

**硕 士 学 位 论 文**

MASTER’S DISSERTATION

论文题目 大规模无标度网络生成算法研究

**作者姓名** 王倩倩

**学位类别** 工程硕士

**指导教师** 陈子阳 教授

**2018年5月**

中图分类号：TP312 学校代码：10216

UDC：621.3 密级：公开

**工学硕士学位论文**

大规模无标度网络生成算法研究

|  |  |  |
| --- | --- | --- |
| 硕士研究生 | ： | 王倩倩 |
| 导师 | ： | 陈子阳 教授 |
| 申请学位 | ： | 工学硕士 |
| 工程领域 | ： | 软件工程 |
| 所 在 单 位 | ： | 信息科学与工程学院 |
| 答 辩 日 期 | ： | 2018年5月 |
| 授予学位单位 | ： | 燕山大学 |

A Dissertation in Computer Science and Technology

THE RESEARCH On GENERATION OF SCALE-FREE NETWORKS ALGORITHM

by Wang Qianqian

Supervisor：Professor Chen Ziyang

**Yanshan University**

May, 2018

燕山大学硕士学位论文原创性声明

本人郑重声明：此处所提交的硕士学位论文《大规模无标度网络生成算法研究》，是本人在导师指导下，在燕山大学攻读硕士学位期间独立进行研究工作所取得的成果。论文中除已注明部分外不包含他人已发表或撰写过的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。本声明的法律结果将完全由本人承担。

作者签字： 日期： 年 月 日

燕山大学硕士学位论文使用授权书

《大规模无标度网络生成算法研究》系本人在燕山大学攻读硕士学位期间在导师指导下完成的硕士学位论文。本论文的研究成果归燕山大学所有，本论文的研究内容不得以其它单位的名义发表。本人完全了解燕山大学关于保存、使用学位论文的规定，同意学校保留并向有关部门送交论文的复印件和电子版本，允许论文被查阅和借阅。本人授权燕山大学，可以采用影印、缩印或其它复制手段保存论文，可以公布论文的全部或部分内容。

保密□，在 年解密后适用本授权书。

本学位论文属于

不保密□。

（请在以上相应方框内打“√”）

作者签名： 日期： 年 月 日

导师签名： 日期： 年 月 日

# 摘 要

无标度网络是指顶点的度服从泊松分布的拓扑图。随着现代信息化的大量普及，大规模数大据图越来越多地被应用到社交网络等现实场景，而现实世界的大部分数据图结构均满足大规模无标度网络的特性，因而可以通过对大规模无标度网络图数据的处理来验证图处理算法的性能。然而，现有大规模无标度网络图的构建方法存在效率低，扩展性差等问题。本文旨在研究大规模无标度网络图的高效生成方法，具体内容如下。

首先，针对现有方法生成无标度网络所存在的低效性问题，提出一种基于BA模型的大规模无标度网络生成算法及相应的网络图数据存储结构RWBT（roulette wheel binary tree）。设计了相应的遍历策略以便提高算法效率，并对节点的删除操作优化处理，通过采用节点序号互换代替删除桶的操作从而降低更新树的频次，解决了在内存中处理大规模数据图时，重复遍历节点导致的低效性的问题。

其次，提出一种基于RWBT结构扩展的RWKT（roulette wheel k-tree）k叉树索引结构，并进一步改善数据图的存储结构以及相应的节点遍历策略，通过对k叉树结构的内部节点进行排序，减少了冗余比较次数，从而达到进一步优化算法的处理效率的目的。

最后，通过对本文讨论到的几种算法的不同参数进行设置，通过实验分别从结果数量、运行时间以及原子操作次数等多方面进行深入比较和分析，实验结果进一步验证了算法的高效性和可扩展性。

关键词：无标度网络；索引结构；图处理技术；

# Abstract

Scale free net work is a typical muti-objective optimization problem, aiming at identifying the information that users may interested in a multi-dimensional dataset. Skyline query is widely applied in many fields, such as multi-criteria decision making system, navigation system, information recommendation system, data mining, etc. Group-Skyline is the further generalization of the original skyline. Group-Skyline needs to consider not only individual point but also groups consisting of points, aiming at finding the set of groups that are not dominated by any other group in dataset. As the dataset, dimension, and the size of the requested group increases, the result returned by the existing algorithm is extremely large. However, in the practical application, users only care about a small number of representative results. In this paper, the problem of top-k Group-Skyline query is researched. The specific research contents are as follows.

Firstly, through the analysis of the existing query algorithms, we find that the problem of existing methods have a large result and inefficient. We use the characteristic which the points in the low skyline layers dominate the points in the high skyline layers and propose a ranking strategy based on the skyline layer for the result, which is according to the skyline layer and the number of vertices on them. We propose the corresponding SLGS algorithm, used to return the top-k results.

Secondly, for the problem that the SLGS may return the same ranking result, we propose a ranking strategy of Group-Skyline result based on the skyline layer and the vertex coverage and the relative importance of the result of the same ranking is further distinguished by the size of the vertex overlay set. p-k results. In order to improve the efficiency of the algorithm, thus reducing the cost of the enumeration and improving the efficiency of the algorithm.

At last, based on real data set, the result of the experiment shows the effectiveness o

**Keywords:**

目 录

[摘 要 I](#_Toc509768545)

[Abstract II](#_Toc509768546)

[第1章 绪 论 1](#_Toc509768547)

[1.1 研究背景 1](#_Toc509768548)

[1.2 研究现状 2](#_Toc509768549)

[1.3 研究内容 3](#_Toc509768550)

[1.4 本文结构 4](#_Toc509768551)

[第2章 基础知识概述 5](#_Toc509768552)

[2.1 无标度网络的相关概念 5](#_Toc509768553)

[2.1.1 网络的基本概念 5](#_Toc509768554)

[2.1.2 创建网络的基本模型 8](#_Toc509768555)

[2.2 创建无标度网络的基本算法 9](#_Toc509768556)

[2.2.1 简单轮盘算法 9](#_Toc509768557)

[2.2.2 随机选择算法 12](#_Toc509768558)

[2.2.3 基于桶结构的轮盘算法 12](#_Toc509768559)

[2.3 本章小结 14](#_Toc509768560)

[第3章 基于RWBT的网络生成算法 15](#_Toc509768561)

[3.1 问题分析 15](#_Toc509768562)

[3.2 基本思想 16](#_Toc509768563)

[3.4 算法描述 17](#_Toc509768564)

[3.4.1 初始化无标度网络 17](#_Toc509768565)

[3.4.2 取样操作 18](#_Toc509768566)

[3.4.4 更新RWBT权值 20](#_Toc509768567)

[3.4.3 插入节点 21](#_Toc509768568)

[3.4.3 删除节点 23](#_Toc509768569)

[3.4.3 RWBT算法 25](#_Toc509768570)

[3.5 算法分析 26](#_Toc509768571)

[3.5 算法优化 26](#_Toc509768572)

[3.6 本章小结 27](#_Toc509768573)

[第4章 基于RWTT生成算法 28](#_Toc509768574)

[4.1 问题分析 28](#_Toc509768575)

[4.2 算法思想 29](#_Toc509768576)

[4.3 算法描述 29](#_Toc509768577)

[4.4 算法分析 31](#_Toc509768578)

[4.5 算法优化 31](#_Toc509768579)

[4.5.1 基于RWMST生成算法 32](#_Toc509768580)

[4.6 本章小结 33](#_Toc509768581)

[第5章 实验结果与分析 34](#_Toc509768582)

[5.1 实验环境 34](#_Toc509768583)

[5.2 数据集及评价标准 34](#_Toc509768584)

[5.3 性能比较与分析 35](#_Toc509768585)

[5.3.1 新增边数的影响 36](#_Toc509768586)

[5.3.3 初始节点个数的影响 36](#_Toc509768587)

[5.3.3 数据规模的影响 37](#_Toc509768588)

[5.3.5 内存增长情况比较 37](#_Toc509768589)

[5.4 本章小结 37](#_Toc509768590)

[结 论 38](#_Toc509768591)

[参考文献 39](#_Toc509768592)

[攻读硕士学位期间承担的科研任务与主要成果 43](#_Toc509768593)

[致 谢 44](#_Toc509768594)

# 第1章 绪 论

## 1.1 研究背景

现实中诸多领域的网络都带有无尺度的特性，例如[因特网](https://zh.wikipedia.org/wiki/%E4%BA%92%E8%81%94%E7%BD%91)、语义网、金融系统网络、引文网络、[生物信息网络](https://zh.wikipedia.org/wiki/%E7%A4%BE%E4%BC%9A%E6%80%A7%E7%BD%91%E7%BB%9C)等等。而图是最能有效描述这种复杂网络数据的常用数据结构，幂律图生成方法研究也应运而生，而其相应的创建技术是复杂网络领域研究的热点问题之一。

现有的描述现实世界图结构的算法大部分情况下不能满足特殊研究的需求，虽然大部分复杂网络具有相似的统计特征以及结构特性，更多情形下一些特殊群体仍然具有一些独特的性质。比如社交网络以及科学协作网络具有模块化的社交属性[39]，从发性和自相似的权值增加性[16,38]，幂律性[38]等等，而当前面临的这些挑战需要研究创建一种遵循现实世界特性的复杂网络。

目前基于优先连接本身固有的低复杂度的Barabási-Albert ( (BA)模型是应用范围较广的一种。在BA模型中，网络图是通过一种名为优先连接的迭代的启发式过程生成的。而关于复杂网络的研究仍然是一个具有重大意义但是不太成熟的领域。这个领域的第一个贡献要追溯到图理论开始发展的1900年初，网络模型的应用对复杂网络的结构与特性的理解作出了卓越的贡献。

区分复杂网络与其他类型网络(比如，随机网络、小世界网络)的最主要的特征是其固有的复杂特性，例如，复杂网络具有随时间动态生成的不规则特点[7]，很多大型网络结构都具有的普遍特征是节点的连接遵循幂律分布，即，一个节点的连接到其他k个节点的概率P(k)遵循P(k)~k-λ的幂律分布，其中2<λ<3[6]。大量的理论模型提出来解释无标度网络是在如何随时间增长的，其中包括，Barabási-Albert模型[6]，适应度模型[8,12]，修补模型[44]，最优化模型[24,47]，Chung-Lu[1]，BTER[30]，R-MAT[16]，双曲圆盘模型[49]，以及高度优化的容差模型[15]。

图增长模型不仅仅对研究自然世界中的复杂拓扑图提供了理论抽象，而且可以被应用到对生成合成图更长远的分析模拟中。近二十年来，针对不同的增长模型，衍生出了多种随机化的算法用来构造合成图。目前在生成合成图的最主要的动机是现实世界的网络对研究来说很难归结于隐私保护。另外，合成图生成器经常被应用到创建有相似统计特性的独立图中。这些复杂的独立图对于统计测试具有重要意义，比如，性能测试和交叉验证[49]。目前，由于小型图结构不能展现在大规模图结构中的特性，对于网络的研究正在向巨型网络迁移[4,32]。

网络理论中，虽然现有的随机网络能够较为详细地描述网络图的拓扑结构，但是不能完全呈现现实世界网络的所有复杂关系。比如协同性，动态变化的性质。有些网络可以通过少量的随机捷径改变原有的拓扑结构，从而出现小世界效应，无标度网络出现的主要用途是为了能够准确描述现实世界复杂的网络结构，并揭示增长性和择优机制在复杂网络演化过程中的普遍性以及幂律的重要性，且在当前多个研究领域中复杂网络属于热门研究。为了能够顺应复杂网络创建技术的发展，越来越多的行业开始根据现实世界网络的结构特征以及数据存储方式来设计与本领域相关的语义，以便能够实现与无标度网络生成技术的接轨。

在现有研究成果的基础上，本文结合所研究无标度网络的结构特征、节点处理效率的情况，以及进行高效的节点处理研究。

## 1.2 研究现状

复杂网络的理论和应用已经有了相对成熟的发展，在学术界以及社会层面都引起了很轰动的效果，同时对于这些理论以及应用的研究也留下了广阔的发展空间。对于大多数研究机构来说，单一的图模型生成器无法满足现阶段的研究需求，构造大型拓扑结构图是具有重大意义的。因而，网络结构拓扑图的构建算法有很多并行分布式的实现，这些算法都很注重理论上以及实际上对于图模型的高效并行化[4,36,37,40]。

从存在的复杂网络研究内容显示，在现实网络节点分布均匀的情形下，现有的研究成果能够快速且简单的实现网络构建，但是，最新研究结果发现，在节点分布多变，社区结构属性构成复杂的情况下，现有研究成果较难满足需求，本篇论文中着重研究了影响力最大的无标度网络生成器[5,13,17,46,55] Barabási-Albert (BA)模型.如图1-1所示的BA网络即为本篇论文实验的例子。BA模型基于优先连接机制生成一种随机的无标度图，其中，优先连接机制为：图中的节点逐个生成，并每次只连接到一个或几个图中已存在的节点。这种连接机制按照“rich-get-richer”的方式，即，一个节点的连接越多，这个节点被新节点连接的可能性越大。简单来说，对于新加入网络的节点与一个现有网络中的节点相连接的概率与这个被连接的节点的度成正比。虽然BA模型在不同的领域已经被广泛应用[5,17,46,55]，但是几乎没有实验研究注意到这个模型的效率。虽然很多图生成模型是平方级的时间复杂度，比如BA模型的时间复杂度是O(|V|2),这导致这个模型不适用于非常大的网络图。近期这个算法已经在大量图分析软件包中被实现了，比如，iGraph1和graph-tool2，而且在很多图规模很小(节点数小于107，|V|<107)的研究工作中也有了广泛的应用。但是，对于在需要生成较大规模的合成网络图的情形下，BA模型的O(|V2|)的时间复杂度使得现有算法的实现有一定的局限性。

近年来，在很多场景中幂律图生成算法都有广泛的应用，比如，仿生学研究，取样调查，数据汇总[13,33]。关于无标度网络[1,16,21,24,30,40]生成模型的发展的研究层出不穷。类似几何分布或者经验脸谱分布这样的基于取样机制的单一图生成器有确定的概率分布，这使得能够很容易地被转换为并行版本，因为选择复杂样本的过程是独立的，这样就可以并行运行程序。比如，LDBC机构使用的生成器S3G2和DataGen是固有并行的，因为对每个用户生成其在社会中的度是相互独立的。许多现实世界中的网络系统，从细胞组织到因特网，都会收敛到节点的度分布为幂律分布的结构上，而与这些网络的组织方式无关。无标度网络的重要意义在于将网络的结构与演化的不可分割的特性充分解释，即，现实世界中的网络是处于动态变化的。

但是几乎没有对并行优先连接机制的研究。并行化的BA算法(PBA)是一种忽略了其本身序列性的近似算法,PBA算法是被设计用来减少在大规模集群中的沟通代价，图的节点通过两个步骤在随机选择与处理单元之间确定：1)处理单元之间的选择过程：根据节点的权值随机选择一个处理单元；2)处理单元内部的选择过程：在分配到被选择的处理单元中的节点间选择。阿拉姆等人[4]表示第一种基于内存分布的生成随机图的并行算法会严格遵循优先连接机制的增长方式。但是，这个算法仅仅在多于100个进程的情况下比基本算法更高效。Lo等人基于近似算法[36]针对优先连接机制也提出一种分布式版本的解决方案。

Lo等人的研究表明近似算法和并行化的不安全(无锁)计算技术的效果应当认真考虑BA模型的并行化版本，像众多并行配置在网络中的结果一样不用依赖原始模型的一些特性，诸如，节点的度分布。因此，相比于近似算法，当前的研究更关注于研究精确的数据结构来提高在动态轮盘中的选择效率。

本文研究的目的是提出一种在物理机器上能够远远超过现有方法的序列算法。这个算法既能当作串行算法，也能当作类似PBA这样的两步并行一个运行模块，换而言之，在任何并行的BA模型中，无论进程内部还是进程间的操作都需要快速的动态轮盘，因此ROLL可以充分提高PBA以及其它分布式的优先连接算法[36,53]。

## 1.3 研究内容

根据上述分析，在大多数现实场景中，。本文针对现有图生成算法存在的问题进行研究，首先针对算法中很大一部分的计算是根据节点度值选择需要连接的节点，又称作轮盘选择的过程，比如，查找与给定随机数相关联的节点的操作。在网络图存储结构中有很大一部分数据的计算是非必要的。在此主要研究内容是通过改进存储结构提高创建无标度网络的效率。

首先，分析现存算法中存在的问题，发现目前的存储结构在遍历节点过程中存在冗余计算的问题。结合实际网络结构针对在动态轮盘中能够快速随机选择节点，对该问题进行了理论分析及探究，提出了一种先进的存储结构RWBT（roulette wheel binary tree）。

其次，基于降低树结构中由根节点到叶子节点的最短编码长度的基本思想，通过将RWBT分支进行扩展，提出了RWTT（roulette wheel three tree）。该算法通过进一步降低节点存储的平均编码长度减少了冗余比较的操作，减少了重复计算，从而提高了算法的处理效率。

再次，在对RWTT结构扩展的过程中发现当分支扩展过多之后，随着树的孩子节点增多，由根节点向下遍历过程对内部节点的比较次数增多，降低了算法效率，又提出了在分支较多情况下对树的内部节点进行排序的RWMST（roulette wheel muti sorted tree）算法。

最后，通过基于多个数据集对本文算法进行的大量试验，从网络构建时间，节点处理效率两方面对不同方法的性能进行了比较，验证了所提出的所有算法的有效性。

## 1.4 本文结构

本文设计的内容为在内存中快速高效的构建无标度网络构建技术，重点内容为构建RWBT结构、扩展的RWMST结构，本文总体上可分5章，以下为从第2章起的结构。

第2章为相关基础知识概述，主要介绍了无标度网络的特征以及涉及到的网络生成模型，同时也介绍了复杂网络以及幂律分布的基本概念。

第3章对已有方法存在的问题进行改进优化，提出了RWBT存储结构，且将基本的RWBT结构进行扩展RWTT，并在此基础上分析相应的节点分布特征以及各个结构对节点处理效率的影响。最后从理论和示例中对该算法进行了论证。

第4章根据创建多叉树的策略，提出对RWTT的改进算法，改进遍历策略，对内部节点进行排序，提出RWMST结构，分别从理论和示例中对本章的算法进行了论证。

第5章为实验，通过大量数据集对所提算法进行实验测试，并对实验结果进行详细分析，分别对处理节点的效率进行了对比，验证了提出方法的高效性及可行性。

最后是结论，总结了研究生期间的研究成果，分析论文的主要研究内容，阐述算法的核心思想，并提出其以后有待改进的地方以及对未来研究工作方向的瞻望。

# 第2章 基础知识概述

## 2.1 无标度网络的相关概念

定义2.1 复杂网络 复杂网络由许多节点以及节点之间的边构成，比如常见的交通网络、计算机网络、电力网络以及社交网络，复杂网络既是数据的一种表现形式，也同样是一种学术研究手段。对于复杂网络，是一类具有自组织、自相似、吸引子、小世界、无标度中全部或部分性质的网络。即，复杂网络具有高度复杂性，其高度复杂性体现在以下几个方面：

1. 小世界特性

小世界特性，具有较小的网络平均路径长度但是有较大的网络聚集系数的网络称为小世界网络，网络的规模大小和稀疏稠密性质可由网络平均路径长度及网络聚集系数来度量的。

平均路径长度：对于网络图中的任意两个节点，使得这两个节点可以连通的最少边数即为这两个节点之间的路径长度。网络中所有节点对的路径长度的平均值为该网络图的平均路径长度。

网络聚集系数：对于普通的无向图网络，网络的聚集系数用来描述网络中节点之间连接的紧密程度，表示了一个节点与其邻居节点之间的互相连接的程度。网络中第i个节的聚集系数C(i)为：

其中，k(i)是第i个节点的邻居节点的个数，也称为该节点的度。e(i)是d个邻居节点之间的实际连接边数。d个邻居节点之间的最大连接边数是

1. 无标度特性

无标度特性是指复杂网络中的节点的度分布服从幂律分布的一类特性，具体表现为如下两个方面：1)增长性：有新节点加入；2)优先连接：新加入网络中的节点与现存网络中的节点连接的概率与已存在的节点的属性有关。

定义2.1 无标度网络 无标度网络是网络中的节点符合无标度特性的一类复杂网络。现实世界的大部分网络都是少数的节点往往有大量的连接，而大部分节点的连接很少。网络中与其他节点有k个连接的节点出现的频率P(k)随k值的变化满足P(k)~k-λ，其中λ的范围为2<λ<3。图2-1阐释了P(k)随λ变化而变化的趋势。

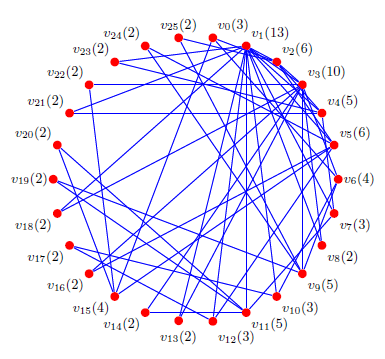
图2-1

无标度网络模型揭示了增长性与优先连接机制在复杂网络中的广泛性以及幂律分布对研究无标度网络复杂性的重要性，图2-1即为一个基于BA模型构建的无标度网络。

定义2.1 幂律分布 形如y=cx-λ的函数,其中x，y为正的随机变量，c，λ为大于零的常数这样的简单幂函数，称之为幂律分布，这种分布的性质是大部分事件的规模很小，而少数事件的规模很大，例如二八定律。将如上公式等号两边同时取对数，可得lny和lnx之间的线性关系为lny=lnc-λlnx，由此线性关系可知在双对数坐标下，幂律分布表现为一条斜率为幂指数为负数的一条直线，这一线性关系也是判断给定的随机变量是否满足幂律特性的重要依据。在确定给定的两个随机变量满足如上所示的线性关系之后，可以利用一元线性回归方程和最小二乘法，可得lny对lnx的经验回归方程，从而得到y与x之间的幂律关系式，进而求解两个随机变量之间的相关系数，得到所给定的随机变量之间的幂律性质。

除了图1-1所示网络呈现的长尾分布，现实网络中的节点还存在其他形式的幂律分布广泛应用在物理学、生态学、计算机科学以及社会科学等众多领域，比如规模-概率分布、名次-规模分布等。

图2-1



定义2.1优先连接 优先连接即为新加入网络的节点与网络中现有节点连接的概率与现有节点的固有概率相关联。基于优先连接机制生成无标度网络是基于节点的度有固定概率分布的取样过程，也表明了这种机制是现实世界中的无标度网络形成的必要因素。

定义2.1 BA模型 Barabasi和其同伴Albert经过详细对比研究部分万维网的拓扑结构发现一些称作hub点的节点比网络中的其他节点有更多的连接，而且整个网络互相连通的节点之间的连接服从幂律分布。通过网络拓展新的节点逐个加入，新节点会优先连接到现存的节点中连接概率较大的节点，基于这两种性质创建复杂网络的模型称为BA模型。

基于BA模型生成无标度网络大致分为两步，首先，从一小部分(m0)节点开始，然后，将其余节点依次加入现有网络，且规定网络中新加入的节点被网络中第i个节点连接的概率*p*(*ki*)与网络中该节点的度成正比，其中*p*(*ki*)满足公式（2-1），而且每次只在网络中加入一个新节点。

 (2-1)

BA模型的这种特性再现了观测到的平稳的无标度分布，揭示了大规模网络的发展受到自组织现象的支配。除了幂律分布的特性，基于BA模型生成的网络图结构还具有一些现实世界的特性，比如图中的平均路径长度随网络规模的增长成对数增长，这种特性使得BA模型比一般的随机图[24]具有更短的平均路径长度。另外，BA模型可以更灵活的被应用到构建度的分布指数在(2,)之间的任意网络图[23]。

## 2.2 现有图生成模型

具体来说，现有复杂网络具有统计学以及结构上的共性，比如幂律分布的性质。目前研究成果中已经存在诸多模型来创建具有上述性质的复杂网络图，本小节将针对几种常见的复杂网络创建模型进行简要的概述。

Block Two-Level Erd˝os-Rényi（BTER）模型[11]描述了综合网络中的复杂拓扑关系，该种模型可以用来获取两个不同网络之间的性质差别以及度分布差异。Chung-Lu模型适用于创建一类可以根据已有的度分布性质预期到度分布的复杂网络[1,18]，是一种特殊的BTER模型，基于Chung-Lu模型创建的网络图中的每条边的概率与其两端顶点的概率成正比。 Recursive MATrix method(R-MAT)模型是一种迭代式的图生成模型，常用于描述图的邻接矩阵存储形式，但是该种模型对于表达不同规模的网络图涉及到的现实特性有明显的缺陷。 hyperbolic unit-disk模型使用了一种类似两级四叉树的空间数据结构，可以快速地生成一种随机双曲图[49]。

文献[27]中详细阐释了各个模型的，本小节只对比了上文提及的几种现有的网络图生成模型的时间复杂度并进行了汇总，如表2-1所示。通过总结比较，更直观地显示了BA模型在创建无标度网络方面的优势。

表2-1 图生成模型时间复杂度

|  |  |
| --- | --- |
| 算法名称 | 最优时间复杂度 |
| Watts-Strogataz[51] | O(|E|2) |
| Barabasi-Albert[24] | O(|V|2) |
| Chung-Lu[1] | O(|V|+|E|) |
| R-MAT[16] | O(|E|log|V|) |
| BTER[30] | O(|V|+|E|\*dmax) dmax表示网络中节点的最大度 |
| Hyperbolic unit disk[49] | O(|V|3/2+|E|)logn |

## 2.3 创建无标度网络的基本算法

表2-1 本文所用符号及其意义

|  |  |
| --- | --- |
| 符号 | 意义 |
| |*V*| = *n* | 节点的个数 |
| |*E*| | 边的个数 |
| |*B*| | 包含唯一度节点的桶的个数 |
| *m0* | 初始节点的个数 |
| *m* | 每个节点的边数 |
| *LCW* | 平均编码长度 |
| *deg*(*vi*) | 节点*vi*的度 |
| *Sn* | 总的度数 = *deg*(*vi*)的和 |

平均编码长度：从树的根节点开始遍历到叶子节点的平均距离。

### 2.3.1 简单轮盘算法

Simple-RW是简单轮盘算法

基于BA模型串行创建有n个节点的无标度幂律图，需要以一个包含m0个节点，0个边的空网络开始。对于剩下的n-m0个节点，每个节点需要连接到现有网络中的m个不同的节点。网络中第i个节点vi被新节点连接的概率与这个节点的度deg(vi)成正比。

比如给定一个包含k个节点的集合v1,v2,…,vk。当产生了一个新节点vk+1，这个节点需要与集合v1,v2,…,vk中m个不同的节点连接m条边，对于这个新节点的每一条边，连接到现有网络中的节点的概率为与该节点的度成正比。

 (2-1)

每个新节点需要连接的m个节点应该是不被替代的取样，也就是说，当一个节点被选择与新节点连接，这个被连接的节点在网络中的概率就会变成0.这种基于舍选抽样的处理机制：如果被选中的节点已经与新加入的节点进行连接，第二次进行取样的过程中会把这个节点进行过滤。但是在大多数情况下，例如BA模型中m远远小于n，这使得拒绝概率非常低，也就是说这种情况几乎不会发生。拒绝概率是指一个已经被选择进行连接的节点被再次选择的概率。

上文所述的优先连接机制应用了一种简单轮盘算法来选择目标节点。其中，简单轮盘算法随机选择一个节点的主要思想如下：一个轮盘即为一系列具有与其相应的选择概率的节点。随机选择一个节点时，首先需要产生一个0到1之间的随机数，然后选择的节点vk的累计概率满足P(vk-1)<x≤P(vk)进行连接操作。其中P(vk)表示该节点的累计概率密度，即，。

在图1-1中，通过BA模型创建的含有26个节点的无标度网络，其中，m=m0=2,该图中节点的度总数Sn为

当继续增长该图的扩充，新加入的节点v26需要在{v1,…,v25}中选择两个节点连接，表2-1表示了与图1-1中的节点的度以及相应的概率的轮盘结构，在该轮盘结构中选取两个节点与新节点v26连接的过程需要进行两次随机数构造，对于每次的随机选取，都需要生成一个随机数。假设产生的随机数分别是0.30和0.72，跟据表3所示的轮盘结构，第一个目标节点是v3(0.23<0.30<0.33=P(v3))，第二个节点是v13(0.71<0.72<0.73=P(v13))。

在上述例子中，必须从P(v1)一直比较到P(v13)才能找到与产生的随机数0.72相关联的节点。当网络中需要添加一个新的节点时，与这个网络对应的轮盘也要相应的增加节点。所以，无论何时在网络中加入一条边，这条边两端的节点的概率都会随其度的增加而相应的增加，而且为了保证整体的概率为1，这个轮盘中的所有节点的概率都需要进行更新。这样，在每次的选择之后轮盘都需要进行调整，累计概率P也需要在每次循环后重新计算。由于等式2-1中的分母是常量，可以去除该分母避免这种重复计算。这样，轮盘中的节点的概率p(v1),…,p(vn)可以转换为权值w1,…,wn，其中，wi=deg(vi)=Sn\*p(vi)。同理可知，可以通过计算部分和σ1,…, σn代替计算累计概率，其中，在轮盘取样的过程就简化为，产生一个随机数r∈[1,Sn],然后选择一个节点vk，使得vk满足σk-1<r≤σk。

使用节点的权值w代替其度值的优势在于一些部分和不需要做改变也不需要重复计算。比如，在上述的例子中由于在v3与v13的度增加以后σ0到σ2的值没有改变，则不需要重复计算从σ0到σ2的值。另外一点，使用节点的权值也就是节点的度来替代节点的概率更高效的原因是节点的权值是按整数存储的。按照前文所述的基于优先连接机制的简单轮盘算法，每当一个新加入的节点vn连接到现有网络中的一个节点vk，σk,…,σn-1的累积和都会增加1，σn的累积和会增加2。综上所述，对于每一个连接的更新轮盘操作的时间复杂度是O( n= |V|)。而随机树r的生成只需要线性时间，时间复杂度为O(1)。因此，生成整个包含|E|条边的无标度网络需要的时间是O(|E|(|V| + 1)) = O(mn2)。

表2-1 图1-1所示无标度网络的轮盘结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| vi | wi=deg(vi) | p(vi) | P(vi) | σi(vi) |
| v0 | 3 | 0.03 | 0.03 | 3 |
| v1 | 13 | 0.14 | 0.17 | 16 |
| v2 | 6 | 0.06 | 0.23 | 22 |
| v3 | 10 | 0.10 | 0.33 | 32 |
| … | … | … | … | … |
| v12 | 3 | 0.03 | 0.71 | 68 |
| v13 | 2 | 0.02 | 0.73 | 70 |
| … | … | … | … | … |
| v25 | 2 | 0.02 | 1.00 | 96 |

### 2.3.2 随机选择算法

目前有科学家提出了一种基于随机接受节点的轮盘选择算法[35]来替代遍历整个无标度网络结构。在这个算法中，选择目标节点的过程被分为两步：

1)在无标度网络中等概率随机选择一个节点vi。

2)如果该节点vi在网络中的概率满足pacc(vi) = wi/wmax，则选择该节点vi与新加入网络的节点连接，如果不满足以上等式则跳向第一步继续随机选择一个节点。其中，wmax是该无标度网络对应的轮盘结构中的最大度数。

大致来说该算法在轮盘中选择一个节点的时间复杂度是O()，其中， 是轮盘中所有节点的权值的平均值。在基于BA模型创建的无标度网络中，理想情况下的权值(即，节点的平均度)为。同理，对于服从幂律分布的网络图来说节点的最大度 的理想期望值为 ，其中，，这样， 。综上所述，基于随机选择机制在BA模型生成的无标度网络中选择单个节点的算法的时间复杂度是O()。因而构造整个网络图需要对这个轮盘进行 次操作，所以该算法的整体时间复杂度是O()。

虽然该算法减小了待比较节点的规模，并且时间复杂度略优于简单轮盘算法SimpleRW，但是，这种基于随机机制的节点选择算法对于生成无标度网络来说仍然是低效并且过度依赖于网络图中的节点被接受的平均概率，即，依赖于要选择一个待连接的节点时预估的图中节点的最大度数以及需要进行的平均操作次数。这种特性反过来也取决于对网络轮盘中的节点的度的最大值与平均概率的估算，简而言之，无标度网络中节点的度分布是服从幂律分布的，这不是一种均匀分布，对于大部分节点来说，wi远远小于wmax，因此，每个节点被接受的概率pacc(vi)大多数情况下是很小的，由于遍历或者修改一大部分轮盘中的节点是不必要的，这样会导致在找到最终节点之前有大量不必要的重复取样的操作，其中取样操作对于规模较大(|V| > 106)的图来说是非常耗时的。

### 2.3.3 基于桶结构的轮盘算法

RW-bucket适用于轮盘中节点的数量过大，并且这些节点被选中的概率是具有最大公约数的有理数，大量节点有相同的选中概率的情况。比如，无标度网络的度分布服从幂律分布，并且网络中节点的度为整数，理论上来说，具有k条边的节点被连接的期望值为p(k) ~ k-λ，因而，大量的节点，尤其是度较低的节点更有可能拥有相同的度，进而拥有相同的选中概率。

直观的说，RW-bucket是通过将具有相同度的节点按度分组到桶，其中，桶就是节点的子集，即，Bd={vi|wi=d}。因此，只需保存不包含空桶的邻接表B。这样，图一所示的无标度网络中的26个节点被分为如表2-2所示的7个桶，表2-2中p(Bd)表示Bd的概率密度函数PDF，P(Bd)表示Bd的累计分布函数。

表2-2 图一所示网络的桶结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bucket | Degree d | #Nodes | p(Bd) | P(Bd) | Nodes |
| B2 | 2 | 13 | 26/96 | 26/96 | {v8,v13,v14,v16,…,v25} |
| B3 | 3 | 4 | 12/96 | 38/96 | {v0,v7,v10,v12} |
| B4 | 4 | 2 | 8/96 | 46/96 | {v6,v15} |
| B5 | 5 | 3 | 15/96 | 61/96 | {v4,v9,v11} |
| B6 | 6 | 2 | 12/96 | 73/96 | {v2,v5} |
| B10 | 10 | 1 | 10/96 | 83/96 | {v3} |
| B13 | 13 | 1 | 13/96 | 96/96 | {v1} |

这样取样的过程可以被划分为两步，第一步，在现有的桶中选择一个桶；第二步，在选中的桶中随机选择一个节点。每个桶被选择的概率是其包含所有节点被选中概率的总和，更准确的说，选择一个桶Bd的概率是(d\*|Bd|)/Sn，其中，|Bd|是包含的度为d的桶Bd内的节点的个数。是常量，可以通过将概率转换为权值消除掉，如果被选中的桶包含不同的节点，那么这些节点被选中的概率是相等的。如下定理2-1证明。

定理2-1 在RW-bucket的两步取样操作与Simple-RW中的选择操作是等价的，也就是说，每个节点被选择的概率在这两个算法中是相同的。

证明：给定一个权值为wi的节点vi，那么在轮盘中vi被选中的概率为，对于两步取样算法，节点被选中的概率与在桶Bwi中选中vi的概率相等，其中选中Bwi的概率为，在桶Bwi中选中vi的概率为，由此可得：

无论何时桶中节点的权值有变化，当该节点的权值更新时都会被从当前的桶中移动到与其权值相对应的桶中。尤其对于基于BA模型的算法，一个现有网络中的节点每增加一条边，该节点的度都会从d增加到d+1，那么该节点就需要从Bd移动到Bd+1。

## 2.4 本章小结

本章内容主要介绍了关于幂律图相关的基础知识以及现有的创建无标度网络的算法。首先，介绍了无标度网络的相关概念，包括无标度网络，优先连接机制，BA模型的基本概念。其次，介绍了目前基于BA模型生成无标度网络的相关算法并给出代表性算法的主要思想，其中包括简单轮盘SimpleRW算法、随机选择算法、基于桶结构的RW-bucket算法。再次对现有算法在时间复杂度以及处理方面进行比较，分析了基于桶结构算法在大规模网络创建过程中的限制。最后分析了基于树结构创建无标度网络的可行性以及可扩展性。

# 第3章 基于RWBT的网络生成算法

## 3.1 问题分析

在每个节点的选择概率是固定不变的情况下，从一个轮盘中选择节点是简单的，但是，轮盘变成动态的之后，当每个节点的概率在随机选取的过程中有变化时，重复计算累计概率密度会耗费大量的时间。虽然累计概率密度仅仅需要在桶之间进行计算，由于桶的个数远远小于节点的个数，在查找目标节点时仅仅需要遍历很小的列表。尽管RW-bucket可以有效的降低问题规模，但是整体上依然存在无法处理的情况，在不影响处理结果的前提下通过最小化找到目标节点之前需要遍历的桶的个数可以有效加快节点处理效率。一种贪心的策略可以通过将桶按其权值降序排序来减少遍历冗余部分的桶，从而减少一大部分累计概率密度的计算。但是，这种算法在按照桶的权值保持桶有序的过程是费时的，每当加入的新节点引起桶的度数改变时都需要对桶序列进行重新排序，并且该种基于桶结构的算法依赖于最终分配的桶的个数，当在最坏情况下，网络中节点的度互不相同时，遍历桶的次数与遍历图中所有节点的次数相同，该种情况下不会降低需要操作的规模，并且由于桶是按照其包含节点的度值升序排序的，所以基于RW-bucket结构的算法会在调整桶按度排序的过程中耗费大量时间。

虽然现有的无标度网络生成算法在一般情况下可以明显缩小问题的规模，但是桶依然需要顺序遍历，并且在桶之间依然不可避免的需要计算累计概率密度。换而言之，如果与产生的随机数相关联的目标节点在桶Bi内，那么就需要计算从B1到Bi之间所有不为空的桶的累计概率密度，因此，这个现线性查找的时间复杂度为O(|B|)。

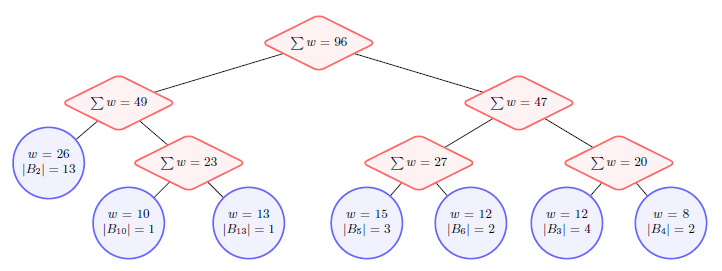
例如当需要从图2-2结构中选择一个待连接的节点，首先通过随机数生成算法产生一个随机数r=0.7，然后需要对轮盘桶结构进行两步操作，第一步，逐个遍历桶结构的索引值，由于每个桶都对应一个预先计算好的累计概率密度P(Bd)，这时只需要在遍历到桶时比较P(Bd)的值是否满足P(Bd)rP(Bd+1)，该例中P(B5)=0.63，P(B6)=0.76，由于P(B5)0.7P(B6)，所以B5是最终选定的桶。第二步，由于每个桶内包含的节点的度值都相同，所以只需要在选定的桶内随机选择一个节点作为目标节点与新加入网络中的节点进行连接，假设随机选到的节点为v4，那么连接到新节点后。v4的的度会相应的增加1，此时v4就需要从度为5的桶B5内移出，并添加到度为6的桶B6中，经过这样的选择步骤之后与v4相关联的两个桶B5和B6的总权值以及两个桶相应的累计概率密度P(B5)和P(B6)都需要重新计算，在对P(B5)和P(B6)重新计算的过程中，需要重新遍历所有的桶并按照每个桶包含的节点个数以及与节点相对应的度值d对Bd计算P(Bd)，上述重复计算的过程以及逐个遍历桶的过程是非常耗费时间的，而且也是非必要的。

通过以上分析，鉴于现有基于RW-bucket结构的算法在内存中处理大规模网络图数据时有接近30%的数据计算是非必要的，而RW-bucket结构本身所存在的线性遍历桶结构的不足，使得该结构不能满足快速高效地创建符合现实场景应用的无标度网络，本文旨在通过创建网络图数据的索引，提高索引在无标度网络创建的过程中的处理能力，提出一种改进的算法称为基于二叉树索引结构的RWBT (Roulette Wheel Bucket Binary Tree)算法。这个算法的主要思想是，将具有相同度的节点聚集到桶，然后将包含节点的桶作为二叉树的叶子节点，通过采用分治策略利用二叉树建立桶结构的索引缩小遍历规模，进而降低累计概率密度的重复计算，达到提高创建无标度网络效率的目的。

## 3.2 基本思想

RWBT结构通过在内存中保存一个桶列表，RWBT(roulette wheel binary tree)将桶按权值保存在带权二叉树上，首先，在RWBT结构上的随机选择算法，如下图3-1所示是与图1-1所示无标度网络相关联的RWBT结构，其中叶子上的圆形节点是桶集合，桶内包含现存网络中的节点，图3-1所示树中有包含26个节点的7个叶子节点，每一个叶子节点即为一个桶，桶的标记Bk则表示该桶内包含的节点的度均为*k*，桶的权值则与其在轮盘中的值相同，wi=i\*|Bi|。树结构中的菱形节点是RWBT结构的内部节点，该种节点不包含无标度网络中的节点，内部节点的权值是其左右孩子的权值的和，也就是其子树上桶的权值的和。为了避免混淆RWBT结构中的节点和生成的无标度网络中的节点，现将树结构中的节点表示为Rnode，Rnode由叶子节点和内部节点构成，这样的命名结构层次分明，便于区分，并基于RWBT结构提出了创建无标度网络的算法。

图3-1 图1-1所示网络对应的RWBT结构



## 3.4 算法描述

### 3.4.1 初始化无标度网络

在基于BA模型创建的无标度网络中，由于图的存储结构受到算法实现的具体方式影响，可以以邻接表、邻接矩阵等多种形式存储在内存中，所以在算法执行过程中暂且不考虑存储节点之间的边信息，只需要通过节点之间有对应关系表示两个节点之间存在连接。

基于RWBT结构的算法创建无标度网络的具体思想是：给定一系列需要创建无标度网络的节点，在初始化网络之前需先将所有的节点初始化保存到一个一维数组中，每增加一个节点则按照与现有网络中节点的权值相应的概率对现有网络数据图进行取样操作，对每一个新增加的节点逐个与现有网络中的m个节点建立连接，然后依次加入现有网络。算法3.1，3.2详细描述了这个初始化并从现存网络中的节点取样的过程。

|  |
| --- |
| 算法3.1 InitializeGraph |
| **Input**: m，m0 |
| **Output**: G(V,E)：包含m0个节点的初始化的无标度网络 |
| **Begin** |
| 1: *add m*0 *nodes to G*  2: **for** *i* from 0to *m* **do** |
| 3: addNode(*i*,1) **//**如算法3.1所示 |
| 4: updateTree(Tree)//如算法3.4所示  5: write the node ToGraph  6: addNode(m,m)  7: **return** *G* |
| **End** |

在网络初始化步骤中，初始化网络为含有m0个节点0条边，对于剩下的n-m0个节点，每个节点需要根据与现存网络中节点的度相关联的概率连接到已存在的m个不同的节点。由此步骤执行过程可知，算法的取样比较次数与m值存在必然的联系，所以图3-1所示结构是m值为2的情况下。

算法3.1先创建一个包含m0个节点的集合{*v*1,*v*2,…, }，对于需要新加入网络的下一个节点*vk*+1，节点*vk*+1需要从集合{*v*1,*v*2,…,}中选择m个不同节点连接产生m条新边，第2到6行描述了在逐个创建初始网络时，每个新节点都需要增加m条边。在插入新节点的过程中往往伴随着更新权值的操作的发生，第4行对增加节点以后的树结构自底向上进行更新权值的操作，第7行算法循环结束并返回初始的无标度网络。并由上述算法过程可知，对于每一个新加入网络的节点产生的新边只采用该边两端的节点之间的对应关系表示，并不保存两点之间的边信息。

### 3.4.2 取样操作

在RWBT结构中，随机选择的过程本文采用了一种分而治之的思想，从RWBT的根节点root开始，由于目标节点不是在该节点的左子树就是在其右子树，所以需要根据左右子树的权值进行选择。其中每个子树的权值代表了该子树对应的叶子节点被选中的概率，这个递归选择左右子树的过程会一直重复直到达到叶子节点。详细的选择过程如算法3.2所描述。

|  |
| --- |
| 算法3.2 SampleTreeNode算法 |
| **Input:** *Tree* |
| **Output:** 随机选择的节点*vi* |
| 1: Rnode*Tree*.root |
| 2: **while** Rnodeis not a bucket **do** |
| 3: Randomly generate *r* [1…(Rnode.wR + Rnode.wL)] |
| 4: **if** *r*Rnode.wL **then**  5: RnodeRnode.Lchild  6: **else**  7: RnodeRnode.Rchild  8: Bibucket(Rnode)  9:vkRandomly select an node in Bi |
| 10: **return** *vk* |

算法3.2详细描述了从现有网络中选取一个与新加入无标度网络的节点进行连接的具体步骤，其执行过程如下，首先初始化一个根节点root，这使得从上至下构建RWBT索引结构更方便，第1行初始化一个内部节点Rnode，并将其指向根节点root。按照算法第2-7行开始自顶向下依次遍历，依据遍历到的节点是否为叶子节点进行循环操作，其中叶子节点即为桶节点，如果非叶子节点，则利用随机数生成算法产生一个范围在1到左右孩子权值和之间的一个随机数*r*，并依次分别比较随机数*r*与左右孩子权值的大小，如果左孩子节点的权值不小于随机数*r*，则由左子树中向下进行遍历操作，如果左孩子节点的权值小于随机数*r*则由右子树向下进行遍历操作，第8行循环过程直到Rnode指向一个桶节点Bi结束，最后第8行由上述循环过程选中的桶Bi中随机选择一个节点*vk*作为该算法的返回值，算法结束。

以图1-1为例对上述算法进行详细描述，假设需要在v0,…,v25之间随机选择一个节点与新加入网络的节点v26相连接，另外假定均匀地生成了如下一个值为[0,1]的随机数序列{0.83,0.12,0.54,0.63}。首先从根节点开始遍历，从左右孩子节点中按照与左右孩子节点的权值相应的选择概率进行选择。在图3-1示例的树结构中根节点左右孩子被选中的概率分别是 和 ，根据产生的第一个随机数0.83，根节点的右孩子的概率满足 < 0.83，所以需要选择根节点的右孩子。然后从根节点的右子树开始重复上个根据左右孩子节点的权值对应的概率选择相应节点的步骤，根据随机数序列的第二个随机数，如果该节点的左子树的概率 > r，则选择左子树，否则选择右子树进行重复遍历。由于随机数序列的第二个随机数为0.12，第三个为0.54，> 0.12，0.54，则第二步第三步均选择树的左分支进行遍历。对该树循环遍历的最终结果是选择B5作为目标节点所在的桶，由于B5内的节点的概率都为5，则在B5内v4,v9,v11每个节点被选中的概率都为0.33，根据上述产生的随机数序列第四个随机数0.63,由于p(v4)=0.33<0.63，所以选择第二个节点v9与新加入网络的节点v26相连接。由于B5桶内每个节点被选中的概率都相同，所以避免了在桶内计算累计概率密度。

RWBT结构的优势在于其避免了遍历大量的桶列表，是由于RWBT结构在每个内部节点包含了其分支上对应的桶的权值和，因而避免了在自顶向下遍历节点并选取待连接节点的过程中对节点累计概率密度的重复计算。

### 3.4.4 更新RWBT权值

每当桶的权值发生了变化，其对应的祖先节点的值也需要相应地更新，例如对于基于BA模型创建无标度网络的算法，当一个具有度为d的节点vk被选中与新加入网络的节点进行连接，该节点vk的度随之增加，那么该节点则需要从桶Bd内移动到Bd+1，相应地，桶Bd与Bd+1对应的内部节点的祖先节点也需要更新，如上述例子中当v9被选中后，需要将v9从B5移动到B6。那么B5和B6的权值也需要随之更新。 ，相似地，其父节点，祖先节点到根节点需要自底向上分别进行如下更新2728，4748，9697。详细过程如算法3.4所描述。

|  |
| --- |
| 算法3.3 updateTree 算法 |
| **Input**: *Tree* , *v* , *oldweight* , *newweight* |
| 1: remove *v* from the old bucket |
| 2: if *Boldweight* is empty then |
| 3: deleteNode(*Tree*,Rnode(*Boldweight*)) //算法3.5 |
| 4: **else** |
| 5: Rnode=Rnode(*Bwold*) |
| 6: Rnode.w=wold\*|Bwold| |
| 7: **while** RnodeTree.root **do** |
| 8: Rnode Rnode.parent |
| 9: Rnode.w = Rnode.Rchild.w+Rnode.Lchild .w |
| 10: add *v* to the new bucket |
| 11: **if** Rnode(*Bwnew*) is not in the *Tree* **then** |
| 12: addNode(*Tree*,Rnode(*Bnew*)) |
| 13: else |
| 14: Rnode = Rnode(*Bnew*) |
| 15: Rnode.*w* = *wnew* \* |*Bnew*| |
| 16: **while** Rnode*Tree*.root **do** |
| 17: Rnode Rnode.parent |
| 18: Rnode.w = Rnode.Rchild.w + Rnode.Lchild.w |

### 3.4.3 插入节点

在上述节点更新度值导致桶的权值发生变化产生新权值的桶Bwnew，并且Bwnew不存在时，则需要将新产生的桶作为树的叶子节点插入到树中，另外，当节点移动后导致先前所在的桶Bwold为空之后，Bwold则需要从RWBT中删除。这两步对树结构的插入删除操作将分别下文addNode算法与deleteNode算法详细描述。

插入节点的过程可以被看作为在二叉树中从根节点开始随机选择一条到桶节点的路径。显而易见的是，任意一棵包含所有节点的二叉树都可以被用来作为随机选择节点的结构，一般而言，在任意一棵二叉树上到达桶节点，即叶子节点的时间复杂度均为O(LCW)，其中LCW表示这棵树的平均编码长度，值为。根据这个公式可知最简单的方法就是将桶保存在一棵完全二叉树中，这样新产生的桶会被添加到树的末尾中第一个空闲的位置。这种插入方式保证了树的深度可以保持在log*n*，进而同时保证了树的插入，删除，随机选择节点的操作的时间复杂度均为O(log*n*)。同时出于避免树的所有叶子节点的选中概率相等的考虑，这种插入方式是最优的方案。但是由于树的叶子节点上各个桶的权值是互不相同的，在无标度网络结构中，所以将所有叶子节点放到相同的深度是不合理的。相反，由于无标度网络中大部分节点具有相同的选中概率，则需要将具有更高选中概率的叶子节点放到更低层次的深度，这样这些具有较高被选中概率的节点可以在较短短的时间内很容易的被选择，进而提高算法效率。

通过以上分析，本小节将关注点放在优先连接机制的随机性是如何维持轮盘二叉树RWBT在创建无标度网络的过程中的平衡。通过对优先连接过程的观察发现了一些具有重要意义的性质：一些具有较小索引值的桶，相应的，这些桶内包含的节点的度也较小，比如B2，B3则具有更高的选中概率。将每个桶被选中的概率记为P() = ，其中，。因此，桶对应的索引值越小，该桶被选中的几率越大。这种机制导致的结果就是，桶内包含的节点的度越小，该桶在树中的编码长度应该越短，越应该靠近根节点。除此之外，包含较小度的节点的桶在无标度网络创建过程中更不易为空更稳定。而恰恰相反的是，类似或者包含更高的度值的节点的这种具有较高索引值的桶的稳定性，在创建无标度网络时不断增加节点的过程中更容易为空，因为由于优先连接机制的性质，这类桶经常只包含几个节点甚至一个，当桶内的节点被选中与新加入网络的节点进行连接之后，该桶会很快变空，那么相应的这类桶也会被从轮盘二叉树RWBT结构中删除。详细插入桶节点的步骤如算法3.4所述。

|  |
| --- |
| 算法3.4 addNode to the tree |
| **Input**: *Tree* , *vk* , B*d* //d为节点的度，*vk*为选中连接的节点 |
| **Begin** |
| 1:Init *currentHufNode* = *Tree.root* |
| 2: Init B*d* // Bd为包含度为d的节点v的集合 |
| 3: **if** B*d* != **then** |
| 4: add *vk* to B*d* |
| 5: updateTree(B*d*.parent) //算法3.3 |
| 6: **return** |
| 7: create B*d* |
| 8: add *vk* to the new bucket B*d* |
| 9: newNode = TreeNode(B*d*) //初始化桶B*d* 为树的叶子节点 |
| 10: **while** the newnode is not a leaf **do** |
| 11: **if** currentHufNode is a leafnode **then** |
| 12: init midNode=TreeNode |
| 13: midNode.LChild = currentHufNode |
| 14: midNode.RChild = newNode |
| 15: **if** currentHufNode.LChild == **then** |
| 16: currentHufNode.Lchild = newNode |
| 17: **if** currentHufNode.RChild == **then** |
| 18: currentHufNode.Rchild = newNode |
| 19: updateTree(*Tree*) //算法3.3 |
| **End** |

算法3.1详细描述了在RWBT结构中插入节点的步骤，其中代码第1至2行为初始化的过程，本段代码需要初始化一个指针指向Tree的根节点root，并且将待插入的桶根据其对应的度初始化为树的叶子节点。第3至6行判断桶是否为空，如果不为空则直接将选中的需要与新加入网络的节点相连接的节点vk添加到桶B*d*中，并且相应地更新桶B*d*对应的分支上所有节点的权值。如果节点vk需要添加到的桶为空，则需要新创建一个与vk的度d一致的桶B*d*，并将B*d*初始化为RWBT的节点。第10至18行为从根节点root开始自顶向下遍历树的节点查找新创建的桶B*d*需要插入的位置的过程，其中11至14行为遍历到的当前节点为树的叶子节点则新建一个内部节点类型的中间节点，使该中间节点指向当前节点的父节点，当前节点作为该中间节点的左孩子插入，新建的桶节点Bd作为该中间节点的右孩子插入，15至16行为遍历到的当前节点为内部节点且左孩子为空，则将新建的桶节点Bd作为当前节点的左孩子插入，17至18行为遍历到的当前节点为内部节点且右孩子为空，则将新建的桶节点Bd作为当前节点的右孩子插入，最后调整插入节点相应分支上的权值，算法结束。

例如图1-1所示的无标度网络对应的拓扑图，其中包含度为2的节点的桶B2含有13个节点，更不容易被删除，相应的桶B2在每次的取样选择中具有更高的概率，即此类桶距离根节点更近，与此相反的是，B13是很不稳定的，虽然B13被选中的几率也很大，但是它只包含一个节点v1，当v1作为目标节点被选中以后，需要被移动到B14中，那么B13就被置空进而被删除。

### 3.4.3 删除节点

当需要从RWBT结构中删除一个桶节点时，只需要断开该节点与树中节点的连接，然后，将已删除节点的父节点替换其孩子节点。这个操作可以保证树中的内部节点都有孩子节点，从而确保整棵二叉树的完全性。详细过程如算法3.1所述。虽然这种随机删除树节点的过程会导致暂时的失衡，增加平均编码长度，但是，树的平衡性会在新节点插入后恢复。

|  |
| --- |
| 算法3.5 deleteNode from the tree |
| **Input**: *Tree*, *oldBucket* |
| **Begin** |
| 1:remove the *oldbucket* from the bucket set |
| 2: *father* *oldBucket*.parent |
| 3: *sibling* |
| 4: **if** *father*.Lchild == *oldBucket* **then** |
| 5: *sibling* = *father*.Rchild |
| 6: **else** **if** *father*.Rchild == *oldBucket* **then** |
| 7: *sibling* = *father*.Lchild |
| 8: **if** *father**root* **then** |
| 9: *grandfather* *father*.parent |
| 10: **if** *grandfather*.Lchild == *father* **then** |
| 11: *sibling* = *grandfather*.Lchild |
| 12:  **else if** *grandfather*.Rchild == *father* **then** |
| 13: *sibling* = *grandfather*.Rchild |
| 14: *sibling*.parent = *grandfather* |
| 15: updateTree(father) //算法3.3 |
| **End** |

算法3.1首先将需要删除的桶*oldBucket*从桶集合中删除，并初始化一个指针*father*记录旧桶的父节点，以及一个空节点*sibling*用来保存旧桶节点*oldBucket*的兄弟节点。代码第4至5行为判断*oldBucket*节点是左分支则将*sibling*指向其兄弟节点，即*father*节点的右孩子，第6至7行为判断*oldBucket*节点是右分支则将*sibling*指向其兄弟节点，即*father*节点的左孩子。由于此时father节点只有一个孩子节点，且*father*节点为树的内部节点Rnode，不包含实际网络中的节点，则为了缩短从根节点到达叶子节点的距离需要将该类型节点进行删除，代码第10至14行为删除*father*节点的过程。代码第8行需要对于*father*节点是否为根节点进行判断，如果非根节点则需要记录其父节点，记为*grandfather*节点，代码第10至11行判断如果*father*节点为*grandfather*节点的左孩子则将*sibling*节点作为*grandfather*节点的左孩子，代码第12至14行判断如果*father*节点为*grandfather*节点的右孩子则将*sibling*节点作为*grandfather*节点的右孩子，并将*grandfather*节点作为*sibling*节点的父节点，第15行为更新此更改分支上权值的过程，如果*father*节点为根节点则只需更新该分支上的权值，算法结束。

### 3.4.3 RWBT算法

基于前文对RWBT结构中各个操作的详细解释，算法3.6详细描述了基于RWBT结构创建无标度网络的过程中向初始化的包含m0个节点0条边的现有网络中每添加一个节点产生m个连接的步骤。

|  |
| --- |
| 算法3.6 RWBT算法 |
| **Input**: n=|V|，m，degree |
| **Begin** |
| 1: Init a hashset to contain m nodes to be selected |
| 2: **for** *i* from0to *m* **do** |
| 3: **while** not find an Random node **do** |
| 4:  sample a bucket Bi use the sample() // 算法3.2  5:  select a position in the bucket  6: select a ndoe to be connected by the nodeid  7: insert the selected node to the set  8: *updateTree*() //详见算法3.3  9: **End** |

算法3.6首先初始化一个集合用来记录将被选中作为与新加入网络的节点vk相连接的m个节点，代码第2至8行为选取m个节点的过程，其中第4至8行即为对vk选取一个连接节点的过程，先根据算法3.2在RWBT中选取一个桶节点Bi，然后再从选取的桶Bi中按照桶内节点的概率选择一个与vk相连接，最后对增加节点后的RWBT结构更新相应分支上的权值，算法结束。

## 3.5 算法分析

基于RWBT结构的算法对于包含n个节点，每加入一个新节点需要添加m个连接的无标度网络来说，在轮盘二叉树RWBT结构中每建立一条新的连接需要遍历Lcw次，所以总的取样的时间复杂度是O(nm\*|LCW|)，维护该结构即每添加一条连接需要自底向上对树的权值进行更新，更新操作的时间复杂度为O(Lcw)，所以总的RWBT结构维护的时间复杂度为O(nm\*|LCW|)。

## 3.5 算法优化

最后，提出了一种通过替换桶节点的删除插入操作，例如给定一棵RWBT树*T*，假设需要从只包含一个节点vi的桶Bd中进行取样操作，但是Bd+1*T*，如上述例子中v1B13，但是当B13桶中唯一的节点v1被选中，该节点的度相应地加1之后，v1需要从B13桶中移到B14中，B13被置空但是B14不存在的情况下，则只需要将B13重命名为B14，而不是删除B13然后插入B14。这种替换操作在节点规模到达一定程度之后会明显减少删除，插入的次数，从而进一步提高算法的效率。其中替换操作的折减系数 ，其中*rep*，*del*，*ins*分别表示替换次数，删除次数，插入次数。详细步骤如算法3.1所描述。

|  |
| --- |
| 算法3.7 RWBT+算法 |
| **Input**: *Tree*, *oldDegree，newDegree* |
| **Begin** |
| 1: *oldBucket*Bucket with the *oldDegree* |
| 2: **if** *oldBucket.*size == 1 and *newDegree*  *Tree* **then** |
| 3: *oldBucket*.degree*newDegree* |
| 3: delete *oldDegree* |
| 4: add *newDegree* |
| 5: updateTree(*Tree*) //算法3.1 |
| 6: **else** |
| 7: *oldBucket*.size - 1 |
| 8: updateTree(Tree) //算法3.1 |
| 9: **if** *oldBucket*.size **==** 0 **then** |
| 10: deleteNode(*oldBucket*) //算法3.1 |
| 11: **if** *newDegree*  *Tree* **then** |
| 12: addNode(*vk,newDegree*) //算法3.1 |
| 13: add *vk* to the bucket with higher degree |
| 14: **End** |

算法3.1第2-5行详细描述了以替换桶名称代替删除旧桶并添加新桶的过程，第2行判断需要更改的桶内的元素是否唯一并且与待添加的度值相关联的桶在树*Tree*中是否存在，如果旧桶内元素唯一并且*Tree*中不包含新桶，则将旧桶的度值标号改为新度值，第3-5行为将旧度值从列表删除，添加新度值，并更新整棵树中每个节点权值的过程。第7-12行则为不满足上述改进条件的情形下，按照基础算法删除旧桶并增加新桶的过程，如果选中的旧桶中的节点个数不唯一，则将旧桶的个数减1，第8行为更新相应分支上各个节点的权值的操作，第9-10行为在旧桶内的节点个数为0时删除该桶的操作，第11-13行为判断与选中连接的节点*vk*产生的新度相关联的桶在*Tree*中是否存在，如果不存在则添加新桶，如果存在则将选中的节点*vk*添加到更高度值的桶中的操作。

## 3.6 本章小结

本章首先对已经存在的无标度网络创建方式存在的问题进行了详细的分析，主要是由于逐次遍历网络中的节点而导致累计概率密度的重复计算的问题，进而导致了无标度网络创建效率低的问题。通过进一步分析网络遍历方式以及无标度网络中优先连接机制的特性发现，这些问题可以通过改进节点的存储结构从而减少累计概率密度的计算加以解决，在此基础上，本章提出了基于最优二叉树思想的RWBT结构及其相应的改进遍历策略，通过构建网路图数据的索引，避免了对累计概率密度的重复计算以及冗余节点的比较，然后根据。具体描述了算法思想，并且通过示例详细介绍了RWBT算法创建无标度网络过程中添加删除节点的操作步骤。

# 第4章 基于RWTT生成算法

## 4.1 问题分析

由上一章中分析可知，在RWBT中随机选择的过程可以被认为是一种随机地从根节点开始寻找一条到桶节点最短的路径。换而言之，当前研究正努力达到的目标是研究一种能够尽可能缩小最短平均编码长度的算法。

定义4-1 哈夫曼树 对一棵带权值的二叉树进行，带权路径长度最小，并且权值较大的节点距离二叉树根节点较近。

定义4-2 哈夫曼编码 是一种可变长度的最优前缀编码格式，该种编码方式对每一个符号选择特殊的标记进行表示，最终形成的编码与字符出现的概率成正比，并且是平均长度最短的编码方式。

由于当前研究的主要目的是尽可能缩小二叉树的平均编码长度，进而提高算法的处理效率，所以试图寻找一种能将查找节点的时间降到最低的树结构，很显然地，在树结构中对一个节点进行取样的过程耗费的时间与该节点对应的桶距离根节点的路径长度成正比，该路径长度也就是在左右子树之间需要进行的比较操作的次数。在上一章中为了降低取样过程的时间，以便桶对应的叶子节点可以更容易地被找到，在插入桶的过程采取了具有较高权重的桶对应的节点距离根节点更近的插入操作。这种思路更接近于构造哈夫曼树的过程，在哈夫曼树结构中访问频度更高的节点更接近于根节点，这样距离根节点越近的节点被分配到的编码长度越短。更具体地说，二叉树中的每一个桶都会被分配一个唯一的二叉树编码，这种编码结构用第i个0或1表示该层的目标节点在左或右子树。例如图3-1中，B3所在节点对应的编码为110，这表示当从根节点开始查找B3时的遍历分支的顺序为“右右左”的遍历方式。

上一章节的算法思路是缩短具有较大选中概率的桶距离根节点的编码长度，为了最大化的减少算法的执行时间，应该尽量减小对桶进行取样操作耗费的时间，以及维护树结构的时间，也就是需要降低对树进行更新权值，添加新桶，删除空桶的时间。由上述分析，虽然拥有最短平均编码长度的哈夫曼树会大大降低在取样操作中耗费的时间，但是其他包括对树结构进行维护的时间也需要降低。换而言之，基于权值平衡树的结构的基本原理是通过降低执行时间来保证平衡树的自身调整的过程不会过度占用时间。因此，采用类似最优哈夫曼树的编码结构这种计算上开销很高的基于权值平衡的算法来创建无标度网络会因为过度关注于优化平均编码长度而耗费时间，这种方式对于解决无标度网络的创建这个问题是不适合的。

## 4.2 算法思想

定理4.1： ，其中2x < y。

证明：对不等式两边分别除以 ，得到不等式，在n趋于无穷大时对该不等式求极限得，等于，又由于x < y，且x,y均不为0，可得所以不等式得证。

根据定理4.1的证明可知在节点规模n扩大到一定程度之后，时间复杂度随着分支d的增加降低，那么创建无标度网络的时间在不考虑随节点增加而动态调整平衡的操作过程也会随着网络图的存储结构RWTT的分支的增加而降低。

上述第3章中对RWBT结构的改进经过实验验证，仅仅在节点规模大于107的情况下，每添加一个新节点需要添加的边数大于20以上创建无标度网络的效果才会有显著改善，因此，通过分析RWBT结构处理节点的时间复杂度发现结构中每添加一条连接需要遍历Lcw次，其中Lcw是最短路径长度，在二叉树结构中具体值为，其中n为节点的规模，那么在将树结构的分支扩展为三叉树时，Lcw会相应地降为，由此可知，可以通过扩展RWBT结构的分支数来进一步缩短平均路径长度。

通过以上分析，并综合第三章的遍历策略可知，在RWBT结构基础上增加一个分支，可以进一步提升创建无标度网络的效率。

## 4.3 算法描述

通过对算法时间复杂度的分析可知，算法执行效率较大程度上依赖于平均路径长度LCW，这主要是因为在查找节点的待插入位置的过程中，随着数据规模n以及每增加新节点添加的连接数m的增大，遍历现存网络中节点的次数也会相应增大。而实际上由于优先连接机制的特性，网络中大量的节点拥有较小的连接概率，因此只有一小部分节点需要遍历。如果将二叉树结构的RWBT结构扩展为三叉树，进而缩短遍历操作的平均路径长度，只针对节点应插入的位置所在分支进行相应的计算概率操作，那么则会进一步加速遍历的过程，进而提高算法的效率。接下来详细介绍基于扩展分支思想的RWTT结构的构建过程，具体步骤如算法4.1所示。

|  |
| --- |
| 算法4.1 RWTT算法 |
| **Input**: *Tree* , *vk* , B*d* //d为节点的度，*vk*为选中连接的节点 |
| **Begin** |
| 1:Init currentNode = *Tree.*root |
| 2: Init B*d* // Bd为包含度为d的节点v的集合 |
| 3: **if** B*d* != **then** |
| 4: add *vk* to B*d* |
| 5: updateTree(B*d*.parent) //算法3.3 |
| 6: **return** |
| 7: create B*d* |
| 8: add *vk* to the new bucket B*d* |
| 9: newNode = TreeNode(B*d*) //初始化桶B*d* 为树的叶子节点 |
| 10: **while** the newnode is not a leaf **do** |
| 11: **if** currentNode is a leafnode **then** |
| 12: create midNode=TreeNode |
| 13: midNode.LChild = |
| 14: midNode.RChild = currentNode |
| 15: midNode.TChild = newNode |
| 16: **if** currentNode.LChild == **then** |
| 17: currentNode.LChild = newNode |
| 18: **if** currentNode.RChild == **then** |
| 19: currentNode.RChild = newNode |
| 20: **if** currentNode.TChild == **then** |
| 21: currentNode.TChild = newNode |
| 22: **while** newNodeTree.root **do** |
| 23: newNode newNode.parent |
| 24: newNode.w = newNode.TChild.w + newNode.RChild.w + newNode.LChild.w |
| **End** |

执行算法4.1之前需要经过算法3.2完成对需要进行连接的节点的取样操作，然后执行该算法。首先初始化一个指针指向树Tree的根节点root，再初始化一个桶bucket记录当前的桶节点，代码第3至6行是对在RWTT结构中是否存在相应度值的桶的情形进行判断，如果存在则直接将插入选中连接的节点vk添加到桶Bd中，并更新分支上的各个节点的权值。第7-9行是在桶Bd不存在的情形下创建一个对应度值d的新桶Bd，将选中的节点vk添加到Bd中，并将初始化为RWTT结构的节点的过程。代码第10至21行则为将Bd作为在RWTT树中查找到的位置currentNode节点的孩子节点插入的过程，在对桶Bd执行插入操作的过程中，从根节点开始自上而下遍历整个树结构，第11至15行表示当前三叉树的除叶子节点以外的所有节点都有三个孩子节点，需要创建一个中间节点midNode并将当前桶Bd作为midNode的最右节点插入。代码第16行至21行表示遍历到的当前节点至少有一个孩子节点为空，则将该桶Bd作为当前节点的该孩子节点插入。第22行至24行表示从Bd插入位置的父节点开始自底向上更新该分支上所有节点权值的过程。更新完成后算法结束。

## 4.4 算法分析

在第一小节详细讨论了哈夫曼编码这种理论上最优的二叉树需要进行的最少的取样操作的次数，并且客观分析了哈夫曼树高昂的计算代价以及提出一种接近最有哈夫曼树的轻量级的贪心算法。当前算法希望达到的效果是当在RWBT中插入一个节点时，树中节点的平均编码长度可以保持最小。如果轮盘是静态不变的，那么哈夫曼树将会是最优方案[19]。但是出于该结构在动态添加节点的过程中较大的计算开销考虑，哈夫曼树的效率是不高的。所以原始的哈夫曼树对于创建基于BA模型的无标度网络是不适合的，因此选择概率一旦改变则需要重建二叉树，这种情况出现在节点的度增加或者产生新节点的连接时。

为了设计一种针对权值平衡树的在线算法，可以将原始哈夫曼树进行改进，比如FGK算法[22,25,29]和Vitter[48]，这两种算法均支持动态更新概率，并且能够保证单个节点的插入时间为O(Lcw)。但是这种改善后的编码算法最重要的缺点是不支持动态删除节点以及降低节点的权值，即不支持当一个节点与新加入的节点连接后，从原来的桶移动到新桶的过程中导致旧桶为空而需要删除旧桶的操作。同理，该种算法也不适用于当一个节点被从桶中取样抽中并产生新连接后，该节点的度相应的增加需要移动到新桶，因此而导致的原桶的权重降低的过程。因此，就目前研究来看，暂时没有找到一种适应性哈夫曼编码算法可以有效支持删除节点的过程。综上所述，试图研究一种对于权值平衡树的自适应哈夫曼编码算法是不可行的。所以最优的并不代表是最适合的。

## 4.5 算法优化

经过实验研究发现当树的分支从三扩展到四的时候，在树的内部节点向下一层遍历查找新加入的节点需要插入的位置时，按先前的逐个遍历节点的方式会大大降低算法的处理效率，是因为107个节点规模在RWBT结构中的LCW最大为11，在改进后的分支为3的RWTT结构中的LCW最大为6，如果通过再进一步扩展分支进而降低LCW会使得RWTT近似退化为一维数组，又由于在插入节点的过程中涉及到节点的删除操作，逐个比较子树的过程会由之前的多路查找退化为顺序查找，这样使得RWTT结构的优势被弱化。

根据以上思路发现在扩展树结构的分支过程中当分支过多，如果依然按照逐个比较子节点的顺序选择满足条件的节点会大大降低遍历效率，本章节提出了基于RWMT(roulette wheel multi tree)结构的多叉排序树结构RWMST(roulette wheel multi sorted tree)，通过对在插入结点过程树结构的内部节点排序来降低后续查找过程的比较时间。

通过对上述算法的分析，本小节采用了一种基于启发式的贪心策略对删除策略进行改进，在对RWTT扩展分支的基础上，对各个分支上的节点按其权值进行排序处理，进而保持RWMST结构在逐步删除过程中的近似平衡。算法4.2为以三叉树结构为例对RWMST算法的插入过程进行阐释的具体步骤。

|  |
| --- |
| 算法4.2 RWMST算法 |
| **Input**: *Tree* , *vk* , *d* |
| **Begin** |
| 1:Init *Rnode* = *Tree.root* |
| 2: Init *Bd*= buckets[*d*] |
| 3: **if** *Bd* != **then** |
| 4: add *vk* to *Bd* |
| 5: updateTree(*Bd*.parent) //算法3.1 |
| 6: **return** |
| 7: create a new bucket *Bd* which contains nodes with *d* |
| 8: add *vk* to the new bucket *Bd* |
| 9: newnode TreeNode(*Bd*) //将桶初始化为树节点 |
| 10: **while** the Rnode is not a leaf **do** |
| 11: **if** Rnode.Tchild.*w* > Rnode.Rchild.*w* **then** |
| 12: RnodeRnode.Rchild |
| 13: **else if** Rnode.Rchild.*w* > Rnode.Lchild.*w* **then** |
| 14: RnodeRnode.Lchild |
| 15: **else** |
| 16: RnodeRnode.Tchild |
| 17: midRnodeInitialize a new internal TreeNode |
| 18: midRnode.parentRnode.parent |
| 19: midRnode.Tchildnewnode |
| 20: midRnode.RchildRnode |
| 21: midRnode.Lchild |
| 22: **for** pRnodenewnode.ancesteors() **do** //自底向上更新 |
| 23: pRnode.*w*pRnode.Lchild.*w* + pRnode.Rchild.*w* + pRnode.Tchild.*w* |

算法4.2为对创建无标度网络的插入操作优化的详细过程，代码第1至9行表示判断待插入的桶Bd是否存在在当前网络中，如果存在则直接将选中待连接的节点vk添加到桶中并更新与该桶相关联的节点的权值(3至6行)。代码第7至23行表示如果不存在则需要新建与更新后的节点的度d相应的桶Bd，将Bd存入RWMST并更新相应分支上节点的权值，其中第10至16行表示在从非叶子结点自顶向下遍历的过程中需要将Rnode节点的三个孩子分支节点的权值大小逐个进行比较，选择权值较小的分支向下遍历，代码第17至21行表示经过以上步骤找到了Bd需要插入的位置，将Bd作为选中叶子节点的最右孩子插入的过程。代码第22至23行表示在插入操作完成后，对相应分支自底向上更新节点权值的过程，更新完成后算法结束。

## 4.6 本章小结

本章首先针对第3章提出的RWBT结构进行了详细的分析，指出基于RWBT结构的算法在对更大数据规模的数据图的处理中的扩展能力有限，然后提出了扩展RWBT结构的分支的理论依据和正确性保证。详细阐释了基于RWTT结构的算法创建无标度网络的步骤流程，并分析了进一步扩展RWTT结构分支的局限性，进而提出了RWMST结构，并比较了基于RWTT结构的算法与基于RWMST结构的算法在内存中创建无标度网络效率上的差异，并给出了详细的创建流程。

# 第5章 实验结果与分析

## 5.1 实验环境

本篇论文针对已有算法存在的缺陷，提出了基于轮盘二叉树结构RWBT的算法。然后，提出了基于轮盘树结构的多叉树结构RWTT以及该算法的优化算法基于轮盘多叉树的排序算法RWMST。通过在产生的随机数数据集上进行实验，对上述算法的有效性进行验证。实验主要从多个方面进行测试针对不同数据集分别改变参数m的值，首先验证基于轮盘二叉树结构RWBT的算法和基于轮盘树结构的多叉树结构RWMT的算法对创建不同规模的无标度网络的执行效率。其次，针对多种影响因素将RWBT算法，RWTT算法，RWMST算法进行比较，对比RWMST算法中运用的排序策略，验证RWMST算法的高效性。

接下来首先介绍实验环境。然后介绍实验用到的数据集和评价指标，最后对各个算法的性能进行比较，验证所提出的算法的有效性和可行性。

实验所使用的软硬件平台是Intel(R) Pentium(R) 主频为2.9GHz CPU，500GB的硬盘，4GB的RAM内存和32位的Windows 7 Professional OS，实验编程环境为Eclipse；编程语言为Java；JDK版本为Java 1.8 SDK。

实验中，涉及到基于BA模型创建无标度网络的算法包括(1) 基于轮盘二叉树结构RWBT的算法;(2)基于轮盘树结构的三叉树结构RWTT;(3)基于轮盘多叉树的排序算法RWMST;(4)创建无标度网络的基本算法SimpleRW，SA，RW-bucket。由于实验中涉及到的参数m0对各个算法生成的无标度网络不会产生影响，则本实验部分对于所有算法中关于参数m0的初始值均设置为m0=m。

## 5.2 数据集及评价标准

由于本文算法研究的是在内存中快速创建无标度网络，所以不需要涉及文件的读写过程，因此实验部分采用的数据集是采用一种随机数生成器在程序执行时产生n个节点规模的数据。

为了研究论文所述算法的性能，实验将对影响算法的几个重要因素进行测试，其中相关的实验参数见表5-1。

表5-1 实验参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| *m* | 测试数据中每新添加一个节点增加的边数 |
| *m*0 | 初始图中节点个数 |
| *n* | 测试数据集的规模 |

实验中，由于本文所涉及算法的随机特性是相同的，这些算法都是通过网络中节点的度进行随机选择的，所以本章实验内容主要分析了算法的效率，只验证了算法的执行时间。

对于优先连接机制来说最主要的限制因素是需要在内存中同时存储所有节点的索引以及度值，这导致算法的空间复杂度为O(|V|)，比如节点的索引按32位整数存储，当生成一个含有1010个节点的无标度网络时。至少需要4GB的空间在内存中存储该网络对应的轮盘结构，因此目前机器上仅支持生成109个节点的无标度网络。

## 5.3 性能比较与分析

在研究每新增一个节点需要添加的边数m对算法效率的影响时，保持节点规模n的值不变。大多数基于优先连接机制的模型产生的无标度网络中的节点的度都存在一种“rich-get-richer”的现象，即，一个节点在先前的操作中被选中的可能性越大，那么该节点在接下来的操作中也更有更大的被连接的可能。这种模型会导致轮盘中的节点形成长尾分布的概率分布。本文中提到的每一种基于轮盘的算法都高度依赖于与应用到的模型相关的概率分布的统计特征。其中，SA算法的性能优劣取决于网络中节点的平均接受概率。同样地，RW-bucket算法的性能会受到参数|B|大小的限制，对于RWBT来说需要得到网络中节点对应的桶的分布形态，以及其随时间增长的模型。

本章节通过实验分析了几种基于轮盘结构的不同算法在创建基于BA模型的无标度网络时的不同表现，更具体地讲，本文着重关注了影响这几种算法效率的几个参数，例如，对于每种算法从轮盘中选择一个节点需要进行的操作的次数，表5-1列出了每种算法从轮盘中选择一个节点需要的比较次数。例如，在创建含有1010个节点的BA无标度网络的过程中，使用简单轮盘算法SimpleRW选择一个目标节点平均需要遍历1.6亿个节点，SA需要进行1320次随机取样操作直到选择的节点被接受，在RW-bucket中需要遍历的桶的平均个数为22.5，在RWBT中遍历的平均深度为6.71，在RWTT中遍历的平均路径长度为4.66，以及RWMST不会缩短平均路径长度但是会明显减少结构内部节点的比较次数进而提升算法执行效率。

接下来的实验，将分别对比本文中讨论的四种算法在内存中针对不同参数的设定创建不同规模的无标度网络的执行时间以及各个算法对内存的占用情况。

### 5.3.1 新增边数的影响

对于表5-1列出的在m=2,5,10,20,30,50,100时的执行时间对比的结果表示了RWMST在处理节点规模较大的图上比其他算法都高效。但是对于较小的图来说，随机接受算法比RWMST更高效，尤其在m特别大的情况下，大体上来说，RWMST是唯一一个随n的增长呈线性增长的算法。

下图5-1表示了创建无标度网络的时间是如何随着m值的改变而变化的，大体上来说本文讨论的任何一个算法的执行时间都可以优化到关于m的线性时间，但是整个轮盘结构需要遍历m(n-m0)次，这就导致了轮盘结构的调用也会随着m的增长而增长，唯一例外的情形是2-2-2中讨论的SA算法，该算法在节点的度分布均匀的情形下选到合适节点的平均取样时间与m成反比，其时间复杂度为O()，也说明了m的值越大，在轮盘中取样的速度越快。但是由于随机接受机制的比率常常远远大于期望值，所以在解决现实世界中的实际问题时，RWBT与RWBMT的效率更高。另一方面，通过增加m的阈值依赖于n，创建的网络会变成节点密度较高的稠密图，这样导致创建的网络不再符合无标度特性。因此，在m远远小于n时创建的无标度网络更能满足对现实世界网络结构研究的需要。例如，当需要对含有百万亿个节点的万维网的拓扑结构进行研究时，平均每个页面的出边不超过150。如图5-1展示了图规模非常大时，对于任意m值RWBMT算法都是最优的。

n=106

n=107

### 5.3.2 初始节点个数的影响

在研究每增加一个节点需要增加的边数m对算法的影响时，保持结点规模107不变

图5-1创建无标度网络的时间随不同的m0增长情况，进一步验证了初始化时节点的个数m0对无标度网络的创建算法不会产生影响。

在算法的执行时间方面，如图5-1所示，横坐标表示所求点组s值，纵坐标表示运行时间，时间单位是ms。实验中，采用PointWise[10]算法计算组RWMST点组，由图中可以看出，每个算法的执行时间都随着s值的增大而增大。当s值较小时，算法的执行速度都很快，且算法随着s值的增大，基本保持持平，没有太大差别。但算法相比执行时间会有所增加，因为算法是在算法计算结果的基础上对点组的进一步排序处理，必然增加了执行时间。从图中还可以发现，随着s的增大，算法和的执行时间呈指数型增长，时间增加的原因是：随着s的增大，前m个层上点的个数会急剧增加，导致组点组的个数增加，数据规模变大。但是，基于的改进算法执行速度在很大程度上要优于其他两个算法，最快提高了十倍之多。

### 5.3.3 数据规模的影响

在算法的执行时间方面，如图5-1所示，横坐标表示所求点组s值，纵坐标表示运行时间，时间单位是ms。实验中，采用PointWise[10]算法计算组RWMST点组，由图中可以看出，每个算法的执行时间都随着s值的增大而增大。当s值较小时，算法的执行速度都很快，且算法随着s值的增大，基本保持持平，没有太大差别。但算法相比执行时间会有所增加，因为算法是在算法计算结果的基础上对点组的进一步排序处理，必然增加了执行时间。从图中还可以发现，随着s的增大，算法和的执行时间呈指数型增长，时间增加的原因是：随着s的增大，前m个层上点的个数会急剧增加，导致组点组的个数增加，数据规模变大。但是，基于的改进算法执行速度在很大程度上要优于其他两个算法，最快提高了十倍之多。

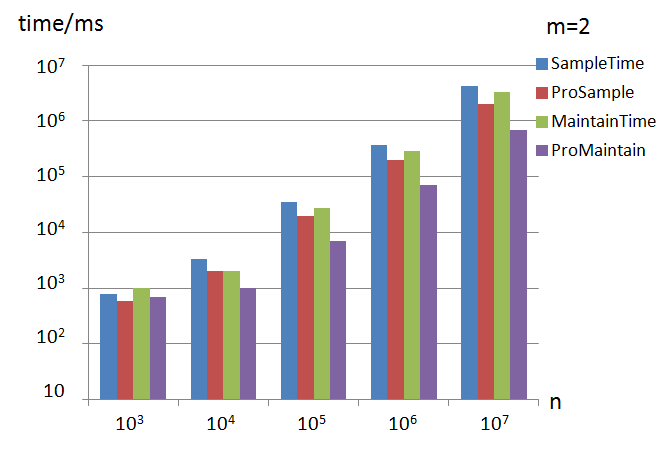


图5-1

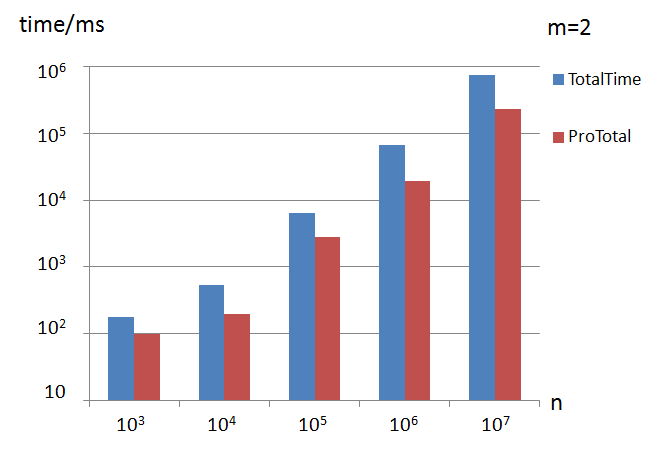


图5-1

在研究数据规模n对算法效率的影响时，保持其他因素不变。图5-1所示为RWMST算法中取样操作随结点的变化情况，图5-1所示为在算法的执行时间方面，如图5-1所示，横坐标表示所求点组s值，纵坐标表示运行时间，时间单位是ms。实验中，采用PointWise[10]算法计算组RWMST点组，由图中可以看出，每个算法的执行时间都随着s值的增大而增大。当s值较小时，算法的执行速度都很快，且算法随着s值的增大，基本保持持平，没有太大差别。但算法相比执行时间会有所增加，因为算法是在算法计算结果的基础上对点组的进一步排序处理，必然增加了执行时间。从图中还可以发现，随着s的增大，算法和的执行时间呈指数型增长，时间增加的原因是：随着s的增大，前m个层上点的个数会急剧增加，导致组点组的个数增加，数据规模变大。但是，基于的改进算法执行速度在很大程度上要优于其他两个算法。

### 5.3.4 内存增长情况比较

如图5-1阐释了四种算法的内存占用轨迹，其中SimpleRW和SA的内存占用情况几乎相同，而RWB与RWBT均多消耗了30%的内存空间，这是由于桶中的节点都是存储在动态数组中的，这些动态数组会比实际需要申请更大的存储空间。

SimpleRW与SA算法需要申请一个大小为n的数组存储节点的度。因此，考虑到性能和效率本文实验仅仅采用了数组的结构存储节点相应的度值。而用到的基于桶的RWB结构采用红黑树进行存储，这样可以保证遍历过程的整体有序，可以加快查询，删除，插入等操作的速度。桶中的节点存储在动态增长的数组中。

## 5.4 本章小结

本章在开始阶段介绍了实验所使用的环境和实验所用到的数据集，然后在Eclipse环境下用Java语言编程实现了SimpleRW算法，RW-bucket算法，RWBT算法，RWBMT算法和RWBMST算法，并在程序自动生成的随机数数据集上进行对比实验。最后客观地对实验的结果进行了详细地分析，从算法的处理时间方面验证了RWBMST算法的高效性和可行性。

# 结 论

经过多年研究，在无标度网络相关的领域的研究已经取得了大量的成果，但是现实应用中图数据规模庞大且复杂程度高，创建符合现实特性的大规模无标度网络一直以来都受到研究者的广泛关注。随着大数据的高速发展，现实场景的数据日益复杂。本文针对现有算法不能适用于实际应用中图规模不断增长的问题，提出了占用内存小，处理效率高的无标度网络图创建算法，取得了以下研究成果：

(1) 针对已有算法时间复杂度高，存在大量无意义中间数据计算的问题，提出了优化的RWBT结构。RWBT结构基于一棵在叶子节点存储具有相同度的节点集合的生成树，减少了节点的重复计算，加速了基于BA模型创建无标度网络的过程中各个操作的速度，并通过将删除操作进行优化，对需要删除的桶采用替换序号标记的策略，减少桶更新步骤所耗费的时间，进一步提升了算法的处理效率。

(2) 为了进一步优化算法的处理效率，将RWBT结构的分支进行扩展提出了RWKT结构，并改进节点的遍历策略，使得整棵树的平均编码长度接近于哈夫曼编码平均长度的最低限度，进而减小了遍历节点过程的用时。考虑到RWKT结构在扩展分支过多之后导致算法性能下降，基于提出的RWKT结构，提出了高效的插入策略，通过对RWKT结构中的兄弟节点之间进行排序处理，提出了优化的RWMST结构，使得RWKT结构具有良好的扩展性，达到了预期的效果。

(3) 通过以上的对比实验，从不同角度对不同算法进行了比较分析，实验结果表明了本文提出的无标度网络创建算法占用内存小，处理时间短的优势。而且对节点遍历过程的优化在时间复杂度上也由最基础算法的O(|V|2)提高到了接近线性的复杂度。同时，对创建更符合现实特性的无标度网络具有良好的扩展性。

综上所述，本文提出的索引结构是一种精确的串行轮盘数据结构，可以在不失精确度的情形下为处理创建大规模网络图结构的可扩展性提供理论依据。目前从实验效果来看虽然达到了预期的研究目的，也取得了一定的成果，但是这种结构也存在一定的局限性，比如不适用于随机图的创建。在未来工作中需要针对特定情形进行更深一步的研究，从而使得更多网络图的创建得到更完善的解决方案。

# 参考文献

1. Aiello W, Chung F, Lu L. A Random Graph Model for Massive Graphs[M]// The Structure and Dynamics of Networks. 2011.
2. Akoglu L, Faloutsos C. RTG: a recursive realistic graph generator using random typing[J]. Data Mining & Knowledge Discovery, 2009, 19(2):194-209.
3. Akoglu L, Mcglohon M, Faloutsos C. RTM: Laws and a Recursive Generator for Weighted Time-Evolving Graphs[C]// Eighth IEEE International Conference on Data Mining. IEEE Computer Society, 2008:701-706.
4. Alam M, Khan M, Marathe M V. Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model[C]// High PERFORMANCE Computing, Networking, Storage and Analysis. IEEE, 2014:91.
5. Arora A, Sachan M, Bhattacharya A. Mining statistically significant connected subgraphs in vertex labeled graphs[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2014:1003-1014.
6. Barabasi A L, Albert R. Albert, R.: Emergence of Scaling in Random Networks. Science 286, 509-512[J]. 1999, 286(5439):509-512.
7. Chakravartula S. Complex networks: Structure and dynamics[J]. Dissertations & Theses - Gradworks, 2014, 424(4-5):175–308.
8. Boguñá M, Pastor-Satorras R. Class of correlated random networks with hidden variables[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2003, 68(3 Pt 2):036112.
9. Boguñá M, Pastor-Satorras R, Vespignani A. Cut-offs and finite size effects in scale-free networks[J]. The European Physical Journal B - Condensed Matter and Complex Systems, 2004, 38(2):205-209.
10. Buesser P, Tomassini M. Supercooperation in evolutionary games on correlated weighted networks[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2012, 85(1 Pt 2):016107.
11. Seshadhri C, Kolda T G, Pinar A. Community structure and scale-free collections of Erd\"os-R\'enyi graphs[J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2011, 85(5 Pt 2):056109.
12. Caldarelli G, Capocci A, De L R P, et al. Scale-free networks from varying vertex intrinsic fitness.[J]. Phys.rev.lett, 2002, 89(25):258702.
13. Campbell C, Shea K, Albert R. Network models. Comment on "Control profiles of complex networks"[J]. Science, 2014, 346(6209):561.
14. Hegeman T, Iosup A, Erling O, et al. Graphalytics: A Big Data Benchmark for Graph-Processing Platforms[C]// Grades. ACM, 2015:7.
15. Carlson J M, Doyle J. Highly optimized tolerance: A mechanism for power laws in designed systems[J]. Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics, 1999, 60(2 Pt A):1412-1427.
16. Chakrabarti D, Zhan Y, Faloutsos C. R-MAT: A recursive model for graph mining[C]// Siam International Conference on Data Mining, Lake Buena Vista, Florida, Usa, April. DBLP, 2004.
17. Chen H H, Ciou Y B, Lin S D. Information propagation game:a tool to acquire humanplaying data for multiplayer influence maximization on social networks[C]// Kdd '12 Proceedings of the, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2012:1524-1527.
18. Chung F, Lu L. The average distances in random graphs with given expected degrees.[J]. Proceedings of the National Academy of Sciences of the United States of America, 2002, 99(25):15879-82.
19. Poltyrev G S. Book Review. CsiszárI. and KörnerJ. “Information Theory. Coding Theorems for Discrete Memoryless Systems”[J]. Probl.peredachi Inf, 1982:108–111.
20. Du N, Wang H, Faloutsos C. Analysis of Large Multi-modal Social Networks: Patterns and a Generator[J]. 2010.
21. Tuza N A Z. on random graphs[J]. Random Structures & Algorithms, 2008, 6(2-3):261–268.
22. Faller N. An adaptive system for data compression[J]. Proc.asilomar Conf.circuits Systems & Computers, 1973.
23. Fazekas I, Porvázsnyik B. Scale-Free Property for Degrees and Weights in a Preferential Attachment Random Graph Model[J]. Journal of Probability and Statistics,2013,(2013-9-22), 2013, 2013.
24. Cancho R F I, Solé R V. Optimization in Complex Networks[J]. Lecture Notes in Physics, 2001, 625:0111222--2001.
25. Author-Kullback S. Information Theory and Reliable Communication (R. G. Gallager)[M]. Society for Industrial and Applied Mathematics, 2006.
26. Gilbert E N. Random Graphs[J]. Annals of Mathematical Statistics, 1959, 30(4):1141-1144.
27. Goldenberg A, Zheng A X, Fienberg S E, et al. A Survey of Statistical Network Models[J]. Foundations & Trends® in Machine Learning, 2010, 2(2):129-233.
28. Klemm K, Eguíluz V M. Growing scale-free networks with small-world behavior.[J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2001, 65(2):057102.
29. Knuth D E. Dynamic huffman coding[J]. Journal of Algorithms, 1985, 6(2):163-180.
30. Kolda T G, Pinar A, Plantenga T, et al. A Scalable Generative Graph Model with Community Structure[J]. Siam Journal on Scientific Computing, 2013, 36(5).
31. Machado C. Assessing Systemic Importance With a Fuzzy Logic Inference System[M]. John Wiley and Sons Ltd. 2016.
32. Leskovec J. Dynamics of large networks[J]. 2008.
33. Leskovec J, Chakrabarti D, Kleinberg J, et al. Kronecker Graphs: An Approach to Modeling Networks[J]. Journal of Machine Learning Research, 2010, 11(3):985-1042.
34. Li J, Wang H, Khan S U, et al. A Fully Distributed Scheme for Discovery of Semantic Relationships[J]. IEEE Transactions on Services Computing, 2013, 6(4):457-469.
35. Lipowski A, Lipowska D. Roulette-wheel selection via stochastic acceptance[J]. Physica A Statistical Mechanics & Its Applications, 2011, 391(6):2193-2196.
36. Lo Y C, Lai H C, Li C T, et al. Mining and generating large-scaled social networks via MapReduce[J]. Social Network Analysis & Mining, 2013, 3(4):1449-1469.
37. ANDREW LUMSDAINE, DOUGLAS GREGOR, BRUCE HENDRICKSON, et al. CHALLENGES IN PARALLEL GRAPH PROCESSING[J]. Parallel Processing Letters, 2008, 17(01):5-20.
38. Mcglohon M, Akoglu L, Faloutsos C. Weighted graphs and disconnected components:patterns and a generator[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2008:524-532.
39. Newman M E J. Modularity and community structure in networks[J]. Proc Natl Acad Sci U S A, 2006, 103(23):8577-82.
40. Nobari S, Lu X, Karras P. Fast random graph generation[C]// EDBT 2011, International Conference on Extending Database Technology, Uppsala, Sweden, March 21-24, 2011, Proceedings. DBLP, 2011:331-342.
41. Nobari S, Tauheed F, Heinis T, et al. TOUCH:in-memory spatial join by hierarchical data-oriented partitioning[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2013:701-712.
42. Pham M D, Boncz P, Erling O. S3G2: A Scalable Structure-Correlated Social Graph Generator[M]// Selected Topics in Performance Evaluation and Benchmarking. Springer Berlin Heidelberg, 2013:156-172.
43. JULIA PONCELA, YAMIR MORENO. COOPERATION IN THE PRISONER'S DILEMMA GAME IN RANDOM SCALE-FREE GRAPHS[J]. International Journal of Bifurcation & Chaos, 2010, 20(03):849-857.
44. Solé R V, Ferrer-Cancho R, Montoya J M, et al. Selection, tinkering, and emergence in complex networks[J]. Complexity, 2010, 8(1):20-33.
45. Soramäki K, Cook S. Algorithm for identifying systemically important banks in payment systems[J]. Economics Discussion Papers, 2012, 7(10):1-27.
46. Looz M V, Staudt C L, Meyerhenke H, et al. Fast generation of complex networks with underlying hyperbolic geometry[J]. Plos One, 2015, 10(7).
47. Wang H, Zheng K, Zhou X, et al. SharkDB:An In-Memory Storage System for Massive Trajectory Data[J]. 2015:1099-1104.
48. FENG, Yuan-Yuan, Liang, et al. Synchronization of Phase Oscillators in Networks with Certain Fre
49. Aiello W, Chung F, Lu L. A Random Graph Model for Massive Graphs[M]// The Structure and Dynamics of Networks. 2011.
50. Akoglu L, Faloutsos C. RTG: a recursive realistic graph generator using random typing[J]. Data Mining & Knowledge Discovery, 2009, 19(2):194-209.
51. Akoglu L, Mcglohon M, Faloutsos C. RTM: Laws and a Recursive Generator for Weighted Time-Evolving Graphs[C]// Eighth IEEE International Conference on Data Mining. IEEE Computer Society, 2008:701-706.
52. Alam M, Khan M, Marathe M V. Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model[C]// High PERFORMANCE Computing, Networking, Storage and Analysis. IEEE, 2014:91.
53. Arora A, Sachan M, Bhattacharya A. Mining statistically significant connected subgraphs in vertex labeled graphs[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2014:1003-1014.
54. Barabasi A L, Albert R. Albert, R.: Emergence of Scaling in Random Networks. Science 286, 509-512[J]. 1999, 286(5439):509-512.
55. Chakravartula S. Complex networks: Structure and dynamics[J]. Dissertations & Theses - Gradworks, 2014, 424(4-5):175–308.
56. Boguñá M, Pastor-Satorras R. Class of correlated random networks with hidden variables[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2003, 68(3 Pt 2):036112.
57. Boguñá M, Pastor-Satorras R, Vespignani A. Cut-offs and finite size effects in scale-free networks[J]. The European Physical Journal B - Condensed Matter and Complex Systems, 2004, 38(2):205-209.
58. Buesser P, Tomassini M. Supercooperation in evolutionary games on correlated weighted networks[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2012, 85(1 Pt 2):016107.
59. Seshadhri C, Kolda T G, Pinar A. Community structure and scale-free collections of Erd\"os-R\'enyi graphs[J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2011, 85(5 Pt 2):056109.
60. Caldarelli G, Capocci A, De L R P, et al. Scale-free networks from varying vertex intrinsic fitness.[J]. Phys.rev.lett, 2002, 89(25):258702.
61. Campbell C, Shea K, Albert R. Network models. Comment on "Control profiles of complex networks"[J]. Science, 2014, 346(6209):561.
62. Hegeman T, Iosup A, Erling O, et al. Graphalytics: A Big Data Benchmark for Graph-Processing Platforms[C]// Grades. ACM, 2015:7.
63. Carlson J M, Doyle J. Highly optimized tolerance: A mechanism for power laws in designed systems[J]. Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics, 1999, 60(2 Pt A):1412-1427.
64. Chakrabarti D, Zhan Y, Faloutsos C. R-MAT: A recursive model for graph mining[C]// Siam International Conference on Data Mining, Lake Buena Vista, Florida, Usa, April. DBLP, 2004.
65. Chen H H, Ciou Y B, Lin S D. Information propagation game:a tool to acquire humanplaying data for multiplayer influence maximization on social networks[C]// Kdd '12 Proceedings of the, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2012:1524-1527.
66. Chung F, Lu L. The average distances in random graphs with given expected degrees.[J]. Proceedings of the National Academy of Sciences of the United States of America, 2002, 99(25):15879-82.
67. Poltyrev G S. Book Review. CsiszárI. and KörnerJ. “Information Theory. Coding Theorems for Discrete Memoryless Systems”[J]. Probl.peredachi Inf, 1982:108–111.
68. Du N, Wang H, Faloutsos C. Analysis of Large Multi-modal Social Networks: Patterns and a Generator[J]. 2010.
69. Tuza N A Z. on random graphs[J]. Random Structures & Algorithms, 2008, 6(2-3):261–268.
70. Faller N. An adaptive system for data compression[J]. Proc.asilomar Conf.circuits Systems & Computers, 1973.
71. Fazekas I, Porvázsnyik B. Scale-Free Property for Degrees and Weights in a Preferential Attachment Random Graph Model[J]. Journal of Probability and Statistics,2013,(2013-9-22), 2013, 2013.
72. Cancho R F I, Solé R V. Optimization in Complex Networks[J]. Lecture Notes in Physics, 2001, 625:0111222--2001.
73. Author-Kullback S. Information Theory and Reliable Communication (R. G. Gallager)[M]. Society for Industrial and Applied Mathematics, 2006.
74. Gilbert E N. Random Graphs[J]. Annals of Mathematical Statistics, 1959, 30(4):1141-1144.
75. Goldenberg A, Zheng A X, Fienberg S E, et al. A Survey of Statistical Network Models[J]. Foundations & Trends® in Machine Learning, 2010, 2(2):129-233.
76. Klemm K, Eguíluz V M. Growing scale-free networks with small-world behavior.[J]. Phys Rev E Stat Nonlin Soft Matter Phys, 2001, 65(2):057102.
77. Knuth D E. Dynamic huffman coding[J]. Journal of Algorithms, 1985, 6(2):163-180.
78. Kolda T G, Pinar A, Plantenga T, et al. A Scalable Generative Graph Model with Community Structure[J]. Siam Journal on Scientific Computing, 2013, 36(5).
79. Machado C. Assessing Systemic Importance With a Fuzzy Logic Inference System[M]. John Wiley and Sons Ltd. 2016.
80. Leskovec J. Dynamics of large networks[J]. 2008.
81. Leskovec J, Chakrabarti D, Kleinberg J, et al. Kronecker Graphs: An Approach to Modeling Networks[J]. Journal of Machine Learning Research, 2010, 11(3):985-1042.
82. Li J, Wang H, Khan S U, et al. A Fully Distributed Scheme for Discovery of Semantic Relationships[J]. IEEE Transactions on Services Computing, 2013, 6(4):457-469.
83. Lipowski A, Lipowska D. Roulette-wheel selection via stochastic acceptance[J]. Physica A Statistical Mechanics & Its Applications, 2011, 391(6):2193-2196.
84. Lo Y C, Lai H C, Li C T, et al. Mining and generating large-scaled social networks via MapReduce[J]. Social Network Analysis & Mining, 2013, 3(4):1449-1469.
85. ANDREW LUMSDAINE, DOUGLAS GREGOR, BRUCE HENDRICKSON, et al. CHALLENGES IN PARALLEL GRAPH PROCESSING[J]. Parallel Processing Letters, 2008, 17(01):5-20.
86. Mcglohon M, Akoglu L, Faloutsos C. Weighted graphs and disconnected components:patterns and a generator[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2008:524-532.
87. Newman M E J. Modularity and community structure in networks[J]. Proc Natl Acad Sci U S A, 2006, 103(23):8577-82.
88. Nobari S, Lu X, Karras P. Fast random graph generation[C]// EDBT 2011, International Conference on Extending Database Technology, Uppsala, Sweden, March 21-24, 2011, Proceedings. DBLP, 2011:331-342.
89. Nobari S, Tauheed F, Heinis T, et al. TOUCH:in-memory spatial join by hierarchical data-oriented partitioning[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2013:701-712.
90. Pham M D, Boncz P, Erling O. S3G2: A Scalable Structure-Correlated Social Graph Generator[M]// Selected Topics in Performance Evaluation and Benchmarking. Springer Berlin Heidelberg, 2013:156-172.
91. JULIA PONCELA, YAMIR MORENO. COOPERATION IN THE PRISONER'S DILEMMA GAME IN RANDOM SCALE-FREE GRAPHS[J]. International Journal of Bifurcation & Chaos, 2010, 20(03):849-857.
92. Solé R V, Ferrer-Cancho R, Montoya J M, et al. Selection, tinkering, and emergence in complex networks[J]. Complexity, 2010, 8(1):20-33.
93. Soramäki K, Cook S. Algorithm for identifying systemically important banks in payment systems[J]. Economics Discussion Papers, 2012, 7(10):1-27.
94. Looz M V, Staudt C L, Meyerhenke H, et al. Fast generation of complex networks with underlying hyperbolic geometry[J]. Plos One, 2015, 10(7).
95. Wang H, Zheng K, Zhou X, et al. SharkDB:An In-Memory Storage System for Massive Trajectory Data[J]. 2015:1099-1104.
96. FENG, Yuan-Yuan, Liang, et al. Synchronization of Phase Oscillators in Networks with Certain Frequency Sequence[J]. Communications in Theoretical Physics, 2014, 61(3):329-3

# 攻读硕士学位期间承担的科研任务与主要成果

1. 参与的科研项目
2. 陈子阳,周军锋,王倩倩.面向XML数据的关键字查询算法辅助生成技术研究, 国家自然科学基金资助项目. 课题编号: 61272124.

# 致 谢

时光飞逝，岁月匆匆，不知不觉间，在燕大三年研究生生活马上就要结束了，回忆起在燕大三年的学习和生活，感慨万千，受益终生。有艰辛与奋斗，也有甜美与欢笑，也有些许遗憾，但更多的是感激之情。在此论文完成之际，非常感谢我的导师陈子阳教授。陈老师严肃认真的治学态度，精益求精的工作精神，诲人不倦的高尚品格时刻激励着我，对我影响深远。无论从平时课题的讨论学习还是从论文的选题、搜集资料、写稿到反复修改，陈老师无不悉心指导，帮助我开拓研究思路，无微不至的关怀与热忱鼓励是论文得以顺利完成的重要保证。在陈老师的帮助和鼓励下，我无论是在阅读文献能力，还是在编程能力上都有了较大的提升。同时还要向课题组的周军锋教授表示衷心的感谢。两位老师平易近人、治学严谨，在学习中遇到困难的时候，他们孜孜不倦的教导和不拘一格的思路给予我无尽的启迪，也在百忙中抽出时间对我的毕业论文给出了宝贵的指导意见。在此向陈老师和周老师表示深切的谢意和祝福，感谢他们让我度过了一段充实而难忘的研究生生活。

同时，我还要感谢我们实验室的兄弟姐妹们，三年来，我们朝夕相处，共同进步，一起营造了一个宽松而愉快的学习氛围，是你们陪我一起走过学习和生活中的中的点点滴滴。感谢你们在整个研究生的学习阶段所给予的热心帮助和热忱鼓励，并在我的课题研究中给出了很多宝贵的意见。

另外，还要特别感谢信息科学与工程学院的各位老师们，他们所提供的良好的学习环境和科研氛围，才使我的论文能顺利完成。

最后，感谢每一位评审老师百忙中审阅我的论文，并给出指导和点评。再次向在学习、工作和生活中给予过自己关心、支持与鼓励的所有老师、同学、朋友们表示最诚挚的谢意！