DA、DA-AES 算法在 web_Stanford 的计算效率

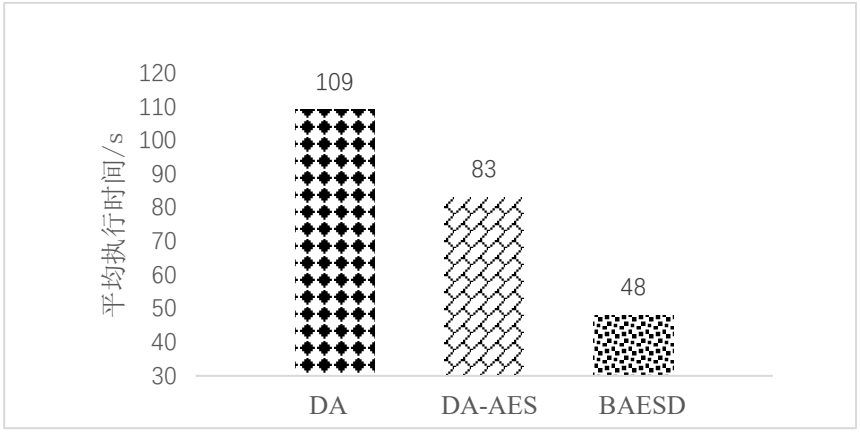


图 4-15 BAESD 算法与 DA、DA-AES 算法在 web_BerkStan 的计算效率

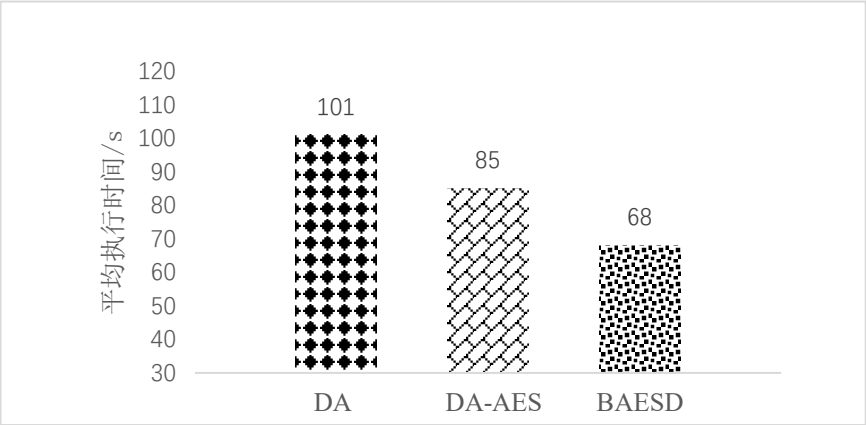


图 4-16 BAESD 算法与 DA、DA-AES 算法在 Slashdot 的计算效率

4.4 本章小结

针对在大规模网络图中个性化 PageRank 算法的计算效率问题，本文章节首先简要描述了提高个性化 PageRank 算法在大规模网络图中计算效率的必要性，其次概述了在大规模网络图中个性化 PageRank 算法的计算效率问题上的一些主流算法，并且针对这类算法的特点描述了它们存在的一些问题，针对这些问题，本章详细介绍了本文提出的 BAESD 算法，本文首先提出了一种基于 Boruvka 算法的分布式图划分算法，完成局部子图的构建与平衡化，将大规模网络图划分为多个局部平衡子图，使个性化 PageRank 算法的迭代计算转移到局部子图数据中，其次提出了一种相邻边缘结构（Adjacent Edge Structure，AES），AES 结构主要从两个方面对算法进行优化：一方面是提高图划分效率，尽可能减少图数据处理的时间；另一方面是该结构可以减小局部分区子图与子图之间的依赖关系密度，最后，本文利用 DSP 模型来加速个性化 PageRank 算法的迭代计算，从而进一步提高个性化 PageRank 算法在大规模图上的计算效率。本文算法在四个不同大小数据集上与 PM、RA 和 TDA 等主流算法进行实验对比，实验结果表明本文所提算法极大提高了个性化 PageRank 算法在大规模网络图上的计算效率。

第 5 章 总结与展望

5.1 总结

随着计算机网络技术的快速发展，大规模网络图已经应用非常广泛，例如社交网络、个性化推荐、图形划分和搜索查询等。由于网络规模图数量的急剧增加，这样不仅会严重降低个性化 PageRank 算法的计算效率，而且也会使算法的查询结果与用户目标需求的匹配度不高。因此，如何在大规模网络图中实现个性化 PageRank 算法的高效计算以及在排序查询问题上提高算法的查询精确度成为近些年研究的热点问题。

本文的研究目标是是针对个性化 PageRank 算法，面向对象是大规模网络图，主要目的是在大规模网络图中实现该算法的高效计算以及在排序查询问题上提高算法的查询精确度。本文的主要研究工作如下：

（1）在本文的研究中，本文对个性化 PageRank 算法展开了研究，该算法在基于 PageRank 算法的基础上，考虑了用户查询的相关性。由于个性化 PageRank 算法的基本概念和计算方法也与 PageRank 算法有很多相通之处，所以本文也对 PageRank 算法的基础知识做了简要的介绍，并对该算法的优势与不足之处进行了总结分析。这些知识为本文深入探究该算法提供了重要的基础。

（2）通过阅读大量的相关知识参考文献，总结概述了与个性化 PageRank 算法相关的基础概念，针对大型网络规模图中个性化 PageRank 算法所面临的一些问题，本文提出了相应的解决方法及对该算法展开了深入剖析。

（3）针对 Enhanced-RatioRank 算法存在 PPR 值下沉、查询内容相关性低和搜索内容时效性低的问题，本文提出了一种基于多特征因子的个性化 PageRank 查询优化算法（Improved Enhanced-RatioRank，IER），该算法主要通过主题漂移优化、内容相关度增强和内容时效性平衡三个模块对个性化 PageRank 算法进行查询优化，在主题漂移优化模块中本文首先引入了权重因子，权重因子可以解决主题漂移问题，使其用户根据主题关键词查询时尽可能减少垃圾网页的情况。其次针对权重因子进行了优化，通过设置α和β权重比调节因子，利用调节权重比因子α和β来调控链入权重及链出权重的影响度，更好的解决 PPR 值下沉问题；在内容相关度增强模块中本文利用频数因子改善用户查准率，使用户搜索目标需求与反馈结果的匹配度更高，提高查询内容相关度；内容时效性平衡模块本文主要利用时间有效性因子来提高查询内容的时效性，使其查询时效性高的网页节点上升，时效性低的网页节点下沉。最后本文采用了五个大小不同数据集将 IER 算法

与 Enhanced-RatioRank、Time-PageRank 等同类算法进行实验对比，实验结果表明 IER 算法不仅可以使搜索结果更偏向用户查询需求，而且还提高了查询内容的精确度，极大地提高了算法的查准率。

（4）针对现在主流的分布式图划分算法在进行图划分过程中产生的子图之间的通信频率高导致计算效率不高的问题，本文提出了一种基于分布式图划分的个性化 PageRank 改进算法（Boruvka Adjacent Edge Structure DSP，BAESD）。本文首先提出了一种基于 Boruvka 算法的分布式图划分算法，完成局部子图的构建与平衡化，将大规模网络图划分为多个局部平衡子图，使个性化 PageRank 算法的迭代计算转移到局部子图数据中，其次提出了一种相邻边缘结构（Adjacent Edge Structure，AES），AES 结构主要从两个方面对算法进行优化：一方面是提高图划分效率，尽可能减少图数据处理的时间；另一方面是该结构可以减小局部分区子图与子图之间的依赖关系密度，最后，本文利用 DSP 模型来加速个性化 PageRank 算法的迭代计算，从而进一步提高个性化 PageRank 算法在大规模图上的计算效率。本文算法在四个不同大小数据集上与 PM、RA 和 TDA 等主流算法进行实验对比，实验结果表明本文所提算法极大提高了个性化 PageRank 算法在大规模网络图上的计算效率。

5.2 展望

本文目标是大规模图中实现个性化 PageRank 算法的高效计算以及在排序查询问题上提高算法的查准率，但因为本人能力有限，时间精力也不足，因此本文提出的优化算法仍然存在一些问题，且这些问题希望可以在未来得到进一步的研究。本文未来研究工作安排如下：

（1）针对在个性化 PageRank 算法的排序查询问题研究中，本文提出了一种基于多特征因子的个性化 PageRank 查询优化算法（Improved Enhanced-RatioRank，IER），该算法在一定程度上可以提高个性化搜索查询精确度，但该算法仅仅局限于在静态数据中，如果在动态大规模网络图中和实时数据中，该算法将不能产生良好的效果，因此，如何在动态大规模图中提高个性化 PageRank 算法的查询精确度，是未来的一个研究方向。

（2）在个性化 PageRank 算法的高效计算问题中，本文提出的 BAESD 算法虽然可以明显提高 PPR 的计算效率。由于 DSP 的加速原理是利用计算来换取通信，因此，如果局部计算次数太多的话可能会给节点计算带来很大压力，导致迭代计算负载不均衡，本文考虑设计一种针对分布式计算节点的动态负载平衡机制，

利用该机制来动态的调整分布式计算资源来均衡各个计算节点的计算负担，实现迭代计算的均衡负载，这将会成为本文下一步研究的主要工作。

参考文献

[1] 孙雨生, 雷晓芳. 国内可视化搜索引擎研究进展: 核心内容[J]. 现代情报, 2020, 40(2): 160-167.

[2] Selvan M P, Sekar A C, Dharshini A P. Survey on web page ranking algorithms[J]. International Journal of Computer Applications, 2012, 41(19): 1-7.

[3] S.Brin and L.Page,“The Antonomy of a Large Scale Hypertextual Web Search Engine,”7th Int.WWW Conf. Proceedings,Australia , April 1998: 1-17.

[4] 贾瑞娜. 基于大图的个性化 PageRank 算法研究[D]. 桂林电子科技大学, 2019: 5-33.

[5] Hou G, Chen X, Wang S, et al. Massively parallel algorithms for personalized pagerank[J]. Proceedings of the VLDB Endowment, 2021, 14(9): 1668-1680.

[6] Zhao H, Xu X, Song Y, et al. Ranking users in social networks with motif-based pagerank[J]. IEEE Transactions on Knowledge and Data Engineering, 2019, 33(5): 2179-2192.

[7] Yeh W C, Zhu W, Huang C L, et al. A New BAT and PageRank Algorithm for Propagation Probability in Social Networks[J]. Applied Sciences, 2022, 12(14): 6858.

[8] D. Horowitz and S. D. Kamvar, “The anatomy of a large-scale social search engine,” in Proceedings of the 19th international conference on World wide web, ACM, 2010: 431-440.

[9] Amelkin V, Singh A K. Fighting opinion control in social networks via link recommendation[C]. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 677-685.

[10] Fujiwara Y, Nakatsuji M, Shiokawa H, et al. Efficient ad-hoc search for personalized pagerank[C]. Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data. 2018: 445-456.

[11] Lofgren P, Banerjee S, Goel A, et al. FAST-PPR: Scaling Personalized PageRank Estimation for Large Graphs[J]. In KDD, 2014: 1436-1445.

[12] Wang S, Tang Y, Xiao X, et al. Hubppr: effective indexing for approximate personalized pagerank[J]. Proceedings of the VLDB Endowment, 2016, 10(3): 205-216.

[13] 李兰英, 周秋丽, 孔银. 子图估算 PageRank 网页排序算法研究[J]. 哈尔滨理工大学学报, 2017, 22(2): 117-123.

[14] 刘齐, 黄树成. 基于主题相似度改进的 PageRank 算法研究[J]. 计算机与数字工程, 2022, 50(1): 45-48.

[15] Nykl M, Campr M. Author ranking based on personalized PageRank[J]. Journal of Informetrics, 2015, 9(4): 777-799.

[16] 王冲, 曹姗姗. 基于用户反馈与主题关联度的网页排序算法改进[J]. 计算机应用, 2014，34(12): 3502-3506.

[17] 臧思思, 李秀霞, 孔月. 基于改进 PageRank 算法的作者影响力评价研究[J]. 情报理论与实践, 2019, 42(11): 123-126.

[18] 杨红果, 申德荣, 寇月, 等. 基于强连通分量的个性化的网页排名高效算法[J]. 计算机学报, 2017, 40(3): 584-600.

[19] 陈星玎, 李思雨. 求解 PageRank 的多步幂法修正的广义二级分裂迭代法[J]. 数值计算与计算机应用, 2018, 39(4): 243-252.

[20] Wen C, Hu Q Y, Yin G J, et al. An adaptive Power-GArnoldi algorithm for computing PageRank[J]. Journal of Computational and Applied Mathematics, 2021, 386: 113209.
[21] Huang J, Wu G. Truncated and Sparse Power Methods with Partially Updating for Large and Sparse Higher-Order PageRank Problems[J]. Journal of Scientific Computing, 2023, 95(1): 34.
[22] Huang J, Wu G. Sparse power methods for large-scale higher-order PageRank problems[J]. arXiv preprint arXiv, 2021: 1-25.
[23] Haveliwala T H. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search[J]. IEEE transactions on knowledge and data engineering, 2003, 15(4): 784-796.
[24] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: Bringing order to the web[R]. Stanford InfoLab, 2007: 65-80.
[25] Gonzalez J E, Xin R S, Dave A, et al. {GraphX}: Graph Processing in a Distributed Dataflow Framework[C]. 11th USENIX symposium on operating systems design and implementation (OSDI 14). 2014: 599-613.
[26] Shao B, Wang H, Li Y. Trinity: A distributed graph engine on a memory cloud[C],. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. 2013: 505-516.
[27] Gonzalez J E, Low Y, Gu H, et al. {PowerGraph}: Distributed {Graph-Parallel} Computation on Natural Graphs[C]. 10th USENIX symposium on operating systems design and implementation (OSDI 12). 2012: 17-30.
[28] Bahmani B, Chakrabarti K, Xin D. Fast personalized pagerank on mapreduce[C]. Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. 2011: 973-984.
[29] Pan W, Li Z H, Wu S, et al. Evaluating large graph processing in MapReduce based on message passing[J]. 2011,34(10):1768-1784.
[30] 黄鹏, 郑淇, 梁超. 图像分割方法综述[J]. 武汉大学学报(理学版), 2020, 66(6): 519-531.
[31] Donath W E, Hoffman A J. Lower bounds for the partitioning of graphs[M]. Selected Papers Of Alan J Hoffman: With Commentary. 2003: 437-442.
[32] Simon H D. Partitioning of unstructured problems for parallel processing[J]. Computing systems in engineering, 1991, 2(2-3): 135-148.
[33] Farhat C, Lesoinne M. Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics[J]. International Journal for Numerical Methods in Engineering, 1993, 36(5): 745-764.
[34] Kernighan B W, Lin S. An efficient heuristic procedure for partitioning graphs[J]. The Bell system technical journal, 1970, 49(2): 291-307.
[35] Xie C, Yan L, Li W J, et al. Distributed power-law graph computing: Theoretical and empirical analysis[J]. Advances in neural information processing systems, 2014, 27.1673-1681.
[36] Pacaci A, Özsu M T. Experimental analysis of streaming algorithms for graph partitioning[C]. Proceedings of the 2019 International Conference on Management of Data. 2019: 1375-1392.
[37] Jafari N, Selvitopi O, Aykanat C. Fast shared-memory streaming multilevel graph partitioning[J]. Journal of Parallel and Distributed Computing, 2021, 147: 140-151.
[38] Awadelkarim A, Ugander J. Prioritized restreaming algorithms for balanced graph partitioning[C]. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 1877-1887.

[39] Abbas Z, Kalavri V, Carbone P, et al. Streaming graph partitioning: an experimental study[J]. Proceedings of the VLDB Endowment, 2018, 11(11): 1590-1603.

[40] Petroni F, Querzoni L, Daudjee K, et al. Hdrf: Stream-based partitioning for power-law graphs[C]. Proceedings of the 24th ACM international on conference on information and knowledge management. 2015: 243-252.

[41] Sanders P, Schulz C. Engineering multilevel graph partitioning algorithms[C]. European Symposium on Algorithms. Springer, Berlin, Heidelberg, 2011: 469-480.

[42] Rahimian F, Payberah A H, Girdzijauskas S, et al. Ja-be-ja: A distributed algorithm for balanced graph partitioning[C]. 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems. IEEE, 2013: 51-60.

[43] Hashem I A T, Anuar N B, Marjani M, et al. MapReduce scheduling algorithms: a review[J]. The Journal of Supercomputing, 2020, 76: 4915-4945.

[44] Maleki N, Rahmani A M, Conti M. MapReduce: an infrastructure review and research insights[J]. The Journal of Supercomputing, 2019, 75: 6934-7002.

[45] Fujiwara Y, Nakatsuji M, Onizuka M, et al. Fast and exact top-k search for random walk with restart[J]. Proceedings of the VLDB Endowment, 2012, 5(5): 442-453.

[46] Yuan A. Two Problems Involving Random Walks on Graphs: Random surfers, PageRank, and short-time asymptotics for the heat kernel[D]. University of Minnesota, 2021: 356-367.

[47] Pashikanti R P, Kundu S. FPPR: fast pessimistic (dynamic) PageRank to update PageRank in evolving directed graphs on network changes[J]. Social Network Analysis and Mining, 2022, 12(1): 141.

[48] 蒲冰远. 马尔科夫链与网页排序问题的数值算法研究[D]. 四川: 电子科技大学, 2015:33-56.

[49] 沈照力. 马尔科夫链的多重网格算法及网页排序问题的求解[D]. 四川: 电子科技大学, 2018: 22-99.

[50] Huang B, Vaidya U. A Convex Approach to Data-Driven Optimal Control via Perron–Frobenius and Koopman Operators[J]. IEEE Transactions on Automatic Control, 2022, 67(9): 4778-4785.

[51] Lawrence Page, Sergey Brin, Rajeev Motwani, et al. The Page Rank citation ranking:Bringing order to the web[R]. State of California, Stanford University, 1999:564-568.

[52] Tsuchida K, Matsumoto N, Shin A, et al. Cache-Efficient Approach for Index-Free Personalized PageRank[J]. IEEE Access, 2023: 6944-6957.

[53] Singhand R，Sharma D K.RatioRank：Enhancing the impact of inlinks and outlinks[J]. Advance Computing Conference（IACC）, 2018, 7903(21): 794-799.

[54] 林盛, 肖旭. 基于 RFM 的电信客户市场细分方法[J]. 哈尔滨工业大学学报, 2006, 38(5): 758-760.

[55] 郭庆宝, 贾代平. 融合反馈信息与内容相关度的 PageRank 改进算法[J]. 计算机工程与设计, 2011, 32（12）: 4071-4074.

[56] Qiao W ,Créput J . GPU implementation of Borůvka’s algorithm to Euclidean minimum spanning tree based on Elias method[J]. Applied Soft Computing Journal, 2019, 76.

[57] Khayyat Z, Awara K, Alonazi A, et al. Mizan: A system for dynamic load balancing in large-scale graph processing[A]. Proceedings of the 8th ACM European Conference on Computer Systems-EuroSys 13[C]. New York, USA: ACM Press, 2013: 169-182.

[58] 柳菁, 李琪. DisHAP: 基于层次亲和聚类的分布式大图划分算法[J]. 电子学报, 2021, 49(10): 2002-2011.

[59] Zhang WD, Cui C. Delta-stepping synchronous parallel model. Ruan Jian Xue Bao/Journal of Software, 2019, 30(12): 3622−3636.

[60] Custódio A L, Rocha H, Vicente L N. Incorporating minimum Frobenius norm models in direct search[J]. Computational Optimization and Applications, 2010: 265-278.