分类号 0175.2 密级 限制

UDC 004.72

学 位 论 文

数据中心网络模型研究

（题名和副题名）

高兴坤

（作者姓名）

指导教师姓名、职务、职称、学位、单位名称及地址 唐杰 副教授

南京大学计算机科学与技术系 南京市栖霞区仙林大道 163 号 210023

申请学位级别 硕士 专业名称 计算机科学与技术

论文提交日期 2018 年 5 月 10 日 论文答辩日期 2018 年 6 月 1 日

学位授予单位和日期

答辩委员会主席： 张三丰 教授

评阅人： 阳顶天 教授

张无忌 副教授

黄裳 教授

郭靖 研究员

南京大学

研 究 生 毕 业 论 文

（申请硕士学位）

论　文　题　目　(论文长标题第一行)

　　(论文长标题第二行)

作　者　姓　名　高兴坤

学科、专业方向　计算机科学与技术

研　究　方　向　大数据与分布式计算

指　导　教　师　唐杰　副教授

2018 年 4 月 25 日

学　　号：MG1533012

论文答辩日期: 2018 年 6 月 1 日

指 导 教 师：　　(签字)

A Research on Network Infrastructures for Data Centers

by

GAO Xing-Kun

Supervised by

Associate Professor Tang Jie

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
Master
in
Computer Science and Techonology

Department of Computer Science and Technology
Nanjing University

May 1, 2018

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：数据中心网络模型研究

计算机科学与技术 专业 2015 级硕士生姓名：高兴坤

指导教师（姓名、职称）：唐杰 副教授

摘 要

复杂网络的研究可上溯到 20 世纪 60 年代对 ER 网络的研究。90 年后代随着 Internet 的发展，以及对人类社会、通信网络、生物网络、社交网络等各领域研究的深入，发现了小世界网络和无尺度现象等普适现象与方法。对复杂网络的定性定量的科学理解和分析，已成为如今网络时代科学研究的一个重点课题。

在此背景下，由于云计算时代的到来，本文针对面向云计算的数据中心网络基础设施设计中的若干问题，进行了几方面的研究。……

关键词： 小世界理论；网络模型；数据中心

南京大学研究生毕业论文英文摘要首页用纸

THESIS: A Research on Network Infrastructures for Data Centers

SPECIALIZATION: Computer Science and Techonology

POSTGRADUATE: GAO Xing-Kun

MENTOR: Associate Professor Tang Jie

Abstract

keywords: Small World, Network Model, Data Center

前　言

作为刻画事物之间连接关系的重要数据结构，图在现代社会中很多领域有着广泛的运用。例如，在互联网、社交网络、物联网、电子商务中，相同或不同实体之间的关系都可以使用图来刻画。进一步地，如何从图中挖掘出更深层次的信息也成了网络时代科学研究的一个重要课题。而随着技术的发展，现实问题中图的规模在剧烈的增长，传统的单机算法已经无法在有效时间内完成计算需求。

在此背景下，由于大数据时代的到来，本文针对图中节点相似性 SimRank 的分布式计算问题，进行了几方面的研究。本文的创造性研究成果主要如下几方面:

1. 针对单源点 SimRank 相似度的计算，提出一种基于随机游走的分布式计算方法。整个算法主要有两个步骤：先生成生成随机游走，再对游走进行匹配计算其对应概率，最后汇总得出相似性。设计了一些优化方法来加速算法，包括减少随机游走的数量，使用更紧凑的数据结构来压缩中间数据，以及通过动态规划的技巧加速随机游走。本文还对算法的复杂度给出了详尽的分析。
2. 提出一种分布式图分割算法，该算法能够对对图进行尽量均匀的分割，同时最小化分割之间边的数量（或权重）。实验表明：该分割方法具有较高的效率，同时拥有较高的分割质量，稠密的子图往往可以处在统一分割中。
3. 基于上诉的图分割方法，提出一种计算所有节点之间相似度的分布式计算方法。该方法使用分而治之的思想，先计算局部稠密子图的相似性，然后把原先图的每个划分当做新的节点形成一个 Coarsen Graph，重新计算这个图的节点相似性，然后使用这两个局部相似性估算出原图中所有节点的相似性。
4. 对上诉算法，给出在流行的通用计算平台 Spark 上的分布式实现，并使用真实的数据集在分布式集群验证算法的精度，效率，以及可扩展性。

高兴坤

2018 年夏于南京大学

目　　次

插图清单

附表清单

绪论

研究背景与意义

图作为一种刻画实体之间关系的重要数据结构，在很多领域中有很多的应用。例如，电子商务中商品与用户之间的交易行为，科学论文领域中各种论文之间的相互参考应用，Web 中个网页之间的链接，以及社交网络中用户之间的好友等等行为，本质上都可以通过基于图来建模。因此，如何从图中挖掘出深层次的信息也成了产业界以及学术界的一个重要课题。

在信息检索，社交网络分析，推荐系统等多个领域，进行相似性度量，即比较不同对象之间的相似度，是一个重要的课题。SimRank 算法是近年来比较流行的一种度量图中节点相似性的模型，与传统方法不同的是它考虑了整个图的拓扑结构，其基本思想与著名的网页排序算法 PageRank 有相似之处：如果两个节点的邻居节点非常相似，那么这两个节点也相似。SimRank 模型可以通过随机游走理论解释，研究表明，它能更准确地衡量相似性。

然而，随着互联网技术的发展，现在已经进入大数据时代。社交网络用户的爆发性增长，电子商务的日益普及，使得现实生活中图数据的规模越来越大。而传统的 SimRank 算法具有较高的时间复杂度和空间复杂度，已经无法适应如今大规模图的分析要求。传统的 SimRank 计算，包括三个问题：1、单节点对的相似度计算 2、单源点的相似度计算 3、全节点对的相似度计算。这三个问题的复杂度依次增加，由计算图中某一堆节点的相似度，到图中某节点到其他所有节点的相似度，再到图中所有点对的相似度。

另一方面，诸如 MapReduce、Spark 等通用式的分布式计算平台充分使用了分布式集群的威力，用户可以快速部署，然后使用其提供的编程范式专注与自己的计算任务，而无需关系分布式平台底层的复杂网络通信、资源分配、资源调度问题。在这些平台上设计出高效的分布式算法，具有现实意义。目前现有的计算方法主要是基于矩阵计算，而对于节点数超过百万的大图，其邻接矩阵的元素规模超过万亿，面对如此庞大的输入规模，普通的矩阵计算方法根本不可行。

针对上述问题，本文提出基于 Spark 平台的分布式算法，以解决大规模图的 SimRank 计算问题。本文重点研究第二类与第三类计算问题，单源点相似性计算以及全节点对的相似性计算，设计在准确率、计算效率以及可扩展性的分布式算法。

相关工作

为了方便叙述，我们用 n 表示输入图的节点个数，用 m 表示图的边的个数，用 d 表示图中节点的平均度数。对于迭代算法，使用 k 表示算法需要迭代的次数。

基于矩阵乘法的计算方法。Jeh 和 Widom 提出了第一个基于矩阵乘法的 SimRank 迭代计算方法，改算法计算全点对的相似性，其复杂度为 $O(kn^2d^2)$.[9] 随后通过剪纸、局部访问等技术在矩阵乘法层面将算法的时间复杂度提升到 $O(kn^2d)$。【10】使用了快速矩阵乘法来加速计算过程。文献【12】进一步将算法的时间复杂度提高到 $O(kn^2d')$, 其中 $d' < d$。文献【13】提出了一种基于 Keonecker 乘积和矩阵奇异值分解（SVD）的非迭代算法，该算法首先在 $O(r^4N^2)$ 时间内计算一些出辅助矩阵，其中 r 是输入图的领接矩阵的秩，然后再在 $O(r^4N^2)$ 时间内获取给定源点与其他所有节点的相似性。文献【14】使用了 GPU 来加速矩阵的计算速度。对于单点对的 SimRank 计算，文献【11】给出了一个时间复杂度为 $O(kd^2 \cdot \min\{n^2, d^k\}$ 的算法。文献【15】进一步地通过使用概率方法将时间复杂度提高到 $O(km^2 - m)$。

基于随机游走的计算方法。图中节点 v,u 的相似性，可以解释为：以节点 u 和节点 v 分别为初始点，同步移动的随机游走第一次在某个节点相遇的概率的期望值。文献【16】给出了第一个基于随机游走的算法，该算法首先建立一个大小为 $O(nN)$ 的 N 个随机游走的“指纹”索引，然后基于索引查询单点对的相似性。文献【17】提出一种在随机游走中使用采样技巧计算单点对相似性，该方法在允许一定精度损失的情况下提高了计算效率。文献【18】研究了图中 top-k 个最相似点对的查询问题，其中 k 往往是一个非常小的值。该方法把单点对相似查询问题转变为规模为 $G \times G$ 的乘积图上的查询问题。

分布式计算方法。文献【19】提出了一种基于增量变化的 $Delta - SimRank$ 算法。该算法发现，SimRank 可以改写为一种迭代的增量计算方式，即迭代过程中，不直接改变点对之间的相似性而是计算相对于上一轮迭代的增量。该方

法充分利用了迭代计算过程中很多点对间相似性增量为 0 的事实，减小了计算过程中的数据传输量。文献【20】提出了一种基于 Spark 平台的分布式计算方法，该方法打破了计算点对相似性之间的递归依赖关系：首先线下计算一个不变矩阵 D，然后线上根据 D 使用蒙特卡洛方法快速给出查询。然而，D 的计算被当做一个解线性方程组的过程，这在分布式环境下非常低效，因此虽然算法线上查询的效率很高，但线下预处理的时间开销非常巨大。

1. 针对单源点 SimRank 相似度的计算，提出一种基于随机游走的分布式计算方法。整个算法主要有两个步骤：先生成生成随机游走，再对游走进行匹配计算其对应概率，最后汇总得出相似性。设计了一些优化方法来加速算法，包括减少随机游走的数量，使用更紧凑的数据结构来压缩中间数据，以及通过动态规划的技巧加速随机游走。本文还对算法的复杂度给出了详尽的分析。
2. 提出一种分布式图分割算法，该算法能够对对图进行尽量均匀的分割，同时最小化分割之间边的数量（或权重）。实验表明：该分割方法具有较高的效率，同时拥有较高的分割质量，稠密的子图往往可以处在统一分割中。
3. 基于上诉的图分割方法，提出一种计算所有节点之间相似度的分布式计算方法。该方法使用分而治之的思想，先计算局部稠密子图的相似性，然后把原先图的每个划分当做新的节点形成一个 Coarsen Graph，重新计算这个图的节点相似性，然后使用这两个局部相似性估算出原图中所有节点的相似性。
4. 对上诉算法，给出在流行的通用计算平台 Spark 上的分布式实现，并使用真实的数据集在分布式集群验证算法的精度，效率，以及可扩展性。

现有解决方法的不足

本文工作的中心观点与思想

分析已有的计算方法我们可以发现，现有的主流方法在计算全点对相似性时，往往基于矩阵计算；计算单源点相似性时，大多采用随机游走的思想。然而大多数方法都是传统的单机计算方法，对于大规模的输入图无能无力。另一方面，现有的分布式算法基本也采用了矩阵计算的思路，而对于一个节点数目为百万级别的图，其矩阵的元素基本在万亿级别以上。面对如此大的计算规模，即便分布式处理也非常吃力。

因此，

表 1-1: 测试表格

文档域类型	Java 类型	宽度 (字节)	说明
BOOLEAN	boolean	1	
CHAR	char	2	UTF-16 字符
BYTE	byte	1	有符号 8 位整数
SHORT	short	2	有符号 16 位整数
INT	int	4	有符号 32 位整数
LONG	long	8	有符号 64 位整数
STRING	String	字符串长度	以 UTF-8 编码存储
DATE	java.util.Date	8	距离 GMT 时间 1970 年 1 月 1 日 0 点 0 分 0 秒的毫秒数
BYTE_ARRAY	byte[]	数组长度	用于存储二进制值
BIG_INTEGER	java.math.BigInteger	和具体值有关	任意精度的长整数
BIG_DECIMAL	java.math.BigDecimal	和具体值有关	任意精度的十进制实数

技术背景

Spark 背景

Aparch Spark 是由美国 UC Berkeley AMP Lab 开发的通用分布式计算引擎，目前它已经成为 Apache Software Foundation 下以及同类大数据开源项目中最受关注的项目之一。Spark 遵循类似于 MapReduce 的编程范式，但是在执行效率上，Spark 改进了 Hadoop[14] 批处理框架在迭代计算与交互式处理方面的不足，引入了 RDD（弹性分布式数据集）的概念，允许将计算的中间结果保存在内存之中而无需写入磁盘，大大改善了迭代计算的效率，应用程序的性能得到数十倍甚至百倍的提升；在用户易用性上，Spark 支持 Java, Scala, Python 等主流编程语言，提供了更加丰富的编程 API；在通用性上，Spark 针对不同的开发需求提供了更高阶的库，包括 SQL 查询，流式计算，机器学习算法以及图计算等等，开发者可以根据自己的需要，单独或组合使用这些库来处理复杂的数据分析任务。目前，Spark 已经成长为包含 Spark SQL，Spark

Streaming, MLlib, GraphX 等多个子项目的完善的生态系统。使用 Spark，用户只需调用编程接口来实现自己的数据计算任务，而无需关心数据的具体分布、并行调度策略、集群节点之间的数据传送与错误恢复等复杂的底层细节。凭借其高效、用户友好、通用的优点，Spark 在学术界和工业界得到了广泛的实际应用。

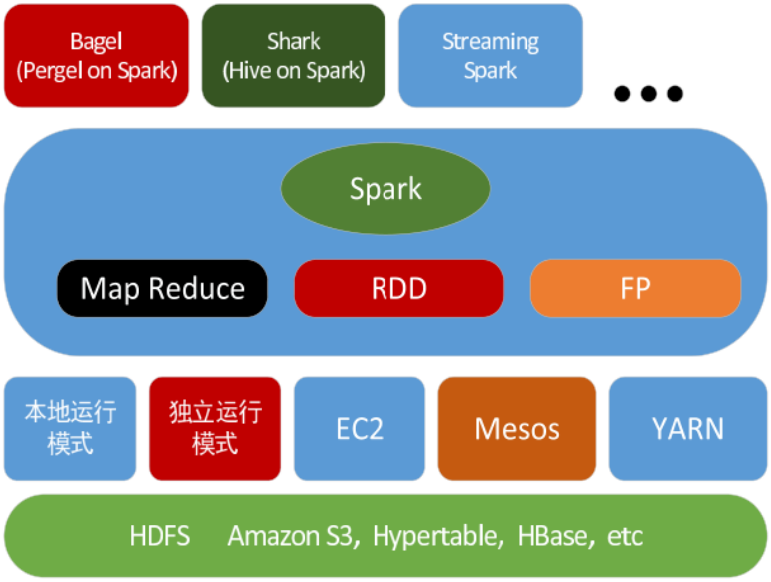


图 1-1: Spark 系统架构

Spark 系统构架

一个典型的 Spark 集群通常包括以下几个组件：1)Driver Program, 每一个 Spark 应用程序都包含一个 driver program，它执行用户的 main 函数，并负责在集群调度所有的并行化操作。当应用程序刚从客户端提交上来的时候，Driver Program 需要连接到负责分配整个集群上各种资源的 Cluster Manager，并向其请求集群上能够使用的 Executor。然后，Driver Program 会把应用程序的 Jar 包发送给这些 Executor，计算过程中会将计算 Task 分配给这些 Executor。2)Cluster Manager, 它是向集群申请各种资源的一个额外的服务。Spark 支持 Mesos, YARN 等等。3)Executor, 它是运行在集群中每一个 Worker Node 上的一个进程，负责本节点上数据的读写以及各种计算。

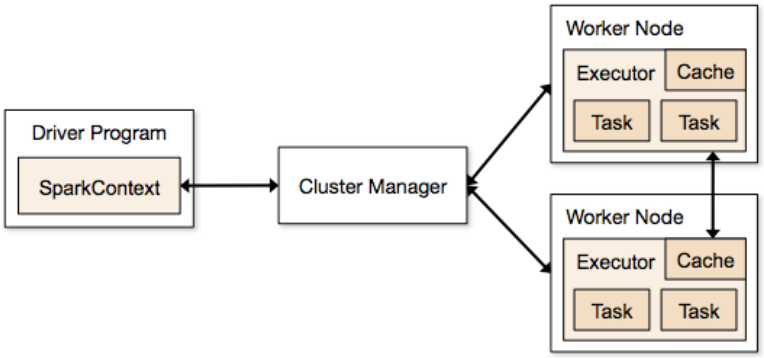


图 1-2: Spark 系统架构

Spark 数据模型

Spark 采用了一种被称为弹性分布式数据集（RDDs, Resilient Distributed Datasets）的分布式数据抽象，它支持工作集的重用，同时拥有非循环数据流模型（acyclic data flow model）的优点，即数据的负载均衡、位置感知和容错机制。同时 RDD 是一种受到限制的抽象，这体现在它是只读的，并且只能由已有的分布式数据集或其他 RDD 执行转换操作来创建。通过递归式地保存自己与其 ParentRDD 之间的转换记录 (lineage)，当发生数据分区丢失时 RDD 可以利用这些记录重新进行计算，快速地恢复数据。RDD 的缺点是它只能提供粒度较粗的转换 (coarse-grained transformation) 并且它是只读的，这对于一些需要频繁更新数据集的一部分的应用，可能会带来性能上的不足。矩阵计算就是一个需要频繁更新矩阵本身的操作，因此实现算法时要格外注意如何解决这个问题。RDD 可以调用的操作包括转换 (transformation) 和动作 (action) 两类。每一个转换操作都会基于调用它的一个或多个 parentRdd 产生一个新的 RDD，所有的转换操作都是懒惰 (lazy evaluation) 的，这意味着用户在一个 RDD 上调用一个转换操作并不会立即执行，Spark 只是把它当作 matadata 记住了这些操作。只有当该 RDD 执行动作操作时，Spark 才会利用这些 metatada 自后往前地追踪到当前 RDD 最古老的 parentRDD，然后从前往后顺序计算这些 RDD 之间的转换。Spark 提供了多种多样的转换，例如 map, mapPartitions, union, join, filter 等等。动作 (action) 指的是需要向 driver program 返回计算结果或需要将数据导出至存储系统的一类操作。每一个动作都会生成一个 Job，对数据进行计算并将计算结果返回给 driver。Spark 同样提供了多种多样的动作, 例如 reduce，collect, count，saveAsTextFile 等等。

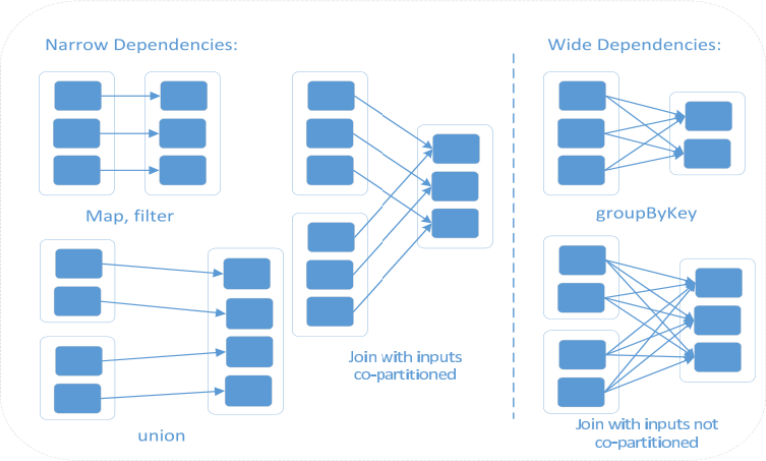


图 1-3: Spark 系统架构

Spark 常用编程接口

Spark 提供的编程接口，主要包括两类：转化和动作。其中，典型的转化包括以下几类：

1. `map` 一个 RDD 可以通过 map 将 RDD 中的每一个元素通过用户指定的函数转化为另一种类型的元素，也就是将原来的 RDD 转变为指定类型的 RDD。map 函数接受的参数是一个函数算子。
2. `mapValues` 与 map 类似，mapValues 用于类型为 (key, value) 的 PairRDD。不同的是 mapValues 不改变元素的 key 而只改变元素的 value，因此新的 RDD 会保留原 RDD 的 partitioner。
3. `mapPartitions` 与 map 类似，它的第一个参数是一个函数，但 mapPartitions 是对原 RDD 的每一个 partition 而不是每一个 element 实施这个函数；它的第二个参数是一个 Boolen 类型，当原 RDD 的元素类型为 (key, value) 键值对，并且 mapPartitions 操作不会改变元素的 key 时，将第二个参数设为 true 告诉 Spark 不要改变 RDD 的 partitioner。
4. `filter` 一个 RDD 可以通过 filter 对 RDD 中的所有元素按照一定条件进行过滤, 过滤后的元素组成一个新的 RDD。因此，filter 函数接受的参数是一个返回类型为 Boolean 的函数，这个函数指明一个元素能不能通过筛选。
5. `join` 两个元素类型为 (key, value) 的 PairRDD 可以通过 join 操作将双方 key 相同的元素合并，返回一个新的 RDD。
6. `union` 两个 RDD 可以通过 union 操作合并为一个 RDD。

7. `reduceByKey` 一个 PairRDD 调用这个操作可以将原 RDD 中 key 相同的若干元素规约为一个元素，返回一个新的 RDD。它的第一个参数是一个函数，这个函数相当于元素之间的运算符；第二个可选的参数可以指定新 RDD 的 partition 个数。类似于 Hadoop 中的 combiner，spark 在 shuffle 之前会首先对每个计算节点本地的元素先进行合并，从而减小 shuffle 的开销。

典型的动作包括以下几类：

1. `reduce` 它的参数是一个满足交换律、结合律的二元运算符，原 RDD 的所有元素经过这个运算符计算，返回一个最终值。
2. `collect` 它将 RDD 中所有元素从集群上以一个数组的形式返回到 Driver Program。collect 是一个开销非常大的操作，只适合在小数据集上调用。

还有一种比较特殊的广播操作。通常来说，当集群中的某个节点计算过程中发现需要使用用户自定义的变量时，会向 driver program 请求将该变量传送到该节点，每一个节点单独地对本地的变量副本进行计算，计算过程中对该副本做出的改动并不会反映到 driver program。Spark 提供了一种广播变量（broadcast variable），它允许开发者主动地将可能在在节点用到的变量广播到集群中的每个节点上，而不用在程序执行过程中当特定节点发现要使用这个变量再从 driver program 发送到这个节点上。Spark 在传送广播变量时使用了一些高效的算法，减少了广播过程中的通信开销。文献 [15] 表明，在进行迭代计算时，广播变量能极大地提高整体性能。

GraphX 背景

GraphX[todo] 是 Spark 生态圈中的一个核心组件，主要用于图并行计算。它提供对图计算和图挖掘简洁易用的而丰富的接口，极大的方便了对分布式图处理的需求 GraphX 通过引入一个新的图抽象来拓展 Spark 中的 RDD: Resilient Distributed Property Graph，一种点和边都带属性的有向多重图这里的属性 (property) 指的是用户自己定义的描述边或定点某些性质的对象。一个典型的属性图中包含了描述节点信息的 VertexRDD 和描述边信息的 EdgeRDD。对属性图的所有操作，最终都会转换成其关联的 VertexRDD 和的 EdgeRDD 上的相关操作。这样对一个图计算任务，最终在逻辑上，等价于一系列 RDD 的转换过程。因此，Graph 最终具备了 RDD 的 3 个关键特性：

Immutable、Distributed 和 Fault-Tolerant，其中最关键的是 Immutable（不变性）。逻辑上，所有图的转换和操作都产生了一个新图；物理上，GraphX 会有一定程度的不变顶点和边的复用优化，对用户透明。

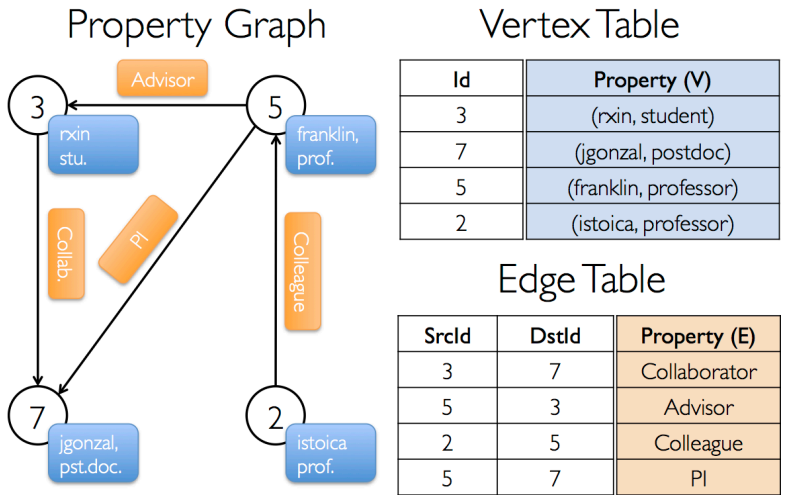


图 1-4: 一个简单的属性图。左图为图的拓扑信息及其关联的属性；右图为该属性图底层存储的 RDD

GraphX 编程范式

论文结构

SimRank 原理及技术背景

SimRank

为了方便描述，首先给出一些通用的符号定义。本文使用关系 (V,E) 表示一个图，其中 V 图的节点集合，$E \subseteq V \times V$ 是图中边的集合。n 和 m 分别是图中节点的个数、边的个数。节点 u 称作节点 v 的入邻点（或出邻点），当且仅当 (u,v)(或 (v,u)) 是 G 中的一条边。节点 u 的所有入邻点用符号 $I(u) = \{v : (v,u) \in E\}$ 表示，所有出邻点用符号 $O(u) = \{v : (u,v) \in E\}$ 表示。图中所有节点的平均入度（同样也是出度）用符号 d 表示。节点 u,v 的间的相似性由 $s(u,v)$ 表示，相应的，所有点对的相似度可以写作 $n \times n$ 的相似度矩阵 S，并且有 $s(u,v) = s_{uv}$。

SimRank 是一个基于结构上下文的相似度模型，它的核心思想是：如果两个节点的邻居节点非常相似，那么这两个节点也相似。节点 u,v 间的相似值用数学公式如下表达：

$$s(u,v) = \begin{cases} 1, & u = v; \\ \dfrac{c}{|I(u)||I(v)|} \displaystyle\sum_{u' \in I(u), v' \in I(v)} s(u',v'), & u \neq v. \end{cases} \tag{2-1}$$

其中，$0 < c < 1$ 衰减系数，用以提高临近结构对最终相似性的贡献权重，而降低更远结构的贡献权重。文献【1】证明了，上面的式子总是存在唯一的解，并且基于 SimRank 的定义，提出了一种基于矩阵乘法的迭代算法。设 S^k 是相似度矩阵 S 在第 k 轮迭代中的计算结果，S^0 为矩阵初始值，并且如果 $u = v$，有 $S_{uv} = 1$，否则 $S_{uv} = 0$。则矩阵 S^{k+1} 的计算方式如下：

$$S_{uv}^{k+1} = \begin{cases} 1, & u = v; \\ \dfrac{c}{|I(u)||I(v)|} \displaystyle\sum_{u' \in I(u), v' \in I(v)} S_{u'v'}^{k+1}, & u \neq v. \end{cases} \tag{2-2}$$

文献【1】已经证明，$\lim_{k \to \infty} S_{uv}^k = s(u,v)$。可以看出，该算法的时间复杂度为 $O(n^2)$，空间复杂度为 $O(kn^2d^2)$ time.

SimRank 的另一个模型是基于随机游走模型。节点序列 $W = v_0v_1v_2\dots v_l$ 如果对任意 $0 \le i \le l-1$ 都满足 (v_i, v_{i+1}) 都是 G 中的边，则称其为 G 中的一个游走。如果 W 还满足以下的 Markov 条件，即下式对所有的 $i \ge 1$, $v_0, v_1, \dots, v_i \in V$ 成立：

$$\begin{aligned} &Pr(X_i = v_i | X_0 = v_0, \dots, X_{i-1} = v_{i-1}) \\ &\quad = Pr(X_i = v_i | X_{i-1} = v_{i-1}) \end{aligned} \tag{2-3}$$

其中，X_i 表示在第 i 部，游走正好到达的节点。对任意 $u, v \in V$，$Pr(X_i = v | X_{i-1} = u)$ 表示在第 $i-1$ 步到达节点 u 的随机游走将在第 i 步到达节点 v 的条件概率。对于 SimRank 问题，每一条随机游走从某个节点出发，然后顺着图 G 中的逆变，每一个步骤随机走向该节点的某一个入邻点，并且特别的，转移概率定义为

$$Pr(X_i = v_i | X_{i-1} = v_{i-1}) = \begin{cases} \frac{1}{|I(v_{i-1})|}, & (v_i, v_{i-1}) \in E; \\ 0, & otherwise. \end{cases} \tag{2-4}$$

相应地，一条随机游走的概率，定义为：

$$Pr(W) = \prod_{i=1}^{l} Pr(X_i = v_i | X_{i-1} = v_{i-1}) \tag{2-5}$$

现在假设图 G 中有两条随机游走以相同速度分别从节点 u 和节点 v 同时出发，每次都顺着图中的逆边移动，也就是从当前所在节点移动到下一个入邻点，并且这两条随机游走在同一个节点 x 相遇并且为初次相遇，我们就把这两条终止在节点 x 的游走称作“匹配游走”。每一对匹配游走的长度就等于随机游走的步骤数。我们基于此定义相遇概率：

$$Pr\Big((W_u, W_v)\Big) = Pr(W_u) Pr(W_v) \tag{2-6}$$

文献【1】表明，节点 u, v 之间的相似度 $s(u, v)$ 等于服从上面概率分布的若干匹配游走的均值：

$$s(u, v) = \sum_{W_u, W_v} c^l Pr\Big((W_u, W_v)\Big) \tag{2-7}$$

其中，(W_u, W_v) 是一对任意的匹配游走，l 是游走的长度，c 是前文定义的衰减系数。

分布式单源点相似度计算方法

基于以上的 SimRank 定义以及其基于随机游走模型的解释，计算单源点相似度 $s(u,*)$ 是非常直接的。直觉上讲，$s(u,*)$ 的计算过程可以分解为对 G 中所有节点 v 计算 $s(u,v)$ 的过程。而要计算 $s(u,v)$，我们首先要找出分别从节点 u 和节点 v 出发的所有匹配游走，然后聚合这些游走计算出它们的相遇概率。注意到想要枚举出任意长度（包括正无穷）的随机游走显然是不可行的，并且显然考虑的长度越长，最后计算得到的相似性越精确。现实应用中，我们考虑长度为 k 的游走得到的数值可以满足一般应用。

算法中心思想

文献【11】给出了一个计算单点对 SimRank 相似度的 spSimRank 算法。算法的核心思想是两个随机冲浪者分别从节点 u,v 出发，每一步跟随所在节点的入边；每经过一个节点 t，原来的随机游走会产生 $|I(t)|$ 个新的游走。这样的话经过 k 步游走之后，总共会产生 $O(d^k)$ 个不同的游走，每条随机游走的长度不超过 k。这些游走使用了一个叫做 Path-Tree 的数据结构来保存。然后对两个 path-Tree 中的所有随机游走进行匹配过程，找出其中的匹配游走。显然与基于矩阵迭代计算的计算方式相比，该算法不需要考虑图中所有的节点。如果我们从这个算法那出发，可以轻易得出一个通过暴力枚举求单源点 SimRank 相似度的算法。

然而，直接通过多次调用 spSimRank 来计算单源点相似性有以下的缺点：

1. 总共生成了 $O(nd^k)$ 个游走，但其中大多数的游走都不能匹配到主游走
2. 尽管 Path-Tree 这种数据结构已经非常高效，但是考虑到会生成 n 个 Path-Tree，空间复杂度仍然比较高。在分布式环境下，因为需要频繁交换节点的局部拓扑信息，这回引起较高的网络开销。
3. 在最后的额匹配过程，统一长度的任意随机游走是以暴力方法匹配的，时间复杂度较高

Algorithm 3.1 Naive Single-source SimRank:nssSimRank

```
1: procedure SingleSourceSimRank(G, u, k)
2:     for l = 1 to k do
3:         W_u[l] ← all walks of length l starting from u;
4:     for v ∈ V(G) do
5:         s(u, v) ← spSimRank(G, W_u[], v, k);
6:     return s(u, ∗).
7: procedure spSimRank(G, W_u[], v, k)
8:     s(u, v) ← 0;
9:     for l = 1 to k do
10:        W_v[l] ← all walks of length l starting from v;
11:        s_l(u, v) ← 0;
12:        for w_u in W_u[l] do
13:            for w_v in W_v[l] do
14:                if w_v and w_u first meet at index l then
15:                    add score(w_u, w_v) to s_l(u, v);
16:                                          ▷ According to Eq. (2-7)
17:        add s_l(u, v) to s(u, v);
18:    return s(u, v).
```

算法的优化

减少随机游走的数量

如果直接使用 spSimRank 算法，总共会生成 $O(nd^k)$ 条随机游走，如果可以将系数 n 减少到某个值 c 使得 $C \ll n$，那么游走的数量会极大地减少。

定义 3-1 (倒叙游走) 我们直接定义随机游走的系列的倒叙序列为倒序游走。

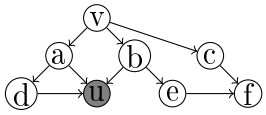
显然，原始的随机游走与其倒序游走存在一一对应关系。注意到随意游走是在 G 中跟随入边生成的，相应的，倒序游走跟随出边生成。并且，倒序游走的转移概率需要调整为

$$Pr(X_i = v | X_{i-1} = v_{i-1}) = \begin{cases} \frac{1}{|I(v_i)|}, & (v_{i-1}, v_i) \in E; \\ 0, & otherwise. \end{cases} \tag{3-1}$$

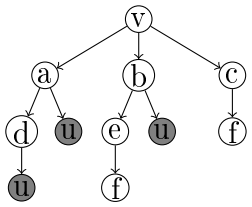
基于倒序游走的定义，可以观察到有下列现象：

$x, w'_1, \ldots, w'_{23}, w'_{32}, \ldots w'_1, x, \ldots$

图 3-1: The left are two walks starting from u and v respectively, first meeting at x; the right are reversed walks starting from x, passing u and v respectively.



(a) $N_G(v, 4)$.



(b) Path-tree of $N_G(v, 4)$.

图 3-2: (a) A neighborhood of v. (b) The corresponding Path-Tree representation of the neighborhood.

事实 3-1 假设两个分别从节点 u, v 开始的随机游走 W_u, W_v 经过 l 步后在节点 x 相遇，那么如果我们从节点 x 出发生成所有长度为 l 的倒序游走，W_u, W_v 的倒序游走一定也在其中

事实 3-2 假设 $W_u = uw_1w_2 \ldots w_l$ 是一条主游走，则所有有可能与 W_u 在 l 步相遇的从游走，只能在 $\{w_1, w_2, \ldots, w_l\}$ 中的任意一点相遇。也就是说，设 Nei 是节点 u 在 k 部可达的点集，那么只要生成从 Nei 出发的倒序游走就可以计算 $s(u, *)$。

以上两个事实的正确性是显然的。可以看出，总共需要生成的倒序游走的数量大约在 $O(|Nei|d^k)$ 而不是暴力算法的 $O(nd^k)$。事实上，$|Nei|$ 的复杂度为 $O(d^k)$，远比图 G 中点数目小，并且独立于 G 真实大小。

压缩中间数据的表示

相比于尽管生成游走的减少了很多，但是 $O(d^k)$ 复杂度依旧很高。而这些游走序列之间会有很多前缀子序列高度重合，如果单独地保存每一条游走，那么整个存储会有较大的冗余。TODO 为了解决多个游走序列共享很多前缀子序列，对 $|Nei|$ 中的每一个节点 v，我们并不使用任何特别设计的数据结构单独存储从 v 的所有随机游走，而是直接使用从 v 开始跟随逆边 k 可达的一个邻域 $N_G(v, k)$。严格的说，$N_G(v, k)$ 是节点从 v 跟随逆边 k 步可达的节点集合在 G 中的生成子图。例如，在图【4】中，展示了 v 的一个邻域。他所对应的 path-tree 显示在右图中，灰色的节点是我们欲查询的节点 u。可以看到，

$N_G(v,k)$ 自己本身就是 Path-Tree 的一种压缩表示。显然，我们在这一步中没有显式地表示出每一条游走，是因为在分布式环境中生成游走的过程需要在不同的计算节点上传输 G 中的局部拓扑信息，如果中间传输数据量过大，那么网络开销会极大地影响算法的最终性能。我们在下面会详细叙述即使没有显式表示每一条游走，算法仍然可以高效地计算出最终结果。

使用动态规划技巧加速随机游走的匹配

当 Nei 中的每个节点 v 收集到了从它自己开始的倒序游走 $N_G(v,k)$ 之后，这些从游走需要与主随机游走 $N_G(u,k)$ 进行匹配，从而得到最后的结果。由于代表所有主游走的 $N_G(u,k)$ 只是一个很小的信息 $(O(d^k))$，我们可以把它预先广播到分布式集群中的每一个节点上。而所有的从游走 $N_G(v,k)$ 按照 v 作为 key 分不到不同的节点上，然后每个计算节点直接在本地进行匹配过程。正因为这样，算法中最耗时的匹配部分是分布到每个计算节点上进行的，这是我们设计该算法的主要目的。

为了方便叙述，我们以在节点 v 第一次相遇的随机游走为例。注意整个匹配过程中，我们只对与主游走在 v 相遇，但是之后再也不和主游走有任何共同节点的逆序游走感兴趣。例如，在3-2b中，路径 vau 和 vbe 是一对对 $s(u,e)$ 有意义的匹配游走。同理，和 vau 能匹配上的随机游走还包括 vau 和 vcf。但是，vad 和 vau 不是一对合法的匹配游走，因为它们首次在节点 a 而不是 v 首次相遇。实际上，vad 和 vau 同样对 $s(d,u)$ 的计算做出了贡献，但是在分布式环境下，这个计算过程是由形如图3-2b、但 root 节点为 a 的 Path-Tree 引导的。这个计算过程也应当发生在该 Path-Tree 的计算节点上。基于以上的简单观察，可以发现以下事实:

事实 3-3 如果我们的查询节点 u 处于某个 Path-Tree 的某一层里，那么那一层的其他任一满足 $LCA(u,w)=v$ 的节点 w 都会对节点 u,w 的相似度 $s(u,w)$ 的计算做出大小为 $c^l Pr(W_u)Pr(W_w)$ 的贡献，其中 LCA 指的是最近公共祖先 (Lowest Common Ancestor)，W_u, W_w 是由对应 Path-Tree 展开的 root-to-node 路径。

我们使用 DFS 算法来搜索整个领域 Nei_v。 在 DFS 过程中，每一条 root-to-node 路径的概率被记录下来。当 DFS 的深度达到 l 时，算法停止，对应逆序游走的概率被保存在一个 HashMap 中。当我们再次匹配长度为 $l+1$ 的

主游走时，就不再需要从节点 v 开始我们的 DFS 过程，因为该 Path-Tree 中长度小于等于 l 的逆序游走的概率之前已经全部计算过了。例如，在图3-2b中，能和游走 $vbdu$ 匹配的唯一游走是 $vbef$，而计算 $vbef$ 的概率时，我们只需要从节点 e 开始 DFS 过程，因为 vbe 的概率在之前搜索 vae 的匹配游走时已经计算被保存过了。

因此为了克服多个游走之间共享了很多重复前缀这一问题，我们把匹配游走这一问题看做是一个动态规划 (Dynamic Programming) 问题，通过 Memorization 的技巧来降低匹配的时间复杂度。算法的具体细节在3.2中列出。程序 LevelMatch 展示了一个 Path-Tree 中的根节点 v 如何寻找一个长度为 l 的主游走。各参数的意义如下：W_l 是长度为 l 的主游走的集合；N 是从节点 v 开始展开的领域；$M_{l'}$ 保存了之前调用 LevelMatch 函数来匹配长度为 l' 主游走得到的匹配概率，具体地说它是一个 (key,value) 形式的 HashMap, 其中 key 是 v 的邻居节点，表示该 k-v 对保存的是以 Path-Tree 中第二层节点为根节点的哪一棵子树的信息，value 是一串元素，每个元素记录着以 key 为根节点的子树中所有的游走的终止节点以及对应游走的概率；类似的 M_l 是一个将要被填充的空的 HashMap，保存匹配长度为 l 的游走的匹配概率；最后一个参数 δ 是一个概率阈值，其具体的含义将在下一节给出。我们首先循环 W_l 中的每一条主游走 p，并且知道该主游走在 Path-Tree 中的哪一棵子树中（2-3 行），如果 M_l 中没有记录该子树中匹配游走的概率，就开始 DFS 过程（4 行）。我们先检查 $W_{\square}l\prime\square$ 中是否有记录该子树的信息，如果有的话就直接从记录的游走的终止节点开始调用 DFS(5-6 行），否则就需要从该子树的根节点开始 DFS。程序 DFS 就是一个常见的 DFS 过程。通过对所有的 $W_l, (l \le k)$，所有游走的匹配概率都可以高效地计算出来。

使用概率阈值剔除极小的概率

很多现实的大图都是 scale-free【2 1】的，这意味这图中的极小比例的节点会有极高的度数。我们的算法的性能与节点的度数紧密相关，因为在生产游走过程中每条游走每经过一个度数为 d 的节点，都会在那个节点分裂为 d 条更长的游走。因此，我们使用一个阈值 δ 来删除那些包含多个拥有极高度数的节点的游走，因为根据公式【TODO】，这些游走对的概率非常之小，对最终的计算精度基本可以忽略不计。δ 的取值应当注意在算法的精度与时间、空间复杂度取得平衡，更大的 δ 意味着更低的精度，当时算法整体的时间复杂度和

Algorithm 3.2 Dynamic Programming Path Matching

```
1: procedure LevelMatch(W_l, N, v, M_{l'}, M_l, δ)
2:     for p ∈ W_l do
3:         br ← second last vertex of p;
4:         for t ∈ (v_N.neighbors − br) & t ∉ M_l do
5:             if !M_{l'}.contains(t) then
6:                 DFS(N, t, l, M_l, δ, t, 1, 1);
7:             else
8:                 for w ∈ M_{l'} do
9:                     for nei ∈ w_N.neighbors do
10:                        DFS(N, nei, l, M_l, δ, br, w.mul, l');
11:    return M_l.
12: procedure DFS(N, v, l, M, δ, br, mul, depth)
13:    mul ← mul * v_N.indegree;
14:    if mul > δ then
15:        return;                                  ▷ Early termination.
16:    if depth = l then
17:        add (v, mul) to M(br);                   ▷ Record probability.
18:    else
19:        for nei ∈ v_N.neighbors do
20:            DFS(N, nei, l, M, δ, br, mul, depth + 1);
21:    return M.
```

空间复杂度更低，反之反是。

算法的复杂度分析

我们综合分析一下算法的复杂度。源点 u 可达的节点的数目大约为 $O(d^k)$，而对于其中的每一个可达节点，都拥有一个大小为 $O(d^k)$，隐式包含 $O(d^k)$ 条逆序游走信息的领域，因此，总的空间复杂度为 $O(d^{2k})$。因为这个过程中所有产生的游走都需要通过网络传输，所以通信开销也是 $O(d^{2k})$。在对游走进行匹配时，对每一个领域，大约有 $O(d^k)$ 个游走被匹配了，所以总的计算复杂度为 $O(d^{2k})$。可以看到，算法总的复杂度与整个图的规模 $O(n+m)$ 没有以来关系，所以我们的算法是非常高效的。

基于 Spark 平台的实现

实验评估

本章节重点描述

实验环境

所有的实验在一个由 6 个硬件完全相同的计算节点组成的集群上完成，每台计算节点处理器为 12 核的 Intel Xeon E-2650，频率为 2.1GHz，内存为 64GB，硬盘为 1TB。计算节点之间由千兆网卡连接。所有的节点上运行 Ubuntu 16.04 操作系统。Spark 运行版本为 1.6.2，底层分布式文件系统 HDFS 的版本号为 2.6.0。所有的 6 个节点都配置为 slave 节点，其中一个节点被二外配置为 master 节点。在 Spark 运行时，我们为其分配了 10GB 的内存。

实验数据集

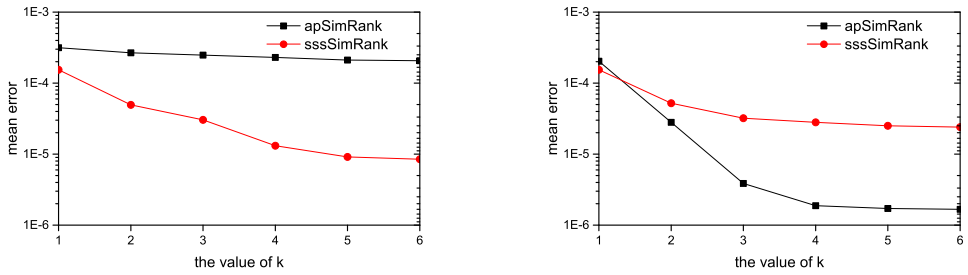
我们一共使用了 5 个真实的图数据来评估算法的性能。数据集的具体细节如表3-1。每个图一开始为普通的文本形式，每一行代表这图中的一条边。在开始实验前，所有的数据集都预先上传到分布式文件系统 HDFS 上。

表 3-1: 数据集描述

数据集	顶点数	边数	顶点平均度数	大小
p2p-gnutella08 ①	6,301	20,777	3.29	215.2KB
wiki-vote ②	7,115	103.689	14.57	1.1MB
eu-2005 ③	862,664	19,235,140	22.29	256.4MB
ljournal-2008 ④	5,363,260	79,023,142	14.73	1.2GB
arabic-2005 ⑤	22,744,080	639,999,458	28.14	10.9GB

实验参数设置

根据文献文献【9】中的描述，通常算法所需要的最大迭代次数 k 是由衰减系数 c 和算法的预期精度决定的。如果希望最终误差 $s^*(u,v) - s^k(u,v)$ 小于某个



(a) p2p-gnutella08 (b) wiki-vote

图 3-3: apSimRank 和 sssSimRank 的相似性误差随迭代次数的变化曲线比较

ϵ，那么 k 至少要被设置为 $k = \lfloor \log_c \epsilon \rfloor$。在我们的实验中，我们选取 $\epsilon = 0.01$，因为这样的精度可以满足大多数实际应用。c 被设置为 0.5，相应的，k 被设置为 $k = 6$。

此外，用于过滤掉极小概率的游走的阈值被设置为 $\delta = 0.002$。主要到这里的 δ 指的是一个随机游走的概率阈值，根据 2-6对于一对匹配好的游走，其对应的匹配概率相应的变成 δ^2。如果再考虑到因子 c^l，那么这个概率是极端小的，完全可以忽略。

算法精度的测试采用的方法是多次实验取平均值，每次实验时随机选取图中的某个顶点为查询顶点。具体的，对与小图（大小 <10MB），我们重复实验 100 次计算其平均值；对于更大的图，重复次数设为 1000。

算法有效性

我们首先比较我们的算法和全点对算法的精度和收敛速度。我们选取的精度指标为平均误差 (mean error)，即 $ME = \frac{1}{n}\sum_{v \in V} \left| s(u,v) - s^k(u,v) \right|$，其中 $s(u,v)$ 为根据 2-2迭代计算至完全收敛的真实相似性，$s^k(u,v)$ 为我们算法迭代 k 次后的相似性。我们在两个小图上进行实验。其中 p2p-gnutella08 是个比较稀疏的图，平均顶点度数 $d = 3.29$，而 wiki-vote 是一个更加稠密的图，平均顶点度数为 $d = 15.57$。图 ??为最终的比较结果。从图中可以看到，全点对算法和我们的算法在 6 次迭代以内精度都得到了收敛，我们的算法 sssSimRank 有更好的收敛速度，三次迭代之后平均误差就在 10^{-4} 以内。另一个现象是在图 3-3a中 sssSimRank 的平均误差与 apSimRnak 的差距比图 3-3b更小，这是因为我们的算法中使用了概率筛除的缘故。对于同一个概率阈值 δ，图的平均顶点度数越大，小概率游走越多，相应的，被剔除的小概率游走越多，所以对最终精度的

影响越来越大。本质上，δ 的作用是牺牲一定的精度来换取计算效率的提高。即便如此，sssSimRank 的相似性误差仍然非常小（$< 10^{-4}$），完全可以满足大部分应用的精度需求。

算法的效率

为了比较算法的运行效率，我们分别基于 Spark 平台实现了分布式的全点对算法 apSimRank，3.1算法 nssSimRank，以及本文提出的算法 sssSimRank。直接比较这三个算法的运行时间非常困难，因为 apSimRank 以及 nssSimRank 实际运行非常耗时，在规模最小的图 p2p-gnutella08 上，apSimRank 需要运行 2.1 个小时才能完成计算，而 nssSimRank 需要计算 1.2 个小时。因为对于基于随机游走模型的算法而言，其运行效率主要取决于所生产的随机游走的数量，进一步地说，取决于需要从多少个领域 *Nei* 来展开生产随机游走。因此，我们比较 sssSimRank 和 nssSimRank 算法中生产 *Nei* 的数量。结果如图3-4所示。从图中可以看出，sssSimRank 极大地减少了生成领域的数量。当 $\delta = 0$ 时，

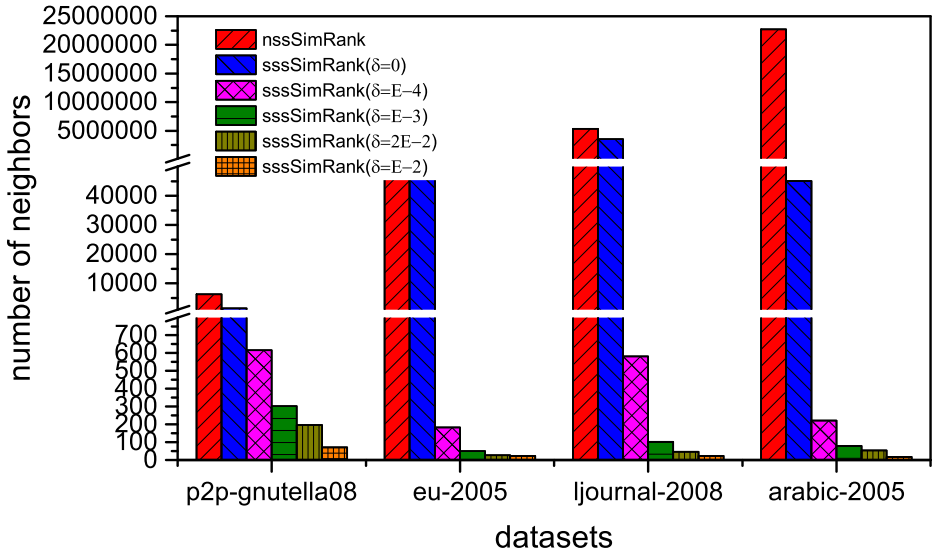


图 3-4: 生成领域数量的比较

即算法没有筛除概率极小的游走时，领域数量减少了大约 1500x 倍。从图中还可以观察到领域数量与概率阈值 *delta* 之间的关系。当 δ 越来越大时，意味这概率“筛子”的“孔”变得越来粗，幸存的随机游走会变得越来越少。图3-4还表明，图 eu-2005 和 arabic-2005 减少领域的比例比图 p2p-gnutella08

和 ljournal-2008 要大很多，这一观察同样表明算法中的概率筛子对稠密的图能更好地提高性能。

算法的可扩展性

我们考察分布式环境下算法的可扩展性。首先考察算法运行时间随着输入图的大小的变化关系，结果如图 3-5所示。

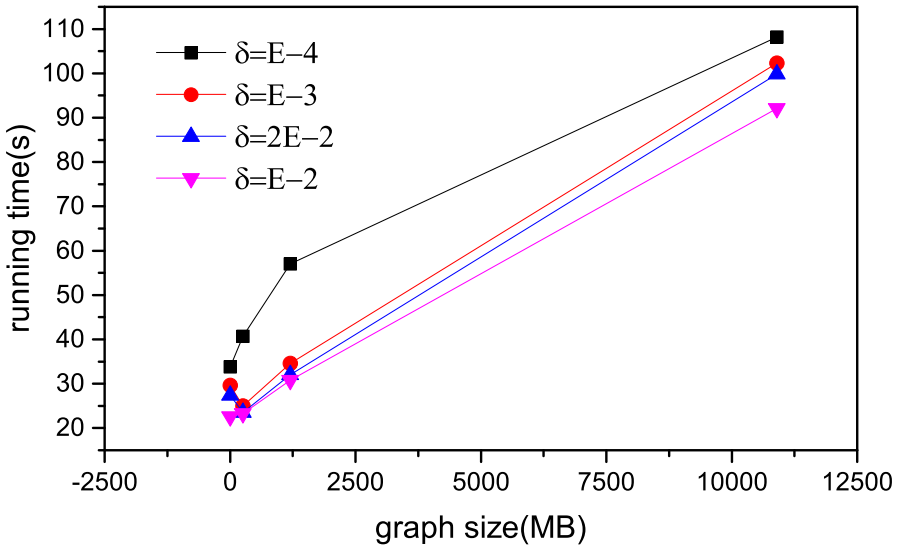


图 3-5: 算法运行时间与输入图大小的变化关系

图中展示了当集群计算节点数目固定时，对于不同的 δ，运行时间随输入图规模大小变化的情况。可以看出，输入图的规模从 215KB 到 10.9GB，而对应的运行时间基本上近似的随着输入规模大小线性的变化，这一结论对不同的 δ 都成立。这展示出 sssSimRank 良好的数据可扩展性。我们还考察了当输入图的规模固定时，算法运行效率随集群中计算节点数量变化的情况。所有的输入图使用同样的参数配置，$k = 6$，$\delta = 10^{-4}$。计算节点数量从 2 增加到 6。注意到我们把 δ 取得非常小，是因为 *delta* 越小，算法剔除的游走越少，算法的计算量越大。这样更能提现计算量较为饱和的情况下算法的节点可扩展性。实验结果如图3-6所示。从图中可以看到，对不同规模的输入图，算法的运行时间随集群计算节点的增加而近乎线性的减少，这表面算法在分布式环境下具有良好的节点可扩展性。

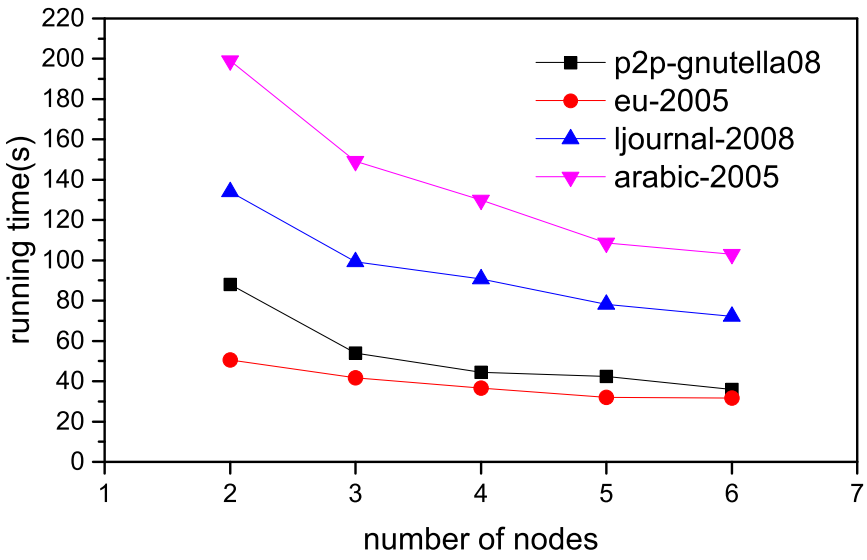


图 3-6: 算法运行时间与集群计算节点数量的变化关系

小结

基于 Spark 的分布式图分割方法

为了计算图 G 中所有节点的相似度，我们将在下一章节介绍 cSimRank 算法。而本章节主要介绍一种分布式的图分割算法，它是 cSimRank 的中间步骤。作为一种常见的基础算法，目前已经有大量的工作对图分割做了各种形式的研究。文献【23】证明了，图分割是 NP 完全问题。而对于目前不断增长的大图来说，图分割问题面临着许多新的挑战：

1. 真实的图拓扑结构往往节点分布极不规则，这往往使得并行算法或分布式算法无法拥有很好的并发度或并行度
2. 真实图的节点的度数往往表写出非常倾斜的分布，这使得不同处理器上的负载无法均衡

针对以上问题，我们提出一种高效的分布式图分割算法，实验结果表面，该算法具有较高的效率，较好的可扩展性。

图分割概述

对于大规模图而言，图分割意味着把图中的节点划分为不同的分割，并将其存储在集群中不同的计算节点上去。在这个过程中，为了能够访问到非局部的拓扑信息，网络开销是必不可少的。换句话说，采取什么样的方法对图分割，对分割过程中的网络开销和负载均衡有种决定性影响。

相关工作

目前针对图分割的一些研究，主要有以下几种 **单节点串行算法**文献【23】证明了图分割问题是一个 NP 完全问题。因此，早起很多工作都集中在设计一些近似算法以求得次优解。文献【5】提出了 KL 算法，该方法将图划分为偶数个划分，基于一些启发式的信息不断交换不同划分对中的节点，使得不同划分之间的边割 $(edge-cut)$ 的权重最小；类似的方法还有文献【6】研究了 FM 算法。还有一些基于模拟退火【24】，遗传算法【25】的解决方案。这些方法研

究的都是基于内存的小图。为了能够应对更大规模的图，一些多层次分割的方法相继被提出，包括 Metis[8]Chaco[7] 和 Scotch[10]。这些方法的主要思想是首先将局部的某些节点变为一个节点，从而提取出整个图拓扑信息的“骨骼”，这一过程成叫做”塌缩“（coarse）; 这样图的规模得到了不断的缩小，当规模达到某个可以接受的规模后，然后再使用传统的 KL 或 FM 算法对其进行处理；然后通过一个“uncoarse”的过程还原出原来的图。

单节点并行算法为了进一步充分利用现代 CPU 的多核特性，从以上这些算法中演化出一些并行化算法，例如 ParMetis[11], Pt-Scotch[12] 等。因为这些方法中有部分步骤可以并行，所以其能够处理的图的规模也获得了的提高。然而总的来说，这些方法仍然是基于单计算节点的，对规模更大的（上百万节点）图无能为力。

分布式图划分近年来，一些专门用于图计算的分布式平台越来越流行，包括 Pregel[15]，Giraph[X]，Spark GraphX[TODO] 等等。这些平台的一个重要特点是支持以节点为中心 (vertex-centric) 的编程范式， 并且支持节点之间的消息通信。 在处理图计算问题时，这些平台内部默认地使用了简单的随机划分方法，其中一些也开放了一些接口支持用户设计按照需求设计自己的分割方法。 但是并没有提供一个能够使用的图分割算法。

算法的基本框架

我们从两方面来考虑图划分算法的优化目标:

1. 最终划分结果的负载均衡。考虑一个由 k 个硬件条件完全相同的计算节点组成的集群，因为存储资源（硬盘）存在于每个计算节点上，所以理想情况下最终划分的结果应当尽可能的均匀分布在集群中。也就是说，每个划分的大小应当在 $\frac{|V|}{k}$ 左右，或者说划分过程中应该避免规模大于 $\frac{|V|}{k}$ 的分块的产生。如果仍然有某个分块分布式地存储在超过一个结算节点上，那所谓的图划分就没有意义。
2. 最终划分结果的质量。理想情况下图划分算法应该可以把图中稠密的子图归类于一个分块中。但是因为有负载均衡的限制，而图中稠密子图的规模天然是不均匀的，因此我们从整体考虑划分的质量。设 C 是图 $G(V,E)$ 的一个划分，定义其权重为 $W_P = \sum_{e(u,v)\in E, u\in C_i, v\in C_j} w\Big(e(u,v)\Big)$，即所有连接不同分块的边的权重之和。理论上，$W_P$ 应尽可能的小。

Algorithm 4.1 Multi-Level Graph Partitioning

```
1: procedure MultiLevelGraphPartition(G, α, β, k)
2:     while #partitions ≥ α do
3:         P ← results of modularity-based community detection for β iterations;
4:         Construct G′ from P;                                  ▷ Coarsen.
5:     P_1 ← Run local graph partitioning on G′.                 ▷ Final refinement.
6:     restore partition P′ from P_1                             ▷ Project back to the orignal graph.
```

我们的算法借鉴了图塌缩的思想，即通过不断的 coarse 使得原来的图的规模越来越小，当图的规模小到某个条件时，再使用 KL 或 FM 处理。METIS 是这种多层次图分割算法的典型代表。它主要有三个步骤，首先对原图进行反复的塌缩，然后对获得的图进行划分，然后再进行反塌缩过程恢复出对原图的划分。第一个步骤，METIS 使用寻找最大匹配 (maxima l match) 的方法来对原图进行 coarsen。形式上说，一个最大匹配就是一个包含做多边的集合，使得集合中每条边的不存在重合的节点。最大匹配中的每条边都可以塌缩为新图 G'_1 中的一个节点，这样图的规模就能不断的减小。第二步骤中，直接使用KL或FM对 G'_k 进行划分；最后一个步骤中，G'_k 的划分被重新映射到原图 G 中去。我们给出算法的基本框架 算法包含三个参数，α,β,k。其中，α 指的是反复塌缩过程的目标划分数目，如果迭代过程中当前的划分数目大于 α，就继续迭代下去。实际应用中，α 的取值与第二步中单节点图划分算法可以处理的输入图规模有关。β 指的是我们基于模量的塌缩方法的迭代次数。k 是调节负载均衡的系数。

图的塌缩

文献【TODO】给出了一种基于标签传播 (Label Propogation，LP) 的分割思想。LP算法一开始被运用于社区发现，即在图 G 中将不同的节点划分为不同的社区，每个社区的内部节点往往连通得比较紧密，社区之间的节点往往连通比较稀疏。这种方法天然地借鉴到图划分问题中，因为在图划分问题中，往往也要求每一个分块之间的节点紧密连通，而不同划分之间应该较少连通或理想状况下不连通。LP算法的思想非常简单：最初的时候我们给每个节点一个单独的社区 ID, 然后我们迭代式地为每个节点更新社区ID。每次迭代中，每个节点按照其邻居节点中出现最多的社区ID作为自己的社区I

D。当 G 中每个节点的社区 ID基本上不再变动时，算法结束。上述算法并没有使用 G 中任何的先验信息或者是前置的目标函数，因此有很搞的提升空间。我们这里借鉴了文献【7】中的模量的思想，即在迭代过程中有方向性地寻找合适的社区 ID。我们首先给出如下定义设关系 (V,E) 表示一个图，其中 V 图的节点集合，$E \subseteq V \times V$ 是图中边的集合。n 和 m 分别表示图中节点的个数、边的个数。 d_v 表示节点 v 的度数, c_v 表示节点 v 所在社区的 ID。这样根据文献【TODO】中的定义， 一个无向连通图的模量定义如下:

$$Q = \frac{1}{2m} \sum_{u,v \in V} \left[w_{uv} - \frac{d_u d_v}{2m} \right] \delta(c_u, c_v), \tag{4-1}$$

其中 $\delta(c_u, c_v)$ 表示节点 u, v 是否处于一个社区，即如果 $c_v = c_u$， 那么 $\delta(c_u, c_v)$ 等于 1；否则为 0。按照上面的定义，我们有如下的定理:

定理 4-1 根据上面的定义，我们实际上可以得到模量 Q 定义的简单形式:

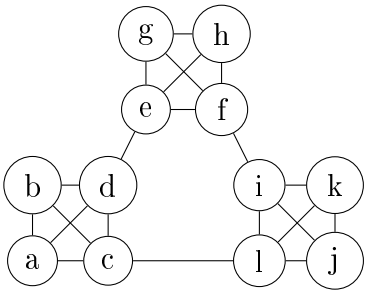
$$Q = \sum_{c \in C} \left[\frac{I_c}{2m} - \left(\frac{S_c}{2m} \right)^2 \right] \tag{4-2}$$

其中， I_c 表示 ID为 c 的社区中两个定点都在 c 中的边的数量， 而 S_c 表示社区 c 中所有节点的度数的综合，即 $S_c = \sum_{v \in V} d_v$

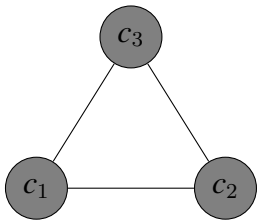
证明: 假设 c 是图 G 中的某个划分， 根据的定义，我们知道如果节点 u, v 属于不同的社区的话，那么显然有 $\delta(c_u, c_v) = 0$, 因此我们只要考虑属于同一社区的点对就可以了， 即我们可以把改写为:

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{u,v \in V} \left[w_{uv} - \frac{d_u d_v}{2m} \right] \delta(c_u, c_v) \\ &= \frac{1}{2m} \sum_{c \in C} \sum_{u,v \in c} \left[w_{uv} - \frac{d_u d_v}{2m} \right] \\ &= \frac{1}{2m} \sum_{c \in C} \left[\sum_{u,v \in c} w_{uv} - \frac{\sum_{i,j \in c} d_u d_v}{2m} \right] \end{aligned} \tag{4-3}$$

其中，显然我们有 $I_c = \sum_{u,v \in c} w_{uv}$, 并且有 $\sum_{i,j \in c} d_u d_v = \sum_{u \in c} d_u \sum_{v \in c} d_v = S_c S_c =$



(a) 一个图 G



(b) 经过塌缩后的图 $G\prime$

图 4-1: (a) 为一个简单的示例图，(b) 为改图经过塌缩 (coarse) 后的图 $G\prime$, 每个节点代表 G 中的一个划分

${S_c}^2$ 因此， 我们有如下的定理：

$$\begin{aligned} Q &= \frac{1}{2m}\sum_{c\in C}\left[I_c - \frac{S_c^2}{2m}\right] \\ &= \sum_{c\in C}\left[\frac{I_c}{2m} - \left(\frac{S_c}{2m}\right)^2\right] \end{aligned} \tag{4-4}$$

□

有了以上的定以后，我们回到算法的框架中。设原始输入图为 $G(E,V)$，并假设迭代塌缩过程中产生的中间图用记号 $G_1, G_2, \ldots, G_t$ 表示， 相应的，迭代过程中由中间图产生的划分称作 $P_1, P_2, \ldots, P_t$。 注意到有 $P_i \leftarrow G_i$，即迭代中图划分与其输入是对应的。然后我们有以下定义：

定义 4-1 设基于图 $G_i(V_i, E_i)$ 有划分 $P_i = (C_1, C_2, \ldots, C_n)$， 基于 P_i 上构造塌缩图 $G_{i+1}(V_{i+1}, E_{i+1}$ 如下：令 $V_{i+1} = P_i$， 即 P_i 中的每一个划分对应 G_{i+1} 中的一个新节点； 边 $(C_i, C_j) \in E_{i+1}$ 当且仅当 $\exists u \in C_i, v \in C_j$ 并且 $(u,v) \in E_i$。

定义 4-2 设基于图 G_{i+1} 迭代过程中基于图 G_i 生产的压缩图，并且有 $V_{i+1} = (C_1, C_2, \ldots, C_n)$，那么边 $e = (C_i, C_j)$ 的新权重定义为：

$$w(C_i, C_j) = \sum_{e(u,v)\in E_i, u\in C_i, v\in C_j} w\Big(e(u,v)\Big) \tag{4-5}$$

即压缩图中的新边的权重等于该边在原图中代表的两个划分中所有相连接的边的权重之和。

我们下面给出基于模优化的图划分方法。我们首先考虑，假设节点 u 属于

划分 c, 而它的所有邻居节点所属的划分为 $C = \{c_1, c_2, c_3, \ldots, c_n\}$，那么对某个 $c_i \in C$，如果我们考虑把 u 从 c 中移出，重新放入到 c_i 中，整个图 G 的模量会一次发生变化，我们用符号 $\Delta Q_{u,c}$。并且根据模量的定义，我们有：

$$\Delta Q_{u,c} = \left[\frac{I_c + w_{i,c}}{2m} - \left(\frac{S_c + l_i}{2m}\right)^2\right] - \left[\frac{I_c}{2m} - \left(\frac{S_c}{2m}\right)^2 - \left(\frac{l_i}{2m}\right)^2\right] \tag{4-6}$$

经过若干步骤的化简，最终我们可以得到模量变化的简单形式：

$$\Delta Q_{u,c} = \frac{w_{i,c}}{2m} - \frac{S_c l_i}{2m^2} \tag{4-7}$$

我们对图进行塌缩的算法如下所示：

Algorithm 4.2 Modularity-based Graph Coarsening

```
1: procedure GraphCoarsening(G, β)
2:     while # iterations ≤ β do
3:         Initialize C, which assigns each node a unique partition identifier;
4:         m = Σ_{u,v∈V} w_{u,v};
5:         for u ∈ V do
6:             maxGain = −1;
7:             for v ∈ N_v do
8:                 ΔQ_{u,c_v} = w_{i,c}/2m − S_{c_v} l_i / 2m²;
9:                 if ΔQ_{u,c_v} > maxGain then
10:                    maxGain = ΔQ_{u,c_v}
11:                    newPartition = n_v
12:            if maxGain > 0 then
13:                c_newPartition ← c_newPartition ∪ {u};
14:                c_u ← c_u − {u};
15:    return C.
```

图的划分

算法的优化

算法的实现

基于 Spark 的分布式全点对相似度计算方法算法

算法理论模型

算法概览

结论

本文在第2章中，通过考虑数据中心网络布局构建中的最大度限制问题，提出了符合数据中心网络基本要求的 DS 小世界模型，并分析了它的性质。随后提出 SIDN，将 DS 模型映射到具体的网络结构中，并分析了所构成网络的平均直径、网络总带宽、对故障的容错能力等各项网络性能。

分析与仿真实验证明，SIDN 网络具有很好的扩展能力，网络总带宽与网络规模成近似线性增长的关系；具有很强的容错能力，链路损坏与节点损坏几乎无法破坏网络的联通性，故障率对网络性能的影响与破坏节点/链路占总资源比率线性相关。

随后在第??章中，分析了无尺度网络在数据中心网络构建应用中的理论方面问题。对 Scafida[13] 文中所述在最大度限制的情况下运用 BA 算法构造的网络并不会损失无尺度性质的观点，进行了深入的分析，并指出了该论点的局限性。

在给出了在引入节点最大度限制之后，利用分治和递归的思想，对无尺度网络进行多层构建，对所构造的网络进行度-度相关性，以及聚类性分析。

表6-1用于测试表格。随后分析了无尺度网络构造过程中，交换机节点与数据节点的角色区别，分析了两者在不同比率下形成的网络形态，以及对网络性能造成的影响。

通过理论分析和仿真实验，分析并找出比率因子 q 的最佳取值。此外，无尺度现象的引入提高了网络的聚类系数，从而在不失灵活性可靠性的基础上，进一步提升了网络的性能。

在第??章中，将关注点转移到交换机本身。由于图论难以描述数据中心网络中的交换设备，因此放弃基于图的抽象模型，转而基于多维簇划分的思想，提出并设计了 WarpNet 网络模型。

该网络模型突破了基于图描述的局限性，并对网络的带宽等指标进行理论分析并给出定量描述。最后对比了理论分析、仿真测试结果，并在实际物理环境中进系真实部署，通过 6 节点的小规模实验以及 1000 节点虚拟机的大规模实验，表明该模型的理论分析、仿真测试与实际实验吻合，并在网络性能、容

表 6-1: 测试表格

文档域类型	Java 类型	宽度 (字节)	说明
BOOLEAN	boolean	1	
CHAR	char	2	UTF-16 字符
BYTE	byte	1	有符号 8 位整数
SHORT	short	2	有符号 16 位整数
INT	int	4	有符号 32 位整数
LONG	long	8	有符号 64 位整数
STRING	String	字符串长度	以 UTF-8 编码存储
DATE	java.util.Date	8	距离 GMT 时间 1970 年 1 月 1 日 0 点 0 分 0 秒的毫秒数
BYTE_ARRAY	byte[]	数组长度	用于存储二进制值
BIG_INTEGER	java.math.BigInteger	和具体值有关	任意精度的长整数
BIG_DECIMAL	java.math.BigDecimal	和具体值有关	任意精度的十进制实数

错能力、伸缩性灵活性方面得到了进一步的提升。

在第 ??章中，针对网络模型研究这一类工作的共性，设计构造通用验证平台系统。以海量虚拟机和虚拟分布式交换机的形式，实现了基于少量物理节点，对大规模节点的模拟。其模拟运行的过程与真实运行在实现层面完全一致，运行的结果与真实环境线性相关。除为本文所涉若干网络模型提供验证外，可进一步推广到更为广泛的领域，为各种网络模型及路由算法的研究工作，提供分析、指导与验证。

致　　谢

白驹过隙间，我的三年研究生就要结束，我在南京大学七年的学习生活就要画上句号。在这即将离别校园之际，回想三年的校园生活，很多情景都历历在目。研究生阶段，我的科研能力得到了提升，生活上也变得更加独立。在这里我衷心感谢我的导师、师兄弟、同学、朋友们，你们平时对我的教导、支持、帮助让我愉快地走完了这平凡的三年，感谢你们!

首先感谢我的导师唐杰。唐老师为人谦和，工作负责，在学术上对我们都尽心尽力地指导，在生活上对我们也多有照顾。无论是本课题的研究及论文的撰写，还是平时对我科研能力的塑造，唐老师都以身作则，对我提供了很多帮助。唐老师宽厚谦和的生活作风也潜移默化地影响着我，指导着我做一个脚踏实地的南大人。

我还要感谢武钢山老师。武老师是多媒体教研室主任，他作风严瑾，要求严格，把握着最新的科研动向，同时管理着整个大组的研究学习工作。在次感谢武老师对我学习和生活上的种种帮助。

还要感谢实验室的师兄师弟以及师妹们。三年以来你们给了我很多的关心和支持，很高兴能和你们同门，你们让我留下了很多美好的回忆。这里祝愿两位老师身体健康，万事如意！也祝各位同学学业有成，科研顺利！最后感谢我的家人，他们永远是我最大的支柱和依靠!

博士 (硕士) 学位论文编写格式规定 (试行)

适用范围

本规定适用于博士学位论文编写，硕士学位论文编写应参照执行。

引用标准

GB7713 科学技术报告、学位论文和学术论文的编写格式。

GB7714 文后参考文献著录规则。

印制要求

论文必须用白色纸印刷，并用 A4(210mmŒ297mm) 标准大小的白纸。纸的四周应留足空白边缘，上方和左侧应空边 25mm 以上，下方和右侧应空边 20mm 以上。除前置部分外，其它部分双面印刷。

论文装订不要用铁钉，以便长期存档和收藏。

论文封面与封底之间的中缝（书脊）必须有论文题目、作者和学校名。

编写格式

论文由前置部分、主体部分、附录部分 (必要时)、结尾部分 (必要时) 组成。

前置部分包括封面，题名页，声明及说明，前言，摘要 (中、英文)，关键词，目次页，插图和附表清单 (必要时)，符号、标志、缩略词、首字母缩写、单位、术语、名词解释表 (必要时)。

主体部分包括绪论 (作为正文第一章)、正文、结论、致谢、参考文献表。

附录部分包括必要的各种附录。

结尾部分包括索引和封底。

前置部分

封面（博士论文国图版用）

封面是论文的外表面，提供应有的信息，并起保护作用。

封面上应包括下列内容:

1. 分类号在左上角注明分类号，便于信息交换和处理。一般应注明《中国图书资料分类法》的类号，同时应注明《国际十进分类法 UDC》的类号;
2. 密级在右上角注明密级;
3. “博士学位论文”用大号字标明;
4. 题名和副题名用大号字标明;
5. 作者姓名;
6. 学科专业名称;
7. 研究方向;
8. 导师姓名，职称;
9. 日期包括论文提交日期和答辩日期;
10. 学位授予单位。

题名

题名是以最恰当、最简明的词语反映论文中最重要的特定内容的逻辑组合。

题名所用每一词语必须考虑到有助于选定关键词和编写题录、索引等二次文献可以提供检索的特定实用信息。

题名应避免使用不常见的缩略词、首字母缩写字、字符、代号和公式等。

题名一般不宜超过 20 字。

论文应有外文题名，外文题名一般不宜超过 10 个实词。

可以有副题名。

题名在整本论文中不同地方出现时，应完全相同。

前言

前言是作者对本论文基本特征的简介，如论文背景、主旨、目的、意义等并简述本论文的创新性成果。

摘要

摘要是论文内容不加注释和评论的简单陈述。

论文应有中、英文摘要，中、英文摘要内容应相同。

摘要应具有独立性和自含性，即不阅读论文的全文，便能获得必要的信息，摘要中有数据、有结论，是一篇完整的短文，可以独立使用，可以引用，可以用于推广。摘要的内容应包括与论文同等量的主要信息，供读者确定有无必要阅读全文，也供文摘等二次文献引用。摘要的重点是成果和结论。

中文摘要一般在 1500 字，英文摘要不宜超过 1500 实词。

摘要中不用图、表、化学结构式、非公知公用的符号和术语。

关键词

关键词是为了文献标引工作从论文中选取出来用于表示全文主题内容信息款目的单词或术语。

每篇论文选取 3 − 8 个词作为关键词，以显著的字符另起一行，排在摘要的左下方。在英文摘要的左下方应标注与中文对应的英文关键词。

目次页

目次页由论文的章、节、附录等的序号、名称和页码组成，另页排在摘要的后面。

插图和附表清单

论文中如图表较多，可以分别列出清单并置于目次页之后。

图的清单应有序号、图题和页码。表的清单应有序号、表题和页码。

符号、标志、缩略词、首字母缩写、计量单位、名词、术语等的注释表符号、标志、缩略词、首字母缩写、计量单位、名词、术语等的注释说明汇集表，应置于图表清单之后。

主体部分

格式

主体部分由绪论开始，以结论结束。主体部分必须由另页右页开始。每一章必须另页开始。全部论文章、节、目的格式和版面安排要划一，层次清楚。

序号

图 A-1: 测试附录中的插图

论文的章可以写成：第一章。节及节以下均用阿拉伯数字编排序号，如 1.1，1.1.1 等。

论文中的图、表、附注、参考文献、公式、算式等一律用阿拉伯数字分别分章依序连续编排序号。其标注形式应便于互相区别，一般用下例：图 1.2；表 2.3；附注 1）；文献 [4]；式 (6.3) 等。

论文一律用阿拉伯数字连续编页码。页码由首页开始，作为第 1 页，并为右页另页。封页、封二、封三和封底不编入页码，应为题名页、前言、目次页等前置部分单独编排页码。页码必须标注在每页的相同位置，便于识别。

$$C_i = \frac{2E_i}{k_i(k_i - 1)} \tag{A-1}$$

附录依序用大写正体 A、B、C、$\cdots$ 编序号，如：附录 A。附录中的图、表、式、参考文献等另行编序号，与正文分开，也一律用阿拉伯数字编码，但在数码前题以附条序码，如图 A.1；表 B.2；式 (B.3)；文献 [A.5] 等。

绪论

绪论（综述）：简要说明研究工作的目的、范围、相关领域的前人工作和知识空白、理论基础和分析，研究设想、研究方法和实验设计、预期结果和意

义等。一般在教科书中有的知识，在绪论中不必赘述。

绪论的内容应包括论文研究方向相关领域的最新进展、对有关进展和问题的评价、本论文研究的命题和技术路线等；绪论应表明博士生对研究方向相关的学科领域有系统深入的了解，论文具有先进性和前沿性；

问题 A-1 测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。

为了反映出作者确已掌握了坚实的基础理论和系统的专门知识，具有开阔的科学视野，对研究方案作了充分论证，绪论应单独成章，列为第一章，绪论的篇幅应达 1 ~ 2 万字，不得少于 1 万字；绪论引用的文献应在 100 篇以上，其中外文文献不少于 60%；引用文献应按正文中引用的先后排列。

正文

论文的正文是核心部分，占主要篇幅。正文必须实事求是，客观真切，准确完备，合乎逻辑，层次分明，简便可读。

图 A-2: 测试附录中的插图

正文的每一章 (除绪论外) 应有小结，在小结中应明确阐明作者在本章中所做的工作，特别是创新性成果。凡本论文要用的基础性内容或他人的成果不应单独成章，也不应作过多的阐述，一般只引结论、使用条件等，不作推导。

图

图包括曲线图、构造图、示意图、图解、框图、流程图、记录图、布置图、地图、照片、图版等。

图应具有“自明性”，即只看图、图题和图例，不阅读正文，就可以理解图意。

图应编排序号。每一图应有简短确切的图题，连同图号置于图下。必要时，应将图上的符号、标记、代码，以及实验条件等，用最简练的文字，横排

于图题下方，作为图例说明。

例 A-1 测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。测试定理环境。

曲线图的纵、横坐标必须标注“量、标准规定符号、单位”。此三者只有在不必要标明 (如无量纲等) 的情况下方可省略。坐标上标注的量的符号和缩略词必须与正文一致。

照片图要求主题和主要显示部分的轮廓鲜明，便于制版。如用放大缩小的复制品，必须清晰，反差适中。照片上应该有表示目的物尺寸的标度。

表

表的编排，一般是内容和测试项目由左至右横读，数据依序竖排。表应有自明性。

表应编排序号。

每一表应有简短确切的表题，连同标号置于表上。必要时，应将表中的符号、标记、代码，以及需要说明事项，以最简练的文字，横排于表题下，作为表注，也可以附注于表下。表内附注的序号宜用小号阿拉伯数字并加圆括号置于被标注对象的右上角，如：xxx[1)]；不宜用“*”，以免与数学上共轭和物质转移的符号相混。

表的各栏均应标明“量或测试项目、标准规定符号、单位”。只有在无必要标注的情况下方可省略。表中的缩略词和符号，必须与正文中一致。

表内同一栏的数字必须上下对齐。表内不宜用“同上”，“同左”和类似词，一律填入具体数字或文字。表内“空白”代表未测或无此项，“—”或“...”（因“—”可能与代表阴性反应相混）代表未发现，“0”代表实测结果确为零。

如数据已绘成曲线图，可不再列表。

数学、物理和化学式

正文中的公式、算式或方程式等应编排序号，序号标注于该式所在行 (当有续行时，应标注于最后一行) 的最右边。

较长的式，另行居中横排。如式必须转行时，只能在 +，−，×，÷，<，> 处转行。上下式尽可能在等号“=”处对齐。

小数点用“.”表示。大于 999 的整数和多于三位数的小数，一律用半个阿拉伯数字符的小间隔分开，不用千位撇。对于纯小数应将 0 列于小数点之前。

示例：应该写成 94 652.023 567 和 0.314 325, 不应写成 94,652.023,567 和 .314,325。

应注意区别各种字符，如：拉丁文、希腊文、俄文、德文花体、草体；罗马数字和阿拉伯数字；字符的正斜体、黑白体、大小写、上下脚标（特别是多层次，如“三踏步”）、上下偏差等。

计量单位

报告、论文必须采用国务院发布的《中华人民共和国法定计量单位》，并遵照《中华人民共和国法定计量单位使用方法》执行。使用各种量、单位和符号，必须遵循附录 B 所列国家标准的规定执行。单位名称和符号的书写方式一律采用国际通用符号。

符号和缩略词

符号和缩略词应遵照国家标准的有关规定执行。如无标准可循，可采纳本学科或本专业的权威性机构或学术团体所公布的规定；也可以采用全国自然科学名词审定委员会编印的各学科词汇的用词。如不得不引用某些不是公知公用的、且又不易为同行读者所理解的、或系作者自定的符号、记号、缩略词、首字母缩写字等时，均应在第一次出现时一一加以说明，给以明确的定义。

结论

报告、论文的结论是最终的、总体的结论，不是正文中各段的小结的简单重复。结论应该准确、完整、明确、精炼。在结论中要清楚地阐明论文中有那些自己完成的成果，特别是创新性成果；

如果不可能导出应有的结论，也可以没有结论而进行必要的讨论。可以在结论或讨论中提出建议、研究设想、仪器设备改进意见、尚待解决的问题等。

致谢

可以在正文后对下列方面致谢：

- 国家科学基金、资助研究工作的奖学金基金、合作单位、资助或支持的企业、组织或个人;
- 协助完成研究工作和提供便利条件的组织或个人;
- 在研究工作中提出建议和提供帮助的人;
- 给予转载和引用权的资料、图片、文献、研究思想和设想的所有者;
- 其他应感谢的组织或个人。

参考文献表

专著著录格式

主要责任者，其他责任者，书名，版本，出版地：出版者，出版年

例：1. 刘少奇，论共产党员的修养，修订 2 版，北京：人民出版社，1962

连续出版物中析出的文献著录格式

析出文献责任者，析出文献其他责任者，析出题名，原文献题名，版本：文献中的位置。

例：2. 李四光，地壳构造与地壳运动，中国科学，1973 (4)：400 － 429

参考文献采用顺序编码制，按论文正文所引用文献出现的先后顺序连续编码。

附录

附录是作为报告、论文主体的补充项目，并不是必需的。

下列内容可以作为附录编于报告、论文后，也可以另编成册;

1. 为了整篇论文材料的完整，但编入正文又有损于编排的条理和逻辑性，这一材料包括比正文更为详尽的信息、研究方法和技术更深入的叙述，建议可以阅读的参考文献题录，对了解正文内容有用的补充信息等;
2. 由于篇幅过大或取材于复制品而不便于编入正文的材料;
3. 不便于编入正文的罕见珍贵资料;
4. 对一般读者并非必要阅读，但对本专业同行有参考价值的资料;
5. 某些重要的原始数据、数学推导、计算程序、框图、结构图、注释、统计表、计算机打印输出件等。

附录与正文连续编页码。

每一附录均另页起。

结尾部分 (必要时)

为了将论文迅速存储入电子计算机，可以提供有关的输入数据。可以编排分类索引、著者索引、关键词索引等。

参考文献

[1] NEWMAN M, BARABÁSI A-L, WATTS D J. The structure and dynamics of networks[M]. Princeton: Princeton University Press, 2006.

[2] NEWMAN M E, STROGATZ S H, WATTS D J. Random graphs with arbitrary degree distributions and their applications[J]. Physical Review E, 2001, 64(2): 026118.

[3] AIELLO W, CHUNG F, LU L. A random graph model for massive graphs[C] //Proceedings of the thirty-second annual ACM symposium on Theory of computing. New York: ACM, 2000: 171–180.

[4] BOLLOBÁS B. Random graphs: Vol 73[M]. [S.l.]: Cambridge university press, 2001.

[5] BARABÁSI A-L, ALBERT R. Emergence of scaling in random networks[J]. science, 1999, 286(5439): 509–512.

[6] ERDŐS P, RENYI A. On the strength of connectedness of a random graph[J]. Acta Mathematica Hungarica, 1961, 12(1): 261–267.

[7] NVIDIA Corporation. Nvidia GRID[EB/OL]. 2013 [2013-03-10]. http://www.nvidia.com/object/cloud-gaming.html.

[8] MARCONI G. Wireless telegraphic communication[G] //Nobel Prize address. 1909.

[9] BARABÁSI A-L. Linked: How Everything is Connected to Everything Else and What It Means for Business, Science, and Everyday Life[M]. New York: Plume, 2003.

[10] de Sola Pool I, KOCHEN M. Contacts and influence[J]. Social Networks, 1978, 1: 5–51.

[11] LUCE R D, PERRY A D. A method of matrix analysis of group structure[J/OL]. Psychometrika, 1949, 14(2): 95 – 116.
http://dx.doi.org/10.1007/BF02289146.

[12] WATTS D, STROGATZ S. Collective dynamics of "small-world" networks[J/OL]. Nature, 1998, 393: 440 – 442.
http://dx.doi.org/10.1038/30918.

[13] GYARMATI L, Trinh T. Scafida: A scale-free network inspired data center architecture[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(5): 4 – 12.

[14] MOLLOY M, REED B. A critical point for random graphs with a given degree sequence[J]. Random structures & algorithms, 1995, 6(2-3): 161 – 180.

[15] FRIEZE A M, ŁUCZAK T. On the independence and chromatic numbers of random regular graphs[J]. Journal of Combinatorial Theory, Series B, 1992, 54(1): 123 – 132.

[16] BENDER E A, CANFIELD E R. The asymptotic number of labeled graphs with given degree sequences[J]. Journal of Combinatorial Theory, Series A, 1978, 24(3): 296 – 307.

[17] ERDÖS P, RÉNYI A. Additive properties of random sequences of positive integers[J]. Acta Arithmetica, 1960, 6(1): 83 – 110.

[18] VÁZQUEZ A, FLAMMINI A, MARITAN A, et al. Modeling of protein interaction networks[J]. Complexus, 2002, 1(1): 38 – 44.

[19] SOLÉ R V, PASTOR-SATORRAS R, SMITH E, et al. A model of large-scale proteome evolution[J]. Advances in Complex Systems, 2002, 5(01): 43 – 54.

[20] GOH K-I, OH E, JEONG H, et al. Classification of scale-free networks[J]. Proceedings of the National Academy of Sciences, 2002, 99(20): 12583 – 12588.

[21] ANTHONISSE J M. The rush in a graph[J]. Amsterdam: University of Amsterdam Mathematical Centre, 1971.

[22] FREEMAN L C. A set of measures of centrality based on betweenness[J]. Sociometry, 1977: 35–41.

[23] GOH K-I, KAHNG B, KIM D. Universal behavior of load distribution in scale-free networks[J]. Physical Review Letters, 2001, 87(27): 278701.

[24] ADAMIC L A, HUBERMAN B A. Power-law distribution of the world wide web[J]. Science, 2000, 287(5461): 2115–2115.

[25] BIANCONI G, BARABÁSI A-L. Competition and multiscaling in evolving networks[J]. EPL (Europhysics Letters), 2001, 54(4): 436.

[26] KRAPIVSKY P L, REDNER S, LEYVRAZ F. Connectivity of growing random networks[J]. Physical review letters, 2000, 85(21): 4629.

[27] DOROGOVTSEV S N, MENDES J F F, SAMUKHIN A N. Structure of growing networks with preferential linking[J]. Physical Review Letters, 2000, 85(21): 4633.

[28] BARABÁSI A-L, ALBERT R, JEONG H. Scale-free characteristics of random networks: the topology of the world-wide web[J]. Physica A: Statistical Mechanics and its Applications, 2000, 281(1): 69–77.

[29] PRICE D. Statistical studies of networks of scientific papers[C] // Statistical Association Methods for Mechanized Documentation: Symposium Proceedings: Vol 269. 1965: 187.

[30] REDNER S. How popular is your paper? An empirical study of the citation distribution[J]. The European Physical Journal B-Condensed Matter and Complex Systems, 1998, 4(2): 131–134.

[31] MERTON R K. The Matthew effect in science[J]. Science, 1968, 159(3810): 56–63.

[32] GILBERT E N. Random Graphs[J/OL]. Annals of Mathematical Statistics, 1959, 30(4): 1141–1144.
http://dx.doi.org/10.1214/aoms/1177706098.

[33] ERDŐS P, RÉNYI A. On Random Graphs. I[J/OL]. Publicationes Mathematicae, 1959, 6: 290 – 297.
http://www.renyi.hu/~p_erdos/1959-11.pdf.

[34] LI L, ALDERSON D, DOYLE J C, et al. Towards a theory of scale-free graphs: Definition, properties, and implications[J]. Internet Mathematics, 2005, 2(4): 431 – 523.

[35] SOLOMONOFF R, RAPOPORT A. Connectivity of random nets[J/OL]. The bulletin of mathematical biophysics, 1951, 13(2): 107 – 117.
http://dx.doi.org/10.1007/BF02478357.

[36] ALBERT R, JEONG H, BARABÁSI A-L. The diameter of the world wide web[J/OL], 1999, 401: 130 – 131.
http://dx.doi.org/10.1038/43601.

[37] WANG X F, CHEN G. Complex networks: small-world, scale-free and beyond[J/OL]. Circuits and Systems Magazine, IEEE, 2003, 3(1): 6 – 20.
http://dx.doi.org/10.1109/MCAS.2003.1228503.

[38] LESKOVEC J, HORVITZ E. Planetary-Scale Views on a Large Instant-Messaging Network[C] //Proceedings of the 17th international conference on World Wide Web. New York: ACM, 2008.

[39] LESKOVEC J, HORVITZ E. Worldwide Buzz: Planetary-Scale Views on an Instant-Messaging Network[R]. [S.l.]: Microsoft Research, 2007.

[40] FASS C, TURTLE B, GINELLI M. Six Degrees of Kevin Bacon[M]. New York City: Plume, 1996.

[41] REYNOLDS P. The Oracle of Bacon[EB/OL]. [2011-07-12].
http://oracleofbacon.org/cgi-bin/center-cgi?who=Kevin+Bacon.

[42] NEWMAN M. The structure of scientific collaboration networks[J/OL]. Proceedings of the National Academy of Sciences, 2001, 98(2): 404 – 409.
http://dx.doi.org/10.1073/pnas.98.2.404.

[43] GROSSMAN J. Publications of Paul Erdős[EB/OL]. [2011-02-01]. http://www.oakland.edu/enp/pubinfo/.

[44] UNIVERSITY O. The Erdős Number Project Data Files[EB/OL]. 2009 [2010-05-29]. http://www.oakland.edu/enp/thedata/.

[45] GOFFMAN C. And what is your Erdős number?[J]. American Mathematical Monthly, 1969, 76(7): 791.

[46] MILGRAM S. The Small World Problem[J]. Psychology Today, 1967, 2: 60–67.

[47] TRAVERS J, MILGRAM S. An Experimental Study of the Small World Problem[J]. Sociometry, 1969, 32(4): 425–443.

[48] GUREVICH M. The Social Structure of Acquaintanceship Networks[D]. Cambridge, MA: Massachusetts Institute of Technology, 1961.

[49] HERRERA C, ZUFIRIA P J. Generating scale-free networks with adjustable clustering coefficient via random walks[C] //Network Science Workshop (NSW), 2011 IEEE. 2011: 167–172.

[50] GREENBERG A, LAHIRI P, MALTZ D A, et al. Towards a next generation data center architecture: scalability and commoditization[C] //Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow. 2008: 57–62.

[51] LEIGHTON F T. Introduction to parallel algorithms and architectures[M]. [S.l.]: Morgan Kaufmann San Francisco, 1992.

[52] GUO C, LU G, LI D, et al. BCube: a high performance, server-centric network architecture for modular data centers[J]. ACM SIGCOMM Computer Communication Review, 2009, 39(4): 63–74.

[53] GUO C, WU H, TAN K, et al. Dcell: a scalable and fault-tolerant network structure for data centers[C] //ACM SIGCOMM Computer Communication Review: Vol 38. 2008: 75–86.

[54] LOCKWOOD J W, MCKEOWN N, WATSON G, et al. NetFPGA–an open platform for gigabit-rate network switching and routing[C] // Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on. 2007 : 160 – 161.

[55] SourceForge. VDE: Virtual Distributed Ethernet[CP/OL]. 2013 [2013-03-09]. https://vde.sourceforge.net.

[56] Quagga Routing Suite[EB/OL]. [2013-03-09]. http://www.nongnu.org/quagga/.

[57] KOHLER E, MORRIS R, CHEN B, et al. The Click modular router[J]. ACM Transactions on Computer Systems (TOCS), 2000, 18(3) : 263 – 297.

[58] HAN S, JANG K, PARK K, et al. PacketShader: a GPU-accelerated software router[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4) : 195 – 206.

[59] DOBRESCU M, EGI N, ARGYRAKI K, et al. RouteBricks: exploiting parallelism to scale software routers[C] // Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. 2009 : 15 – 28.

[60] HU Fei, QIU Meikang, LI Jiayin, et al. A Review on cloud computing: Design challenges in Architecture and Security[J]. Journal of Computing and Information Technology, 2011, 19(1) : 25 – 55.

[61] VARIA J. Architecting for the cloud: Best practices[R]. [S.l.] : Amazon Web Services, 2010.

[62] GREENBERG A, HAMILTON J R, JAIN N, et al. VL2: a scalable and flexible data center network[C] // ACM SIGCOMM Computer Communication Review : Vol 39. 2009 : 51 – 62.

[63] AL-FARES M, LOUKISSAS A, VAHDAT A. A scalable, commodity data center network architecture[C] // ACM SIGCOMM Computer Communication Review : Vol 38. 2008 : 63 – 74.

[64] PALLIS G, VAKALI A. Insight and perspectives for content delivery networks[J]. Communications of the ACM, 2006, 49(1): 101–106.

[65] DEAN J. Designs, Lessons, and Advice from Building Large Distributed Systems.[J]. Keynote from LADIS 2009, 2009.

[66] BRODER A, KUMAR R, MAGHOUL F, et al. Graph structure in the web[J]. Computer networks, 2000, 33(1): 309–320.

[67] KLEINBERG J M. Authoritative sources in a hyperlinked environment[J]. Journal of the ACM (JACM), 1999, 46(5): 604–632.

[68] KLEINBERG J. The Small-World Phenomenon: An Algorithmic Perspective[C] //in Proceedings of the 32nd ACM Symposium on Theory of Computing. 2000: 163–170.

[69] MILGRAM S. The Small World Problem[J]. Psychology Today, 1967, 2: 60–67.

[70] Wikipedia contributors. Moore's law[EB/OL]. Wikipedia, The Free Encyclopedia, 2015 (2015/06/14) [2015/06/15].
https://en.wikipedia.org/wiki/Moore%27s_law.

[71] DUBASH M. Moore's Law is dead, says Gordon Moore[EB/OL]. Techworld, 2010 (2010/4/13) [2015/6/16].
http://www.techworld.com/news/operating-systems/moores-law-is-dead-says-gordon-moore-3576581/.

[72] KANELLOS M. Intel scientists find wall for Moore's Law[EB/OL]. CNET, 2003 (2003/12/1) [2015/6/16].
http://news.cnet.com/2100-1008-5112061.html.

[73] Intel Corporation. Intel discloses newest microarchitecture and 14 nanometer manufacturing process technical details[EB/OL]. Intel Corporation, 2014 (2014/8/11) [2015/6/16].
http://newsroom.intel.com/community/intel_newsroom/blog/2014/08/11/intel-discloses-newest-microarchitecture-and-14-nanometer-manufacturing-process-technical-details.

[74] MOAMMER K. TSMC Launching 10nm FinFET Process In 2016, 7nm In 2017[EB/OL]. WCCF Tech, 2015 (2015/4/19) [2015/6/16]. http://wccftech.com/tsmc-promises-10nm-production-2016-7nm-2017/.

[75] SHROUT R. Intel Xeon E5-2600 v3 processor overview: Haswell-EP up to 18 cores[EB/OL]. 2014 (2014/9/8) [2015/6/16]. http://www.pcper.com/reviews/Processors/Intel-Xeon-E5-2600-v3-Processor-Overview-Haswell-EP-18-Cores.

[76] Intel Corporation. Intel chips timeline[EB/OL]. Intel Corporation, 2012 (2012/7/13) [2015/6/16]. http://www.intel.co.uk/content/www/uk/en/history/history-intel-chips-timeline-poster.html.

[77] Wikipedia contributors. Transistor count[EB/OL]. Wikipedia, The Free Encyclopedia, 2015 (2015/6/10) [2015/6/16]. https://en.wikipedia.org/wiki/Transistor_count.

图论基础知识

简历与科研成果

基本信息

韦小宝，男，汉族，1985 年 11 月出生，江苏省扬州人。

教育背景

2007 年 9 月 — 2010 年 6 月 南京大学计算机科学与技术系 硕士
2003 年 9 月 — 2007 年 6 月 南京大学计算机科学与技术系 本科

攻读硕士学位期间完成的学术成果

1. Xiaobao Wei, Jinnan Chen, "Voting-on-Grid Clustering for Secure Localization in Wireless Sensor Networks," in Proc. IEEE International Conference on Communications (ICC) 2010, May. 2010.
2. Xiaobao Wei, Shiba Mao, Jinnan Chen, "Protecting Source Location Privacy in Wireless Sensor Networks with Data Aggregation," in Proc. 6th International Conference on Ubiquitous Intelligence and Computing (UIC) 2009, Oct. 2009.

攻读硕士学位期间参与的科研课题

1. 国家自然科学基金面上项目 "无线传感器网络在知识获取过程中的若干安全问题研究" （课题年限 2010 年 1 月 — 2012 年 12 月），负责位置相关安全问题的研究。
2. 江苏省知识创新工程重要方向项目下属课题 "下一代移动通信安全机制研究" （课题年限 2010 年 1 月 — 2010 年 12 月），负责 LTE/SAE 认证相关的安全问题研究。

学位论文出版授权书

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名：_____

_____年_____月_____日

论文题名	数据中心网络模型研究				
研究生学号	MG1533012	所在院系	计算机科学与技术系	学位年度	2015
论文级别	☑ 硕士 ☐ 硕士专业学位 ☐ 博士 ☐ 博士专业学位 (请在方框内画勾)				
作者电话	13951785456		作者 Email	xingkungao@163.com	
第一导师姓名	唐杰 副教授		导师电话	13813950617	

论文涉密情况：

☐ 不保密

☑ 保密，保密期：_____年_____月_____日 至 _____年_____月_____日

注：请将该授权书填写后装订在学位论文最后一页（南大封面）。