计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20160568
53(12):2729-2752,2016

基于 GraphX 的传球网络构建及分析研究

张 陶[1] 于 炯[1] 廖 彬[2] 国冰磊[1] 卞 琛[1] 王跃飞[1] 刘 炎[3]
[1](新疆大学信息科学与工程学院 乌鲁木齐 830046)
[2](新疆财经大学统计与信息学院 乌鲁木齐 830012)
[3](清华大学软件学院 北京 100084)
(zt59921661@126.com)

The Construction and Analysis of Pass Network Graph Based on GraphX

Zhang Tao[1], Yu Jiong[1], Liao Bin[2], Guo Binglei[1], Bian Chen[1], Wang Yuefei[1], and Liu Yan[3]
[1](*School of Information Science and Engineering, Xinjiang University, Urumqi* 830046)
[2](*School of Statistics and Information, Xinjiang University of Finance and Economics, Urumqi* 830012)
[3](*School of Software, Tsinghua University, Beijing* 100084)

Abstract In the field of social networking, finance, public security, health care, etc, the application of big data technology is matured constantly, but its application in competitive sports is still in exploratory stage. Lacking of recording the pass data in basketball technical statistics leads that we can not research the statistical analysis, data mining and application on the pass data. Firstly, as the aggregation from of passing data is graph, based on data acquisition, clean and format conversion, Vertex and Edge table construction, we create the pass network graph with GraphX, which lays the foundation for other applications. Secondly, the PlayerRank algorithm is proposed to distinguish the importance of players, player position personalized the graph vertex's color, etc, which improves the visual quality of pass network graph. Finally, we can use the pass network graph created by GraphX to analyze the effect of passing quantity and quality on the outcome of the game, and the pass network graph is also used to analyze the team's passing data, tactical player selection, on-the-spot tactics supporting, subgraph extraction and gaming experience improvement, etc.

Key words big data application; pass network; GraphX frame; PlayerRank algorithm; player importance

摘 要 虽然大数据技术在社交网络、金融、公共安全、医疗卫生等领域的应用不断成熟,但在竞技体育方面的应用还处于探索阶段.常规篮球统计中缺乏对传球数据的记录,更缺乏对传球数据的统计分析、价值挖掘及应用等方面的研究.1)由于传球数据汇聚形态为图,在传球数据获取、数据清洗及格式转化、Vertex 与 Edge 表构建的基础上,通过 GraphX 构建传球网络图为其应用打下基础;2)提出 PlayerRank 值区分球员重要度、球员位置个性化图顶点等方法提高传球网络可视化质量;3)通过 GraphX 构建的传球网络分析传球数量与质量对比赛结果的影响,并例举了传球网络在球队传球数据分析、战术人员选择、临场战术制定、网络子图及游戏体验改进等方面的应用.

收稿日期:2016-08-08;**修回日期**:2016-10-24
基金项目:国家自然科学基金项目(61562078,61262088,71261025);新疆维吾尔自治区自然科学基金项目(2016D01B014)
This work was supported by the National Natural Science Foundation of China (61562078, 61262088, 71261025) and the Natural Science Foundation of the Xinjiang Uygur Autonomous Region of China (2016D01B014).
通信作者:廖彬(liaobin665@163.com)

关键词 大数据应用;传球网络;GraphX 框架;PlayerRank 算法;球员重要性

中图法分类号 TP393.09

据互联网数据中心(Internet data center)发布的报告显示,2015 年全球产生的数据量达到近 10 ZB,而 2020 年全球产生的数据量将达到 40 ZB[1]. 数据的产生过程在经历被动和主动 2 种产生过程后,发展到了自动产生阶段,预示着大数据时代的来临. 数据从简单的处理对象开始转变为一种基础性资源,如何更好地管理和利用大数据已经成为普遍关注的话题,大数据的规模效应给数据存储、管理以及数据分析带来了极大的挑战[2]. 在 Victor 的大数据理论中,大数据最核心的问题并不是数据的种类(variety)及量(volume),而是大数据的价值(value). 大数据时代并不代表所有应用数据量都大,大数据也是由一个个小数据集合而成,正是对小数据的持续采集、融合分析,才有积跬步而致千里的大数据价值能量的爆发. 自 2003 年 Google 发表论文公开分布式存储系统 GFS[3](Google file system)及分布式数据处理模型 MapReduce[4] 以来,诸多的大数据计算系统及框架(如 Hadoop, Storm, Spark[5], Pig, Hive, Hbase, Dryad 等)以 MapReduce 为计算模型,并形成了以 MapReduce 为核心的大数据计算生态系统,如图 1 所示:

Fig. 1 MapReduce computing model ecosystem.

图 1 MapReduce 计算模型生态系统

随着大数据技术的不断成熟,其在社交网络、金融、公共安全、医疗等方面的应用也不断发展成熟;但是大数据技术在竞技体育方面的应用,还处于探索阶段. 大数据技术在足球方面的应用首次亮相是在 2014 年的巴西世界杯,帮助德国再次捧得大力神杯的"秘密武器"之一,则是来自 SAP 公司的足球大数据技术解决方案 Match Insights. Match Insights 能够迅速收集、处理分析球员和球队的技术数据,基于大数据分析优化球队配置,提升球队作战能力,并通过分析对手技术数据,找到在世界杯比赛中的"制敌"方式. 利用大数据分析,德国队教练可以迅速了解当前比赛的状况、每个球员的特点和表现、球员的防守范围、对方球队的空挡区等信息. 通过这些信息,教练可以更有效地对球员上场时间、位置、技战术等情况优化配置,以提升球员及球队的整体表现.

在篮球比赛中,常规的技术统计有:得分、篮板、助攻、抢断、盖帽、失误、犯规、投篮命中率、出场时间等;但缺乏对传球数据的记录,更缺乏对传球数据的统计分析、数据挖掘及应用方法的研究. 导致基于已有的技术统计数据,无法回答 5 个与传球有关的问题:

问题 1. 某球员这场比赛传球多少次?接球多少次?传球质量怎样?传球助攻率多少?

问题 2. 传切战术、挡拆战术及三角进攻战术中,由谁来传球最好?

问题 3. 比赛还剩 5 s,落后 2 分,谁来执行绝杀投篮?谁来传球?

问题 4. 林书豪传球质量比库里好?还是差?好多少?差多少?

问题 5. NBA 中传球质量最好的是哪个队?传球最频繁的是哪个队?

以上仅仅例举了 5 个较为常见的问题,而基于已有的技术统计,无法回答或解决的问题远远不止这些. 因此,为了解决这 5 个问题,本文将球员之间的传球关系进行关联(将传球人与接球人作为顶点,顶点之间的传球关系作为边),发现随着传球数据量的不断增加,最终形成一张稳定的传球网络图. 与 FaceBook、微博等社会关系网络最大不同的是,社会网络中边的属性较为简单(通常为好友或关注关系),而传球网络中的边属性(边属性包括传球次数、助攻次数、传球概率、投篮命中率等)则复杂得多. 由于传球数据汇聚的顶层模型为图,所以本文选取基于 Spark[5] 的图分析工具 GraphX[6-7] 构建传球网络图,并在此基础上分析 NBA 球员之间的传球数据,充分挖掘传球数据的内涵,为球员的训练、战术制定、对手分析、教练决策等提供支持.

1 相关研究及背景

1.1 图的相关研究

早期经典的图理论研究工作,如 AGM[8], FSM[9], GSAPN[10], FFSM[11] 等为图应用系统的开发提供了

理论基础.随着 Hadoop 的快速发展,MapReduce 计算模型得到广泛应用,其中 FSM-H[12] 及 MRFSM[13] 就是基于 MapReduce 框架的图算法,由于 MapReduce 的优势在于处理批处理作业,对于具有复杂业务处理逻辑的图计算,MapReduce 计算效率并不理想[14-15].在此背景下,基于内存计算的分布式图计算框架得到了快速发展(如 GraphX 是基于 Spark 的图计算框架).分布式的图计算框架是将对图的操作(如图的构建、PR 计算、最短路径查找等)封装为接口,使得图的分布式存储及计算等复杂问题对上层应用透明.图计算框架能够让图算法及图应用工程师忽略图底层细节(如图顶点、边及其相关属性的分布式存储及计算方法),将精力聚集到具体的图相关模型设计和应用层面上来.

当前的图计算框架基本上都遵循 BSP(bulk synchronous parallell)[16-17] 计算模式.在 BSP 中,一次计算过程由一系列全局超步组成,每一个超步由并发计算、通信和栅栏同步 3 个步骤组成,同步完成标志着这个超步的完成及下一个超步的开始.基于 BSP 模式,目前较为成熟的图计算模型主要有 Pregel[18-20] 及 GAS[21-22] 两种.其中 Pregel 模型来自于 Google,借鉴 MapReduce 的思想,Pregel 提出了"像顶点一样思考"(think like a vertex)的图计算模式,让用户无需考虑并行分布式计算的细节,只需要实现一个顶点更新函数,让框架在遍历顶点时进行调用即可.但是对于邻居数较多的顶点,Pregel 模型需要处理的消息非常庞大,并且它们是无法被并发处理的.所以对于符合幂律分布的自然图,Pregel 模型容易出现应用假死或者崩溃的现象.相比 Pregel 模型的消息通信范式,GraphLab[23] 提出的 GAS 模型更偏向共享内存风格.它允许用户的自定义函数访问当前顶点的整个邻域,可抽象成 Gather,Apply,Scatter 三个阶段(GAS),与此对应,用户需要实现与 GAS 所对应的函数 gather,apply,scatter.正是由于 gather 和 scatter 以单条边为计算粒度,所以对于顶点众多的邻边,可以分别由相应的计算节点独立调用 gather 和 scatter,从而较好地解决了 Pregel 中存在的问题.

1.2　相关背景知识介绍

1.2.1　基于 Spark 的分布式图框架 GraphX

GraphX 基于 Spark 并扩展了 Spark RDD[24](resilient distributed datasets)弹性分布式数据集的抽象,提出了 Resilient Distributed Property Graph[6](点和边都带属性的有向多重图).由于 GraphX 提供 Table 与 Graph 两种视图,并且这 2 种视图都拥有自己独立的操作符,这使得 GraphX 的操作具有更大的灵活性.GraphX 对 Pregel 及 GAS 模型的改进的同时性能也是优于 GraphLab[23] 和 Giraph[25].GraphX 不仅提供节点度(出度/入度)的计算、子图查询、PageRank、最大连通图及最短路径等基本图算法,并且能够无缝地调用 SparkCore 中的 API 接口(包括 map,filter,flatMap,sample,groupByKey,reduceByKey,union,join,cogroup,mapValues,sort,partionBy 等).虽然基于 GraphX 的应用并不成熟,但是从运行效率及编程模型支撑的角度考虑,运用 GraphX 进行 NBA 传球网络数据的分析是一个较为理想的选择.

1.2.2　相关篮球术语介绍

为了方便读者对本文的阅读及理解,表 1 对本文中涉及到篮球方面的一些专业术语或专有名词(如 NBA、NBA 球队、NBA 赛制、技术统计等)进行解释.

Table 1　Basic Professional Terminology

表 1　基础专业术语

Terminology	Description
NBA	The National Basketball Association (NBA) is the pre-eminent men's professional basketball league in North America, and is widely considered to be the premier men's professional basketball league in the world.
CBA	Chinese Basketball Association is a national non-profit sports organisation in China.
NBA Team	The NBA consists of 30 teams, with the United States is home to 29 teams and one is located in Canada. The current league organization divides thirty teams into two conferences of three divisions with five teams each.
NBA Rules	NBA schedule is divided into 3 stages, the preseason, regular season and the playoffs. The preseason only 5 games, it's main purpose is training, which are not included in the normal statistics. The regular season has 82 games, after the regular season, the East and West League top 8 playoff team to the playoffs, take 7 games 4 wins knockout, regular season winning team won a higher rate of home court advantage, and then by the East and West of the champions of the NBA finals, decide the NBA championship.

Continued (Table 1)

Terminology	Description
Basketball Technical Statistics	Basketball statistics include: points, rebounds, assists, steals, blocks, fouls drawn, turnovers, etc.
Regular Season MVP	The National Basketball Association Most Valuable Player (MVP) is an annual National Basketball Association (NBA) award given since the 1955—1956 NBA season to the best performing player of the regular season.
Final MVP	NBA Finals Most Valuable Player Award is an annual National Basketball Association (NBA) award given since the 1969 NBA Finals. The award is decided by a panel of nine media members, who cast votes after the conclusion of the Finals. The person with the highest votes wins the award.

1.3 本文工作与已有工作的不同

本文研究范畴属于图的应用,图广泛应用于社交网络,包括用户网络的社区发现、用户影响力、能量传播、标签传播等,用以提升用户黏性和活跃度;而应用到推荐领域时,标签推理、人群划分、年龄段预测等可以提升推荐的丰富度和准确性.除社交网络外,图还在生物信息学、化学信息学、智能交通、舆情监控等等领域发挥着巨大的作用.由于 GraphX 于 Spark 版本 1.2 之后才发布正式版,导致基于 GraphX 的算法及应用方面的工作较少,并且,其应用场景通常在互联网领域,如淘宝基于 GraphX 搭建了图谱平台,可以支撑多图合并、能量传播模型计算、用户影响力计算、商品推荐等算法.在国内,严玉良等人在文献[26]中提出一种基于 GraphX 的大规模频繁子图挖掘算法 FSMBUS,使得算法性能较 GRAMI 提高一个量级.

在篮球数据研究方面,文献[27]提出 EPV (expected possession value)的概念,将球员场上的行为(如传球、运球、投篮等)转化为动态的 EPV 值,以此量化球员在球场上每次动作的价值;文献[28]关注篮板球数据,将每次篮板数据的产生分解为 Positioning, Hustle, Conversion 三个阶段,并基于篮板球数据对 NBA 中球员的篮板球能力进行了分析;文献[29]对球员加速行为进行了量化研究,并对球员的加速数据进行了可视化;文献[30]对 NBA 中的投篮选择(2 分球与 3 分球)与对应的风险进行了研究,分析了不同场景下的投篮选择对比赛结果的影响;文献[31]提出了通过防守矩阵(defensive metrics)分析并量化球员防守有效性的方法;文献[32]对球员在投篮结束后的冲抢前场篮板与退位防守的选择性问题进行了研究,研究结果表明退位防守能够提高防守成功率,但减少了抢下前场篮板球的数量,文中还针对不同球队的数据进行了分析,并提出相应的改进策略.

本文与已有研究工作不同的是,本文发现常规的篮球比赛统计中缺乏对传球数据的统计,更缺乏对传球数据的统计分析、价值挖掘及应用等方面的研究.本文将球员之间的传球关系进行关联(将传球人与接球人作为顶点,顶点之间的传球关系作为边),发现随着传球数据量的不断增加,最终形成一张稳定的传球网络图.并且,本文将 GraphX 应用于 NBA 传球数据的分析,通过建立球员之间的传球网络,充分挖掘传球数据的内涵,为球员的训练、战术制定、对手分析、教练决策、游戏体验改进等应用提供支撑.

2 传球网络图的构建

2.1 图的构建总体流程

传球网络图的构建及应用框架流程图如图 2 所示,主要分为 6 个步骤:

1) 原始数据获取.球队及球员的传球数据一般可通过 NBA 官方网站或通过具体场次的录像分析提取数据.

2) 数据清洗及格式转化.通过程序抓取而来的原始数据一般为 XML 或 JSON 格式,需通过数据清洗及格式转化,以满足网络图的构建对数据格式的要求.

3) 定义并构建 Vertex 表及 Edge 表.首先需根据后期应用的需求确定 Vertex 表及 Edge 表中的属性,并通过步骤 1 与步骤 2 确定对应属性数据是否可获得或者是否可计算.

4) Vertex 及 Edge 数据的持久化.由于 Vertex 表及 Edge 表为关系表,所以可选择 MySQL, PostgreSQL, ORACLE 等关系型数据库存储顶点及边的数据.

5) 传球网络的构建及可视化.在步骤 3 的基础上,通过 GraphX 构建出传球网络图,通过提取并计算图相关的数据(如传球次数、PR 值等)并结合大数据可视化技术,可实现传球网络图的可视化.

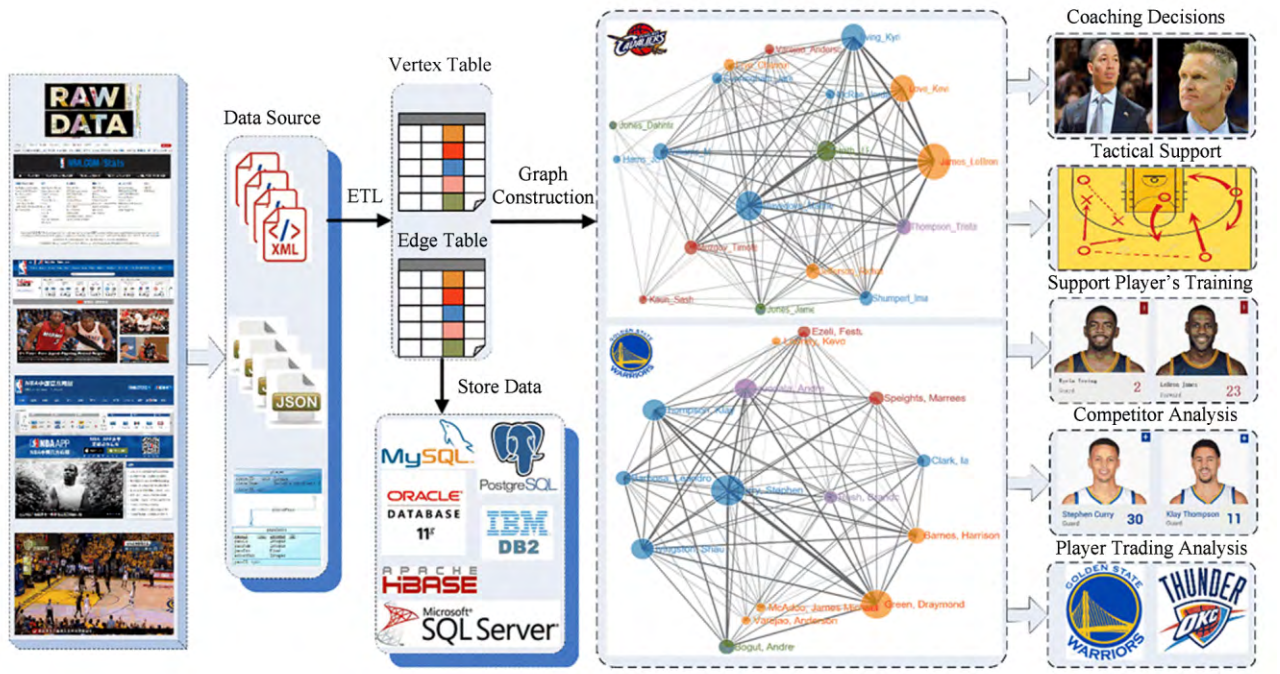


Fig. 2　The construction and application framework of passing network diagram.

图 2　传球网络图构建及应用框架流程图

6）基于传球网络的应用:传球网络可辅助教练的决策、战术支撑、辅助球员的训练、对手分析、球员交易分析等.具体的应用场景分析请参见第 4 节.

下文将围绕上述 6 个步骤进行详细介绍,其中 2.2 节围绕步骤 1、步骤 2 的内容进行了详细的阐述; 2.3 节讨论定义并构建 Vertex 表及 Edge 表的问题;2.4 节介绍传球网络的构建方法. 步骤 4 在步骤 2、步骤 3 的基础上通过调用 SparkSQL 及对应数据库的 API 便可完成. 而第 3 节阐述传球网络的可视化问题;第 4 节围绕传球网络的应用场景进行分析.

2.2　传球数据的获取、清洗及格式转化

篮球比赛中,常规的技术统计,如得分、篮板、助攻、抢断、盖帽、失误、犯规、投篮命中率、出场时间等,在比赛过程中会被技术台工作人员记录下来.比赛结束后,球迷可通过各大体育网站或论坛查询常规的技术统计数据.但是,对于传球相关的数据,如 Stephen Curry 在总决赛的第 1 场比赛中有多少次传球、多少次传给了 Klay Thompson、传球的助攻转化率等并不属于常规的技术统计,加之比赛过程中传球数据产生速度较快,很难通过现场人工的方法进行记录.所以,现阶段得到传球数据的最好方法,就是通过观看比赛录像人工地对传球数据进行记录,但是该方法的弊端就是工作量巨大.

NBA 作为世界上顶级篮球联盟,其官方网站[①]于 2016 年开始公开包括传球、防守影响、移动速度及距离等非常规技术统计.但是,中国的 CBA 及国家篮球队,对于传球数据的获取、统计分析及围绕传球数据的战术、训练等方面的运用还是空白.虽然 NBA 官方网站提供传球数据的查询功能,但是并没有提供转存功能,通过手工收集整理数据效率太低,本文以金州勇士队为例,传球数据获取流程如图 3 所示.

如图 3 所示为通过本文编写的数据下载程序从 NBA 官网获取金州勇士队传球数据的流程.流程步骤如下:

1）收集金州勇士队中每个队员在 NBA 数据库中所对应的 *playerid*,如 Stephen Curry 的 *playerid*=201939,Klay Thompson 的 *playerid*=202691.

2）将步骤 1 中所收集到的 *playerid* 形成 *playerids* 数组,该数组中存放着勇士队中所有队员的 *playerid*.

3）通过调用 CURL API 功能,通过拼接 URL[②],

① http://www.nba.com

② http://stats.nba.com/stats/playerdashptpass

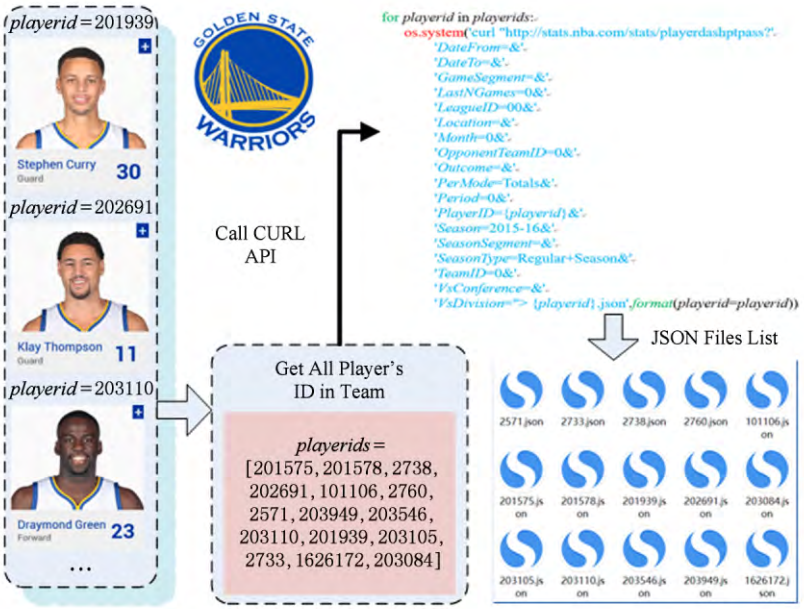


Fig. 3 Data acquisition process.

图 3 数据获取流程

利用 HTTP Get 方法得到返回的数据，生成当前 *playerid* 所对应的 JSON 文件. CURL 的 Python 调用如核心代码 1 所示：

核心代码 1. CURL Python 调用代码.

```
for playerid in playerids:
    os.system('curl "http://stats.nba.com/stats/
        playerdashptpass?'
    'DateFrom=&'
    'DateTo=&'
    'GameSegment=&'
    'LastNGames=0&'
    'LeagueID=00&'
    'Location=&'
    'Month=0&'
    'OpponentTeamID=0&'
    'Outcome=&'
    'PerMode=Totals&'
    'Period=0&'
    'PlayerID={playerid}&'
    'Season=2015-16&'
    'SeasonSegment=&'
    'SeasonType=Regular+Season&'
    'TeamID=0&'
    'VsConference=&'
    'VsDivision="> {playerid}.json'.
        format(playerid=playerid))
```

4) 循环 *playerids* 数组中所有的 *playerid*，重复调用步骤 3 中的 Python 代码，直到执行完 *playerids* 数组中所有的元素. 最终生成如图 3 所示的 JSON 文件集合，每个 JSON 文件对应一个球员的传球数据.

特别地指出：在步骤 3 的代码中，可修改具体的参数实现不同传球数据的获取，如参数 *DateFrom* 表示获取传球数据所对应的比赛开始时间，具体的参数及解释如表 2 所示：

Table 2 Interpretation of Parameters

表 2 参数含义解释

Parameter	Default Value	Description
DateFrom	NULL	Game Start Date Time
DateTo	NULL	Game End Date Time
GameSegment	Entrie Game	Game Time Segmentation: Entrie Game+First Half+Second Half+Overtime
LastNGames	All Games	Recent *N* Games
LeagueID	NULL	League Identifier

Continued (Table 2)

Parameter	Default Value	Description
Location	All Locations	Game Location: All Locations+Home+Road
Month	All Months	Game Month: All Months+Specific Month
OpponentTeamID	Vs All Teams	Opponent Team Identifier
Outcome	All Outcomes	Game Result: All Outcomes+Wins+Losses
PerMode	Totals	Data Return Type: Totals+Per Game
Period	All Quarters	According to Quarter Query Data, eg 2nd Quarter
PlayerID	NULL	Player's ID Value
Season	2015-16	Season Identifier
SeasonSegment	Entire Season	Entire Season+Pre All-Star+Post All-Star
SeasonType	Regular+Season	Regular+Playoffs+All Star
TeamID	NULL	Team Identifier
VsConference	All Conference	Conference: All Conference+East+West
VsDivision	All Division	Division: All Division, Atlantic, Central

JSON(Javascript object notation)作为轻量级的数据交换格式,可以与 XML(extensible markup language)格式相互转化.本文中利用 Python 代码将 JSON 文件中的数据转化为 pandas DataFrame,并将 JSON 中感兴趣的字段解析出来,并存储为 CSV 格式,核心代码如下:

核心代码 2. JSON 数据转化代码.

```
import pandas as pds
df=pds.DataFrame()
for playerid in playerids:
    with open("{playerid}.json".format
        (playerid=playerid)) as json_file:
        parsed=json.load(json_file)
        ['resultSets'][0]
        df=df.append(pd.DataFrame(parsed
        ['rowSet'],columns=parsed['headers']))
df =df.rename(columns={'PLAYER_
    NAME_LAST_FIRST': 'PLAYER'})
df[df['PASS_TO']
.isin(df['PLAYER'])][['PLAYER',
'PASS_TO','PASS']].to_csv('passes.csv',
index=False)
df['id']=df['PLAYER'].str.replace(', ', '')
```

2.3　定义并构建 Vertex 表及 Edge 表

由于 Vertices,Edges,Triplets 是 GraphX 中最重要的 3 个概念.其中 Vertices 类所对应的 RDD 为 VertexRDD,属性有 ID 及顶点属性;Edges 类所对应的 RDD 为 EdgeRDD,属性有源顶点的 ID、目标顶点的 ID 及边属性;Triplets 所对应的 RDD 类型为 EdgeTriplets,而 Triplets 可通过所对应的 Vertices 与 Edges 经过 JOIN 操作得到,所以 Triplets 属性有:源顶点 ID、源顶点属性、边属性、目标顶点 ID、目标顶点属性.由于 GraphX 统一了 Table View 与 Graph View,即实现了 Unified Representation,通过构造 Vertex 及 Edge 表,并将 Vertex Edge 表数据加载到 Spark 内存中便可构造出传球网络图.

定义 1. 传球网络图.传球网络图用 $G=\{V,E\}$ 表示,其中 $V=\{p_1,p_2,\cdots,p_n\}$ 表示传球网络中所有球员的集合(设传球网络中球员的数量为 n);而 $E=\{\langle p_i\rightarrow p_j\rangle \mid p_i\in V,p_j\in V\}$ 表示任意顶点(球员)之间传球关系的集合,例如 $p_i\rightarrow p_j$ 表示球员 p_i 传球给 p_j.

在实际的传球网络应用场景中,设关系表 Vertex(PLAYERID,PLAYERNAME)及 Edge(PASSERID,PASSTOID,FREQUENCY,PASSNUM,AST,FGM,FGA,FGP,FGM2,FGA2,FGP2,FGM3,FGA3,FGP3)分布存储节点集合 V 及关系集合 E 的数据.Vertex 及 Edge 表数据字典如表 3 及表 4 所示.

结合 Spark 中对 Vertices 类的定义,字段 PLAYERID 为顶点 ID,字段 PLAYERNAME 为顶点属性.

结合 Spark 中对 Edges 类的定义,字段 PASSERID 为源顶点 ID,字段 PASSTOID 为目标顶点 ID,字段 FREQUENCY,PASSNUM,AST,FGM,FGA,FGP,FGM2,FGA2,FGP2,FGM3,FGA3,FGP3 都为边属性.特别地,在 Edge 表中,对于任意的 PASSERID

Table 3 Data Dictionary of Table Vertex

表 3 Vertex 表数据字典

Name	Data Type	Description
PLAYERID	INT	The ID of This Player in nba.com Database
PLAYERNAME	VARCHAR	The Name of This Player

Table 4 Table Edge's Data Dictionary

表 4 Edge 表数据字典

Name	Data Type	Description
PASSERID	INT	The Passer's ID
PASSTOID	INT	The Receiver's ID
FREQUENCY	FLOAT	The Frequency of Passes Made
PASSNUM	INT	The Number of Passes Made from the Given Teammate
AST	INT	The Number of Assists Recorded When Making Pass from the Given Teammate
FGM	INT	Field Goals Made—The Number of Field Goals Made When Making a Pass from the Given Teammate
FGA	INT	Field Goals Attempted—The Number of Field Goals Attempted When Making a Pass from the Given Teammate
FGP	FLOAT	Field Goal Percentage—The Field Goal Percentage of Shots that Followed a Pass from the Given Teammate
FGM2	INT	2 Point Field Goals Made
FGA2	INT	2 Point Field Goals Attempted
FGP2	FLOAT	2 Point Field Goal Percentage
FGM3	INT	3 Point Field Goals Made
FGA3	INT	3 Point Field Goals Attempted
FGP3	FLOAT	3 Point Field Goal Percentage

都需满足约束(设 Edge 表中的 FREQUENCY 字段为变量 $Frequency$,并设任意球员的传球队员数为 i,其中 $i\in[1,n]$):

$$\sum_i Frequency_i = 1. \tag{1}$$

并且,对于给定的〈PASSERID, PASSTOID〉, $Frequency$ 的计算公式为(设 Edge 表中 PASSERID 与 PASSTOID 字段为变量 $passerid$ 与 $passtoid$):

$$Frequency\langle passerid, passtoid\rangle = \frac{passnum\langle passerid, passtoid\rangle}{\sum_i passnum\langle passerid\rangle}. \tag{2}$$

式(2)能计算出任意 2 个球员之间的传球概率.设 Edge 表中 FGM, FGA, FGP 字段分别为变量 fgm, fga, fgp,那么对于给定的〈PASSERID, PASSTOID〉, fgp 可计算为

$$fgp = \frac{fgm}{fga}\times 100\%. \tag{3}$$

同样,对于给定的〈PASSERID, PASSTOID〉,字段 FGP2 及 FGP3 的值可由计算为

$$fgp2 = \frac{fgm2}{fga2}\times 100\%, \tag{4}$$

$$fgp3 = \frac{fgm3}{fga3}\times 100\%. \tag{5}$$

在实际的应用场景中,变量 $Frequency$, fgp, $fgp2$, $fgp3$ 是教练制定即时战术最重要的数据支撑.

2.4 传球网络的构建

本节以 2015—2016 赛季的总冠军骑士队为例,构建其 82 场常规赛期间所有场次的传球网络图.首先,通过 2.2 节中的数据获取及清洗获得传球网络图的顶点数据如表 5 所示:

Table 5 The Cavaliers Passing Network Diagram's Vertex Data

表 5 骑士队传球网络顶点数据

ROW	PLAYERID	PLAYERNAME
1	203099	Cunningham_Jared
2	202697	Shumpert_Iman
3	2210	Jefferson_Richard
4	2592	Jones_James
5	201619	Kaun_Sasha
6	202618	Irving_Kyrie
7	2747	Smith_J. R.
8	201567	Love_Kevin

Continued (Table 5)

ROW	PLAYERID	PLAYERNAME
9	202684	Thompson_Tristan
10	203925	Harris_Joe
11	202389	Mozgov_Timofey
12	203521	Dellavedova_Matthew
13	101112	Frye_Channing
14	2760	Varejao_Anderson
15	2563	Jones_Dahntay
16	2590	Williams_Mo
17	2544	James_LeBron
18	203895	McRae_Jordan

以 2015—2016 赛季 NBA 总决赛 MVP LeBron James 为例.通过解析 2.2 节中获得的 JSON 数据,通过如下数据清洗及格式转化程序(Scala 代码,如核心代码 3 所示),得到符合 2.2 节中对边 Edge 表结构的数据如表 6 所示.

核心代码 3. 数据清洗及格式转化.

```
object WebPageDataTransfer {
  def main(args:Array[String]):Unit={
val lines=Source.fromFile("/Cavaliers_data/
  James_LeBron.cvs").
  getLines().toList
var edges=lines.map {
    line=>val fields=line.split("\t")
    ("James_LeBron",fields(0).toString,
      fields(2).toString,fields(3).toLong,
      fields(4).toLong,fields(5).toLong,
      fields(6).toLong,fields(7).toFloat,
      fields(8).toLong,fields(9).toLong,
      fields(10).toFloat,fields(11).toLong,
      fields(12).toLong,fields(13).toFloat)}
    val pw=new PrintWriter(new File
      ("/Cavaliers_data/out_data/James_
      LeBron.cvs"))
    for (i<-edges) {
      pw.write(i.toString()+"\n")
    }
    pw.close()
  }
}
```

数据格式转化程序将原始数据从目录 Cavaliers_data 读入数据,数据处理完毕后将数据写入 out_data 目录中.

Table 6　LeBron James Regular Season's Passing Network Edge Data

表 6　LeBron James 常规赛传球网络 Edge 数据

srcId	dstId	attr
2544	201567	21.5%,820,126,146,322,45.3,80,163,49.1,66,159,41.5
2544	202618	18.9%,721,41,80,191,41.9,64,130,49.2,16,61,26.2
2544	2747	16.6%,634,105,129,313,41.2,42,109,38.5,87,204,42.6
2544	203521	14.6%,558,49,59,124,47.6,22,53,41.5,37,71,52.1
2544	202389	5.7%,216,52,52,102,51.0,52,97,53.6,0,5,0.0
2544	202684	5.5%,209,46,47,76,61.8,47,76,61.8,0,0,0.0
2544	2590	5.2%,199,15,22,51,43.1,15,25,60.0,7,26,26.9
2544	202697	3.9%,148,19,22,61,36.1,12,25,48.0,10,36,27.8
2544	2210	2.9%,109,23,25,50,50.0,18,26,69.2,7,24,29.2
2544	101112	1.7%,64,18,17,32,53.1,2,5,40.0,15,27,55.6
2544	2760	1.3%,48,6,6,15,40.0,6,15,40.0,0,0,0.0
2544	203099	1.2%,47,9,9,17,52.9,3,6,50.0,6,11,54.5
2544	2592	1.0%,40,4,4,17,23.5,0,0,0.0,4,17,23.5

Notes: srcId,dstId and attr field are corresponding to the Edges class source vertex ID,target vertex ID and edge attribute. The attr can be further divided into: FREQUENCY,PASSNUM,AST,FGM,FGA,FGP,FGM2,FGA2,FGP2,FGM3,FGA3,FGP3.

表 6 只是 LeBron James 单个球员的数据,当将骑士队所有队员的传球数据累加便形成整个球队 82 场常规赛所有的传球数据.构建骑士队常规赛骑士对传球网络的 Spark GraphX 如核心代码 4 所示:

核心代码 4. 骑士队常规赛传球网络构建.

① package Cavaliers.pass.anlaysis.datamaker

```
② import org. apache. spark. _import org. apache. spark. graphx. _import org. apache. spark. rdd. RDD
③ object CavalierPassNetWork {
④ def main(args: Array[String]): Unit={
⑤   val conf=new SparkConf(). setAppName("CavalierPassNetWork"). setMaster("local[4]")
⑥   val passes:RDD[String]=sc. textFile("/Cavaliers /edgs. cvs")
⑦   val player:RDD[String]=sc. textFile("/Cavaliers/vertex. cvs")
⑧   val vertices:RDD[(VertexId,String)]=player. map {
      line=>val fields=line. split(" ")
      (fields(0). toLong,fields(1))
    }
⑨   val edges:RDD[Edge[(String,Long,Long)]]=passes. map {
      line=>val fields=line. split(" ")
      Edge(fields(0). toLong, fields(1). toLong,(fields(2). toString(), fields(3). toLong, fields(4). toLong))
    }
⑩   val graph=Graph(vertices,edges)
⑪   }
⑫ }
```

在如上传球网络构建代码中，行⑥⑦代码分别将顶点与边的数据从 cvs 文件中读取到内存，行⑧从 *player* 常量构建图顶点，行⑨在行⑥常量 *pass* 的基础上构建传球网络的边，边的属性可根据应用场景进行自定义. 行⑩在常量 *vertices* 及 *edges* 的基础上，通过调用 Graph 类的构造函数，构造出传球网络 *graph*，为传球网络的应用打下基础.

3 传球网络的可视化研究

3.1 传球网络与社交网络可视化的不同

大数据可视化技术将枯燥的海量数据通过图表的形式展示出来，能够更加直观地向用户展示数据之间的联系，从而减少用户挖掘数据内涵所耗费的时间. 对于具有海量节点和边的大规模网络，如何在有限的屏幕空间中进行可视化，是大数据时代的可视化技术面临的难点和重点[33]. Herman 等人[34]对图的可视化基本方法和技术进行了综述，虽然针对图的可视化研究人员提出了不少的新方法与技术(如树图技术 Treemaps[35]、Voronoi 图填充[36]、综合性的 TreeNetViz[37]等)，但是经典的基于节点和边的可视化方法依然是图可视化的主要形式.

传球网络的可视化与社交网络图的可视化非常类似，其中社交网络图可通过 NodeXL 等工具进行可视化. 但是，本文在尝试利用 NodeXL 绘制传球网络图时，发现 NodeXL 等社交网络的可视化工具并不适合传球网络的可视化. 因为传球网络相比社交网络单纯的好友关系，需要表达出 4 项更多的个性化信息：

1) 传球网络图中球员重要性的区分. 因为不同的球员在传球网络中的地位与影响力有所不同(如常规赛 MVP Stephen Curry 在传球网络中的重要度肯定比其他角色球员高). 本文通过设定不同顶点的半径，以此区分不同球员在传球网络中的重要性，所以，需要探索计算球员重要性的方法.

2) 传球网络中球员角色(位置)的区分. NBA 中场上 5 名队员角色有控球后卫(point guard)、得分后卫(shooting guard)、小前锋(small forward)、大前锋(power forward)及中锋(center)，不同的位置担任的场上任务、跑位、战术、传球方法等等存在着很大的差异，所以传球网络中应区分不同球员的位置. 本文中将球员的角色用不同颜色的顶点表示，解决不同球员之间的角色区分问题.

3) 任意 2 个球员之间传球次数(频率)的区分. 球员之间的传球次数属于边的属性信息，为了表达任意 2 个球员之间的传球次数，可采用具体数字标注的形式，即在边上显示具体的传球数据. 但是，实践中发现由于边的数量庞大，采用数字标注的方法导致传球网络图过于杂乱. 因此，本文中将传球次数(或频率)信息通过边的粗细来表达，即 2 个球员之间的传球次数越多，他们之间的边越粗；反之，他们之间的传球次数越少，边越细.

4) 传球网络图的用户交互能力. 传统数据可视化技术大多通过静态的图片进行数据展示，不具备动态的交互能力. 本文生成的传球网络图由于底层采用 JavaScript 脚本，通过浏览器展示传球网络图的同时提供动态的用户事件响应机制，用户可利用鼠标操作关注某一特定球员的传球情况.

3.2 传球网络可视化的输入数据

综合 3.1 节中的 4 点个性化需求，本节从超过 10 种(如 iCharts, Fusion Charts Suit, Modest Maps, Chartkick, Bonsai, Google Charts, Gephi, Protvis 等)可视化工具(框架)中，挑选出 networkD3(R 语

言中基于 JavaScript 的数据可视化工具)框架中的 forceNetwork 作为传球网络的可视化工具. 利用 networkD3 绘制 forceNetwork 传球网络图需要准备的输入数据有 2 类:

1) 顶点数据. 顶点数据包括顶点名称、顶点 ID、顶点分类信息(区分球员位置)、顶点大小(区分球员重要性). 顶点数据中,顶点名称为球员姓名 PLAYERNAME 字段,顶点 ID 为 PLAYERID 字段.

2) 边数据. 边数据包括源节点、目标节点、边属性. 其中,源节点与 Edge 表中的 PASSERID 字段对应;目标节点与 Edge 表中的 PASSTOID 字段对应;边属性可根据应用场景或用户需求的不同进行自定义. 例如,如果用户关注球员之间的传球次数,则边属性为 Edge 表中 PASSNUM 字段;如果用户关注 2 个球员之间的传球助攻转化情况,则边属性为 Edge 表中的 AST 字段.

球员分类数据需要人工标注,由于某些球员可打 2 个以上的位置(如 Tristan Thompson 可打中锋与大前锋 2 个角色),所以实践中并不是严格按照控球后卫、得分后卫、小前锋、大前锋、中锋的角色进行划定,而是定义出 5 种新的角色分类:Guard, Forward, Center, Forward-Guard, Forward-Center. 同样以骑士队为例,球员位置分类及颜色信息如表 7 所示:

Table 7　The Cavaliers Player's Position Classification Table

表 7　骑士队球员位置分类表

PLAYERNAME	LABLE	POSITION	COLOR
Cunningham_Jared	1	Guard	
Shumpert_Iman	1	Guard	
Jefferson_Richard	2	Forward	
Jones_James	4	Forward-Guard	
Kaun_Sasha	3	Center	
Irving_Kyrie	1	Guard	
Smith_J. R.	4	Forward-Guard	
Love_Kevin	2	Forward	
Thompson_Tristan	5	Forward-Center	
Harris_Joe	1	Guard	
Mozgov_Timofey	3	Center	
Dellavedova_Matthew	1	Guard	
Frye_Channing	2	Forward	
Varejao_Anderson	3	Center	
Jones_Dahntay	4	Forward-Guard	
Williams_Mo	1	Guard	
James_LeBron	2	Forward	
McRae_Jordan	1	Guard	

3.3　球员在传球网络中的重要度计算

计算球员重要度(即传球网络中顶点半径大小)本质上与计算网页重要度算法 PageRank 思想非常类似,都是将图中节点的重要性映射为一个具体的数字. 所以借鉴 Google 的 PageRank 算法,本文提出计算球员在传球网络中重要程度的算法 *PlayerRank*,其计算为

$$PR(p) = (1-\alpha) + \alpha \sum_{i=1}^{n} \frac{PR(Ti)}{C(Ti)}. \tag{6}$$

其中,$PR(p)$表示一个球员 p 在传球网络中的重要程度,其取值范围为[0,10],值越大说明重要性越高;a 为随机权重变量,其取值范围为[0,1],通常 PageRank 中默认取值 0.85;n 表示传球网络中球员(顶点)的数量;$T_i(i=1,2,3,\cdots,n)$表示向球员 p 传球的球员集合;$C(T_i)$表示球员 T_i 向外传球球员的数目;而$\frac{PR(T_i)}{C(T_i)}$则表示给球员 p 传球的球员 T_i 给予球员 p 所给予的 PR 值. 在 PageRank 算法中,一个网页被另一个 PR 值高的页面所链接,说明这个网页的本身质量不错,其 PR 值也会提高;此原理在传球网络中同样适用,因为 NBA 中有首发阵容(第 1 阵容)、替补阵容(第 2 阵容)及第 3 阵容(通常指常规时间无法进入轮换阵容的球员)的区分,球员 p 经常与重要球员(如 Stephen Curry 或 LeBron James)进行传、接球配合,表明该球员 p 与重要球员所处同一阵容中,说明球员 p 也很重要. 与 PageRank 算法计算方法一样,算法 *PlayerRank* 设每个球员 p 的初始 PR 值为 1;然后根据式(6)递归地计算每个球员的 PR 值,直到所有球员的 PR 值趋于稳定.

由于 GraphX 中提供了 PageRank 的实现,在 2.4 节中传球网络构造的基础上,通过调用 PageRank API 能够快速地对球员重要度进行计算. 但是,PageRank 算法的输入参数并不支持计算边的属性(如传球次数、频率等),所以需要将 2.4 节中的传球网络进行变换,才能进行球员重要度的计算. 如图 4 为数据转化示例.

如图 4 所示,将选出 Edge Table 中的一条记录作为示例,示例中 $passerid = 2544$, $passtoid = 202618$, $passnum = 721$,这行记录表示 LeBron James 在常规赛期间总共给 Kyrie Irving 传球 721 次. 转化为只有字段 PASSERID 与 PASSTOID 的临时表,并且 $passerid$ 值为 2544, $passtoid$ 值为 202618,并将行重复 721 次. 数据变换程序的 Scala 核心代码如下:

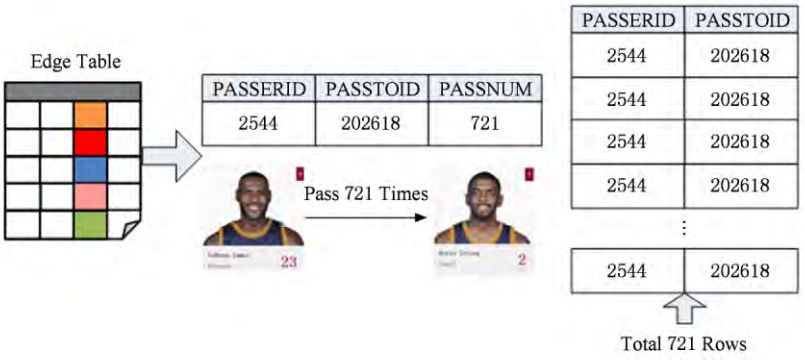


Fig. 4 Input data transformation for PR computing.

图 4 PR 计算输入数据变换示例

核心代码 5. PageRank **算法数据变换程序.**

```
① object ScalaGenPassFile {
②   def main(args:Array[String]):Unit={
③     val lines=Source.fromFile("/passes.
        csv").getLines().toList
④     var edges=lines.map{line=>
        val fields=line.split(" ")
        (fields(0).toLong, fields(1).
      toLong,fields(2).toLong)}
⑤     var javalist=new JavaList[String]()
⑥     for (i<-edges)
⑦       for (j<-0 to i._3.toLong.toInt-1)
⑧         javalist.add((i._1.toLong,
            i._2.toLong).toString())
⑨     val pw=new PrintWriter(new File
        ("/PRdata/RPInputdata.cvs"))
⑩     for (i<-javalist.toArray())
⑪       pw.write(i.toString()+"\n")
⑫     pw.close()
⑬   }
⑭ }
```

代码行③将骑士队常规赛所有的传球数据(表 Edge **的数据)读取到常量** *lines* **中;行④将** Edge **表中的字段** PASSERID, PASSTOID **及** PASSNUM **抽取出来;行⑥～⑧构造临时表(**PASSERID, PASSTOID**)并循环** PASSNUM **次进行数据的填充;行⑨～⑫行将生成的数据写入** RPInputdata.cvs **文件中.** RPInputdata.cvs **文件是计算** *PR* **值的输入数据,计算** *PR* **值调用** GrahpX **的** *pageRank* **的函数,完成球员重要度计算的核心** Spark **代码如下:**

核心代码 6. 球员重要度计算.

```
① object CavalierPR {
②   def main(args:Array[String]):Unit={
③     val conf=new SparkConf().
        setAppName("CavalierPR").
        setMaster("local[4]")
④     val sc=new SparkContext(conf)
⑤     val pass_file:RDD[String]=
        sc.textFile("/PRdata /
        RPInputdata.cvs")
⑥     val player_file:RDD[String]=
        sc.textFile("PRdata/Cavalier__
        players.cvs")
⑦     val vertices:RDD[(VertexId,String)]
        =player_file.map {
        line=>val fields=line.split(" ")
        (fields(0).toLong,fields(1))}
⑧     val edges:RDD[Edge[Long]]=
        pass_file.map{line=>
        val fields=line.split(" ")
        Edge(fields(0).toLong,fields(1).
        toLong)}
⑨     val graph=Graph(vertices,edges)
⑩     val cavalier_PR_value=graph.
        pageRank(0.01,0.15).vertices
⑪     cavalier_PR_value.join(vertices).
      saveAsTextFile("/PRdata/cavalier__
      PR_value.cvs")
⑫   }
⑬ }
```

球员重要度计算的 Spark **代码行③～④完成** Spark **环境的初始化;行⑤载入上文中生成的** RPInputdata.cvs **文件数据(即边的数据);行⑥载入骑士队球员数据(即顶点的数据);行⑦在常量**

player_file 的基础上构造出图的顶点,其类型为 RDD[(VertexId,String)];行⑧在常量 pass_file 的基础上构造出图的边,其类型为 RDD[Edge[Long]];行⑨通过顶点与边构造出图;行⑩通过调用 GraphX 的函数 pageRank 计算球员的重要度,其中 0.01 表示计算精度,0.15 为 resetProb 参数;最后,行⑪将计算结果存储到/PRdata/cavalier_PR_value.cvs 文件中.最终 PlayerRank 的计算结果如表 8 所示:

Table 8　The Cavaliers Player's PlayerRank Computing Result

表 8　骑士队球员重要度计算结果

PLAYERID	PLAYERNAME	PLAYERRANK
203099	Cunningham_Jared	0.4224565607897039
202697	Shumpert_Iman	0.7364259672296204
2210	Jefferson_Richard	0.7784616241827468
2592	Jones_James	0.5112005308670231
201619	Kaun_Sasha	0.2204240081435887
202618	Irving_Kyrie	**1.7713079353999628**
2747	Smith_J. R.	**1.2849631892334719**
201567	Love_Kevin	**1.7702827741966163**
202684	Thompson_Tristan	0.8603951189450415
203925	Harris_Joe	0.1646007801811211
202389	Mozgov_Timofey	0.6979348543020524
203521	Dellavedova_Matthew	**1.992442548658214**
101112	Frye_Channing	0.49740698986676474
2760	Varejao_Anderson	0.36266719917791773
2563	Jones_Dahntay	0.20389846121127372
2590	Williams_Mo	0.9011093825227108
2544	James_LeBron	**2.5503073589131113**
203895	McRae_Jordan	0.33383878452187665

从骑士队球员重要度计算结果我们发现,*PlayerRank* 值超过 1 的一共有 5 名球员,而这 5 名球员在常规赛期间通常出任首发球员.其中,LeBron James 的 *PlayerRank* 值达到了惊人的 2.55;而首发控球后卫 Matthew Dellavedova 其 *PlayerRank* 值也达到了 1.99;而 Kyrie Irving 与 Kevin Love 的 *PlayerRank* 值都在 1.77 左右.虽然 Tristan Thompson 也经常担任先发球员,但是由于他的位置为中锋,传接球的机会较少,导致其 *PlayerRank* 较低.*PlayerRank* 值除了能够评估一个球员在传球网络中的重要程度外,另外一个重要作用就是可以得出一个球队球员之间的球权分配情况.

3.4　ForceNetwork 可视化传球网络

NetworkD3 是 R 语言下的一个绘图工具包,使用前必须安装 NetworkD3 包,然后通过 library(networkD3)导入到程序中.R 语言绘制骑士队传球网络图核心代码如下:

核心代码 7. R 语言绘制传球网络.

```
① library(networkD3)
② passes<-read.csv("RPInputdata.cvs")
③ groups<-read.csv("groupsR.csv")
④ size<-read.csv("cavalier_PR_value.cvs")
⑤ passes$source<-as.numeric(as.factor
   (passes$PASSERID))-1
⑥ passes$target<-as.numeric(as.factor
   (passes$PASSTOID))-1
⑦ passes$PASSNUM<-passes
   $PASSNUM/50
⑧ groups$nodeid<-groups$PlayerName
⑨ groups$name<-as.numeric(as.factor
   (groups$PlayerName))-1
⑩ groups$group<-as.numeric(as.factor
   (groups$label))-1
⑪ nodes<-merge(groups,size[-1],
   by="id")
⑫ nodes$PlayerRank<-nodes
   $PlayerRank^2*100
⑬ forceNetwork(Links=passes,
                Nodes=nodes,
                Source="source",
                fontFamily="Arial",
                colourScale=JS("d3.scale.
                category10()"),
                Target="target",
                Value="PASSNUM",
                NodeID="nodeid",
                Nodesize="PlayerRank",
                linkDistance=350,
                Group="group",
                opacity=0.8,
                fontSize=16,
                zoom=TRUE,
                opacityNoHover=TRUE)
```

代码行①将 networkD3 包导入;行②~③将存储在 RPInputdata.cvs 中的边数据,groupsR.csv 中

的顶点分类数据及 cavalier_PR_value.cvs 文件中的球员 *PlayerRank* 值数据分别导入到变量 *passes*, *groups*, *size* 变量中；行⑤～⑫将 *passes*, *groups*, *size* 变量中的列数据分别解析出来，其中变量 *source* 存储 PASSERID(即传球人 ID)、变量 *target* 存储 PASSTOID(即接球人 ID)、变量 *PASSNUM* 存储传球次数、变量 *nodeid* 用于存储球员的姓名、变量 *PlayerRank* 用于存储球员的 *PR* 值；行⑬调用 *forceNetwork* 函数绘制传球网络图，并设置了字体类型为 *fontFamily*、字体大小 *fontSize*、边的长度 *linkDistance* 等参数. 最终绘制的骑士队常规赛传球网络图如图 5 所示：

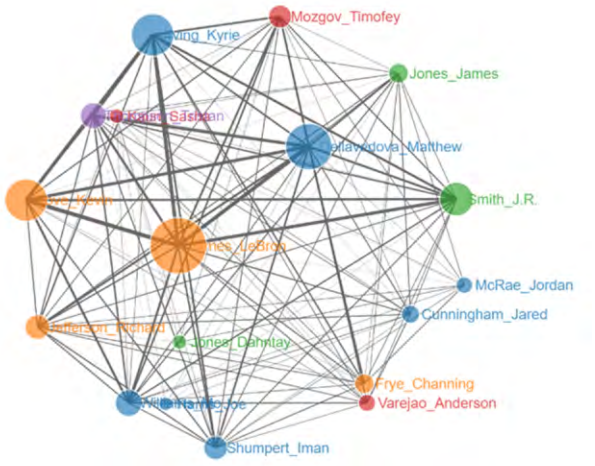


Fig. 5 Cavaliers regular season passing network diagram.

图 5 骑士队常规赛传球网络图

骑士队传球网络图中顶点的数量为 17，边的数量为 216. 图 5 中不同位置的球员颜色不同，并且球员的 *PlayerRank* 值越大，顶点越大，任意 2 个球员之间的传球数越多，连接他们之间的边越粗. 通过图 5 不难发现，首发球员的 *PlayerRank* 值高，并且他们之间的传球数多. 通过颜色可以发现 LeBron James 与 Kevin Love 颜色相同，位置都为前锋(forward)；而在 Kyrie Irving 与 Matthew Dellavedova 都为后卫(guard)；如果将总决赛期间的传球网络与常规赛进行对比，就会发现由于 LeBron James 与 Kevin Love 及 Kyrie Irving 与 Matthew Dellavedova 位置的重叠，同一位置 *PlayerRank* 值接近的 2 个球员很难在比赛中都发挥出色.

在 2015—2016 赛季创造了 NBA 常规赛历史最佳战绩(73 胜 9 负)、NBA 开局最长连胜纪录(24 连胜)以及 NBA 主场最长连胜纪录(54 连胜)的金州勇士队，其常规赛传球网络图如图 6 所示：

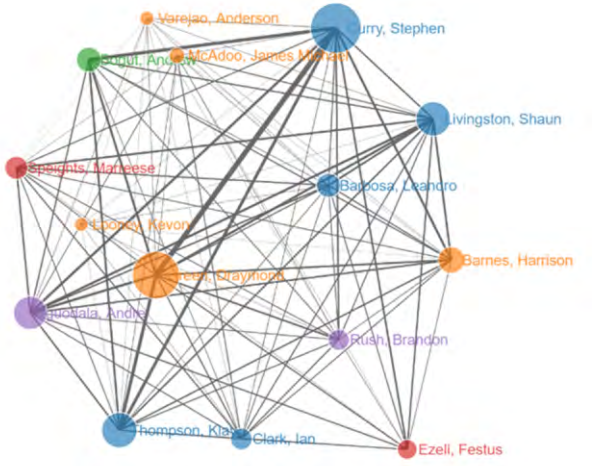


Fig. 6 Warriors regular season passing network diagram.

图 6 勇士队常规赛传球网络图

4 传球网络应用场景及分析

通过 GraphX 构建传球网络，能够利用 GraphX 提供的 API 分析传球网络中顶点及顶点属性、边及边的属性信息，并提供了深入挖掘传球数据内涵的接口. 传球网络图实质上是一个球队传球数据的画像，可帮助球员、教练、球队、球迷各种不同角色的人从不同的层面去挖掘传球数据，并以此给他们带来帮助. 传球网络从不同角色、不同方面的应用场景很多，本文无法穷举.

4.1 NBA 传球数据对比赛结果的影响分析

通过数据获取、清洗及统计整理后，对 2013—2014，2014—2015，2015—2016 赛季常规赛期间的传球数据进行了总体统计，统计结果如表 9 所示. 以 2015—2016 赛季为例，NBA 中 30 支队伍在 2015—2016 赛季总共进行了 1 228 场比赛，比赛总时间为 593 840 min. 而在这 1 228 场比赛中，总共产生了 747 761 次传球，平均每场每队的传球数为 304.46. 而这些传球产生的助攻总数为 54 728，即 NBA 比赛平均传球 13.663 次才能产生 1 次助攻.

观察表 9 中数据不难发现 3 个赛季中场均传球次数在不断地增加，从 2013—2014 赛季的场均 299.98 次，增长到 2014—2015 赛季的 301.36 次，以及到 2015—2016 赛季的 304.46 次. 场均传球次数的增长原因是因为联盟中不少球队受到金州勇士队“小球”战术(篮球术语中“小球”又称“跑轰”，就是主要以突破和外线为主的打法)的影响，从而改变了这些球队的比赛风格和战术选择. 而“小球”战术弱化了传统中锋的作用，比赛节奏快，相同时间内的进攻回合数更多，所以导致场均传球次数增加.

表 10 将 2015—2016 赛季 30 支球队的场均传球数据进行了汇总,包括场均传球数、场均助攻数、传球助攻比(产生 1 次助攻需要的传球数量)、传球助攻率(1 次传球转化为助攻的概率)以及球队在 2015—2016 赛季的胜率(赛季总场次除以胜利场次).通过表 10 数据可统计出联盟 30 支球队的场均助攻数为 22.277,传球助攻率平均为 7.347%.其中场均传球数最多的为 Utah Jazz 队,达到了场均 354.8 次;而最少的为 Oklahoma City Thunder 队,场均传球数量仅为 264 次,比场均最多的 Utah Jazz

Table 9　NBA Three Season's Overall Statistical Results for Passing Data

表 9　NBA 三个赛季传球数据总体统计结果

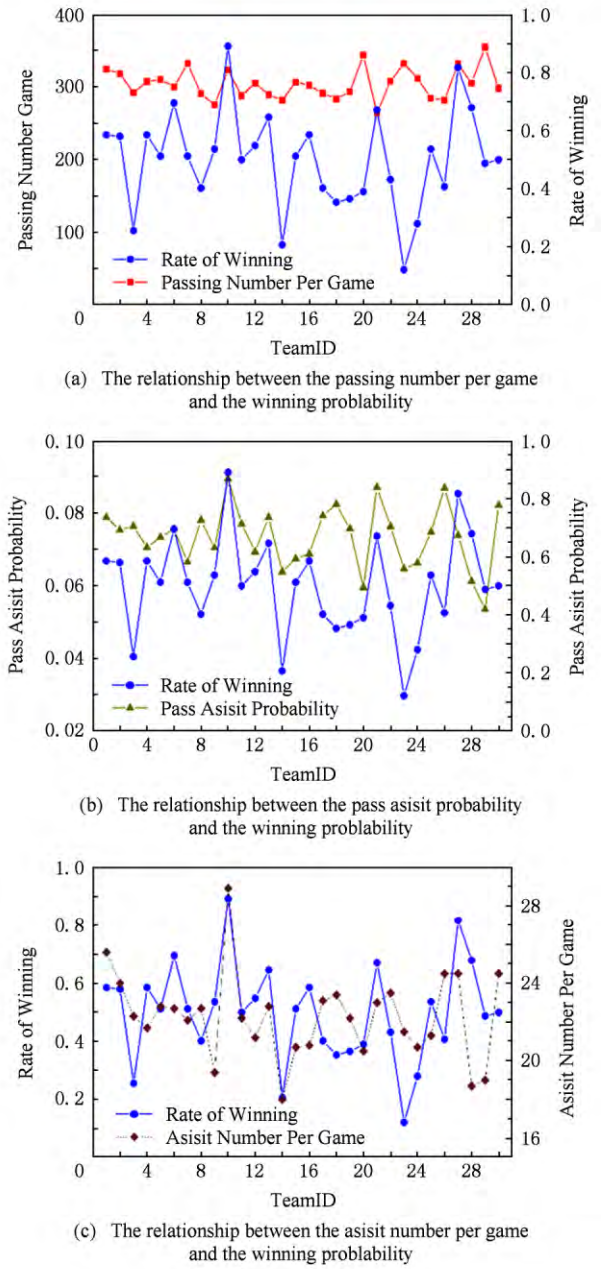
Season	Matches Played	Total Game Time/min	Total Passes	Passes Per Game	Total Assists	Assists Conversion Ratio	Assist Ratio/%
2013—2014	1 226	593 280	735 553	299.98	53 950	13.634	7.335
2014—2015	1 227	593 760	739 541	301.36	54 069	13.678	7.311
2015—2016	1 228	593 840	747 761	304.46	54 728	13.663	7.319

Table 10　Per Game Passing Data Summary for All Team in Season 2015—2016

表 10　2015—2016 赛季各球队场均传球数据汇总

ROW	Team	Average Pass Per Game	Average Assists Per Game	Pass Assists Ratio	Pass Assists Probability/%	Winrate/%
1	Hawks	324.5	25.6	12.676	7.889	58.537
2	Celtics	318.4	24	13.267	7.538	58.025
3	Nets	292	22.3	13.094	7.637	25.610
4	Hornets	307.5	21.7	14.171	7.057	58.537
5	Bulls	310.5	22.8	13.618	7.343	51.220
6	Cavaliers	300.2	22.7	13.225	7.562	69.512
7	Mavericks	332.2	22.1	15.032	6.653	51.220
8	Nuggets	290.8	22.7	12.811	7.806	40.244
9	Pistons	275.5	19.4	14.201	7.042	53.659
10	Warriors	323.1	28.9	11.180	8.945	89.024
11	Rockets	288	22.2	12.973	7.708	50.000
12	Pacers	305.6	21.2	14.415	6.937	54.878
13	Clippers	289.2	22.8	12.684	7.884	64.634
14	Lakers	281.7	18	15.650	6.390	20.732
15	Grizzlies	306.9	20.7	14.826	6.745	51.220
16	Heat	302.3	20.8	14.534	6.881	58.537
17	Bucks	291.4	23.1	12.615	7.927	40.244
18	Timberwolves	283.8	23.4	12.128	8.245	35.366
19	Pelicans	293.2	22.2	13.207	7.572	36.585
20	Knicks	344.1	20.5	16.785	5.958	39.024
21	Thunder	264	23	11.478	8.712	67.073
22	Magic	307.8	23.5	13.098	7.635	43.210
23	76ers	332.4	21.5	15.460	6.468	12.195
24	Suns	312	20.7	15.072	6.635	28.049
25	Blazers	284.7	21.3	13.366	7.482	53.659
26	Kings	281.9	24.5	11.506	8.691	40.741
27	Spurs	331.7	24.5	13.539	7.386	81.707
28	Raptors	305.4	18.7	16.332	6.123	67.901
29	Jazz	354.8	19	18.674	5.355	48.780
30	Wizards	298.2	24.5	12.171	8.216	50.000

少了 90.8 次之多. 场均助攻数方面 Golden State Warriors 达到联盟最高的 28.9 次,比垫底的 Los Angeles Lakers 的 18 次多出整整 10.9 次. Golden State Warriors 以高达 8.945%的传球助攻率领跑全联盟,而 Utah Jazz 的传球助攻率仅为 5.355%. 本文将球队的胜率与传球数据(包括场均传球数、场均助攻数、传球助攻率)进行对比分析,其对比结果如图 7 所示:



(a) The relationship between the passing number per game and the winning problability

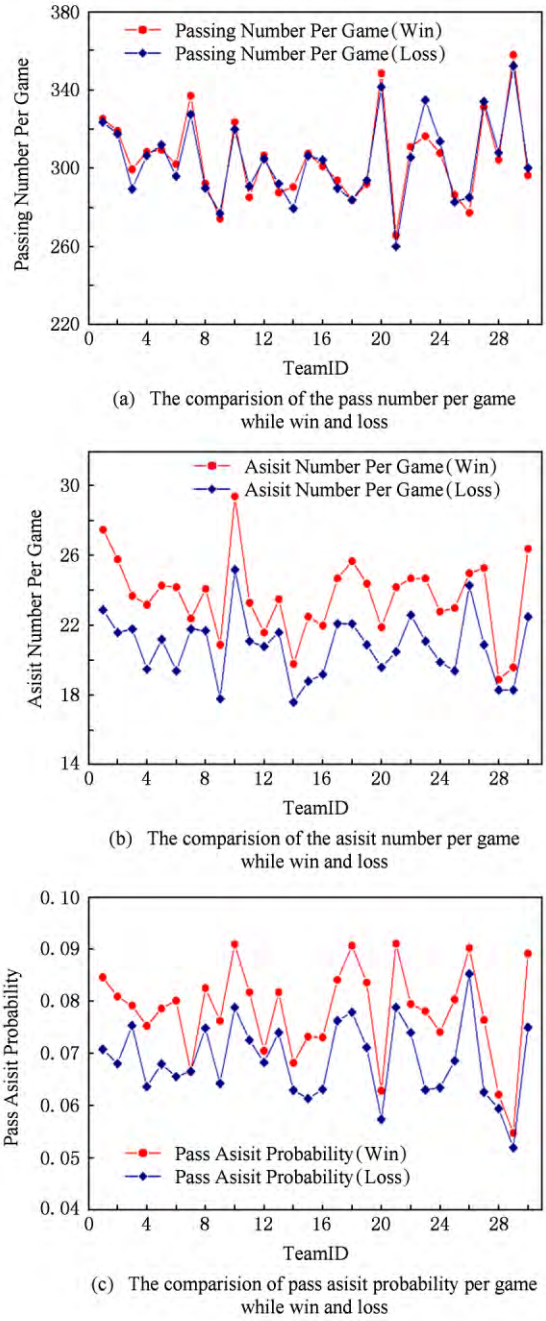
(b) The relationship between the pass asisit probability and the winning problability

(c) The relationship between the asisit number per game and the winning problability

Fig. 7 The relationship between the passing data and winning rate.

图 7 球队胜率与传球数据之间的关系

图 7(a)表示场均传球次数与胜率之间的关系;图 7(b)表示传球助攻率与胜率之间的关系;图 7(c)表示场均助攻数与胜率之间的关系. 从图 7(a)中不难看出场均传球次数与胜率之间并无直接关系(或相关性很小);而图 7(b)则能看出总体上传球助攻率越高,球队战绩越好;而从图 7(c)中能看出场均助攻数与球队胜率之间的较强相关性,场均助攻数越高的球队胜率越高,场均助攻数与胜率呈现较强的正相关. 从图 7 中的对比可以得出结论:球队战绩(或胜率)与传球的数量并无直接关系,但是与传球



(a) The comparision of the pass number per game while win and loss

(b) The comparision of the asisit number per game while win and loss

(c) The comparision of pass asisit probability per game while win and loss

Fig. 8 Analysis of the impact of passing data on the outcome of the game (Win or Loss) in 2015—2016 session.

图 8 2015—2016 赛季传球数据对比赛结果(赢或输)的影响分析

质量(传球助攻转化率与场均助攻数越高,传球质量越高)相关性较大.常规赛创造了 NBA 常规赛历史最佳战绩(73 胜 9 负)的金州勇士队,其传球助攻转化率及场均助攻数均为联盟第一.以上数据及结论可以辅助球队及教练,1)可以从数据中找出球队传球方面存在的问题(无论是传球数量,或是传球质量);2)可以辅助制定提高传球质量的战术或具体的训练项目.

为了分析传球数据对比赛结果(赢球或输球)的影响,本文将 30 支队伍的赢球(总计 1 228 场赢球记录)与输球(总计 1 228 场输球记录)条件下的场均传球数、场均助攻数及传球助攻转化率进行了统计,其结果如图 8 所示.其中图 8(a)为赢球与输球条件下的场均传球数对比,虽然总体上赢球条件下的场均传球数(304.66)略大于输球条件下的场均传球数(304.3),但是观察图 8(a)可以发现场均传球数并不直接影响比赛结果.图 8(b)为赢球与输球条件下的场均助攻数的对比,观察图 8(b)不难发现 30 支球队赢球时的场均助攻数都大于输球时的场均助攻数;实际上,2015—2016 赛季 30 支球队的 1228 场赢球记录的平均场均助攻数为 23.65,而输球的平均场均助攻数为 20.82.图 8(c)为赢球与输球条件下的传球助攻转化率的对比,不难发现 29 支队伍赢球条件下的传球助攻转化率都比输球时候高(其中只有 Dallas Mavericks 队赢球与输球条件下的传球转化率基本持平);实际上所有赢球时的平均传球助攻转化率为 7.799%,而输球时的平均助攻转化率为 6.876%.结合图 8(a)(b)(c)可以得出结论:场均传球数本身不对比赛结果产生影响,而传球质量(如场均助攻数、传球助攻转化率)会对比赛结果产生巨大的影响,同时这个结论与图 7 中得出的结论相吻合.

4.2　球队传球网络应用

4.2.1　基于传球网络的整体传球数据分析

以 2015—2016 赛季总冠军球队骑士队常规赛数据为例,在构建表 Vertex 及 Edge 并构建传球网络(如图 5 所示)的基础上,可通过 Vertex 及 Edge 的 JOIN 操作导出传球表及接球表的数据(过滤掉传球或接球数小于 100 的球员).为了实现对球队整体传球数据的分析,为将来的战术制定打下基础,其中表 11 为球队传球数据表,表 12 为接球数据表.

Table 11　Passing Data Summary for Cavalier in Regular Season 2015—2016

表 11　骑士队 2015—2016 常规赛传球数据汇总

Number	Player	FREQUENCY/%	PASSNUM	AST	AST Probability/%	FGP/%	FGP2/%	FGP3/%
7	Cunningham	1.4	356	19	5.34	35.7	42.2	24
8	Dellavedova	13.3	3272	335	10.24	47.4	53.2	36.9
9	Frye	2.7	653	25	3.83	43.5	52.4	25.5
2	Irving	11	2699	248	9.19	47.4	55.2	36.6
23	James	15.5	3813	513	13.45	45.1	49.7	39.8
24	Jefferson	4.5	1095	59	5.39	41.8	45.3	34.6
30	Jones	2.3	568	14	2.46	41.2	51.7	20.5
0	Love	13.9	3425	185	5.40	50.1	54.2	40.9
12	McRae	0.5	133	15	11.28	44.7	48.3	38.9
20	Mozgov	5.5	1352	33	2.44	46.6	54.7	30.4
4	Shumpert	4.5	1116	92	8.24	44.6	47.7	40.5
5	Smith	7.4	1820	131	7.20	45.2	50.5	35.7
13	Thompson	9.5	2330	63	2.70	41.9	47.6	32.2
17	Varejao	2	487	20	4.11	36.5	39.4	30.3
52	Williams	5.3	1304	97	7.44	43.3	49.3	29.2

图 9 对整个 2015—2016 赛季骑士队核心球员(传球或接球次数大于 100)的传接球数据进行了对比.其中图 9(a)(b)关注球员的传球与接球量(PASSNUM 与 Received Number),包括传接球的总数及在全队传接球中所占的比例(FREQUENCY).从图 9(a)中可以看出,在骑士队中传球最多的前 5 位球员为:James,Love,Dellavedova,Irving,Thompson;而接球数最多的前 5 位球员为:James,Dellavedova,Irving,Love,Smith.而图 9(c)(d)关注球员传球与接球的质量,指标包括传球与接球产生助攻的概率(Pass_AST_Probability 与 Receive_AST_Probalility)、总体命中率(FGP)、2 分球命

中率(FGP2)及 3 分球命中率(FGP3). 图 9(c)反映的是球员传球的质量,其中最重要的指标为助攻转化率,其中 James, McRae, Dellavedova, Irving, Shumpert 的传球助攻转化率较高,其中 James 达到了惊人的 13.45%,很好地解释了 James 持球率高的原因. 图 9(d)反映的是球员接到传球后转化为得分的能力,其中 Mozgov, Thompson, Frye, Smith, Jefferson 的接球助攻转化率较高;而投篮命中率可以辅助教练制定战术,并帮助球员做场上的传球决策.

Table 12 Passing Data Summary for Cavalier in Regular Season 2015—2016

表 12 骑士队 2015—2016 常规赛接球数据汇总

Number	Player	FREQUENCY/%	Received NUM	AST	AST Probability/%	FGP/%	FGP2/%	FGP3/%
7	Cunningham	1.7	414	26	6.28	36.5	39.6	31.3
8	Dellavedova	14.4	3538	114	3.22	40.9	40.2	41.6
9	Frye	2.1	519	64	12.33	42.6	56.1	37.7
2	Irving	13.5	3319	114	3.43	44.5	49.8	32.4
23	James	19.3	4752	297	6.25	50.5	56	30.5
24	Jefferson	4.2	1041	118	11.34	46	56.5	38
30	Jones	2	498	50	10.04	39.7	45.9	37.4
0	Love	12	2952	308	10.43	41.9	47	36.4
12	McRae	0.7	179	5	2.79	39.6	34.2	60
20	Mozgov	3.8	930	174	18.71	57.7	58.7	14.3
4	Shumpert	4.3	1047	78	7.45	36.9	43.2	30.7
5	Smith	8.8	2154	250	11.61	41.3	43	40.3
13	Thompson	5.3	1291	171	13.25	61.6	61.6	0
17	Varejao	1.4	340	24	7.06	43.8	44.4	0
52	Williams	6	1479	53	3.58	44.2	48.6	36.4

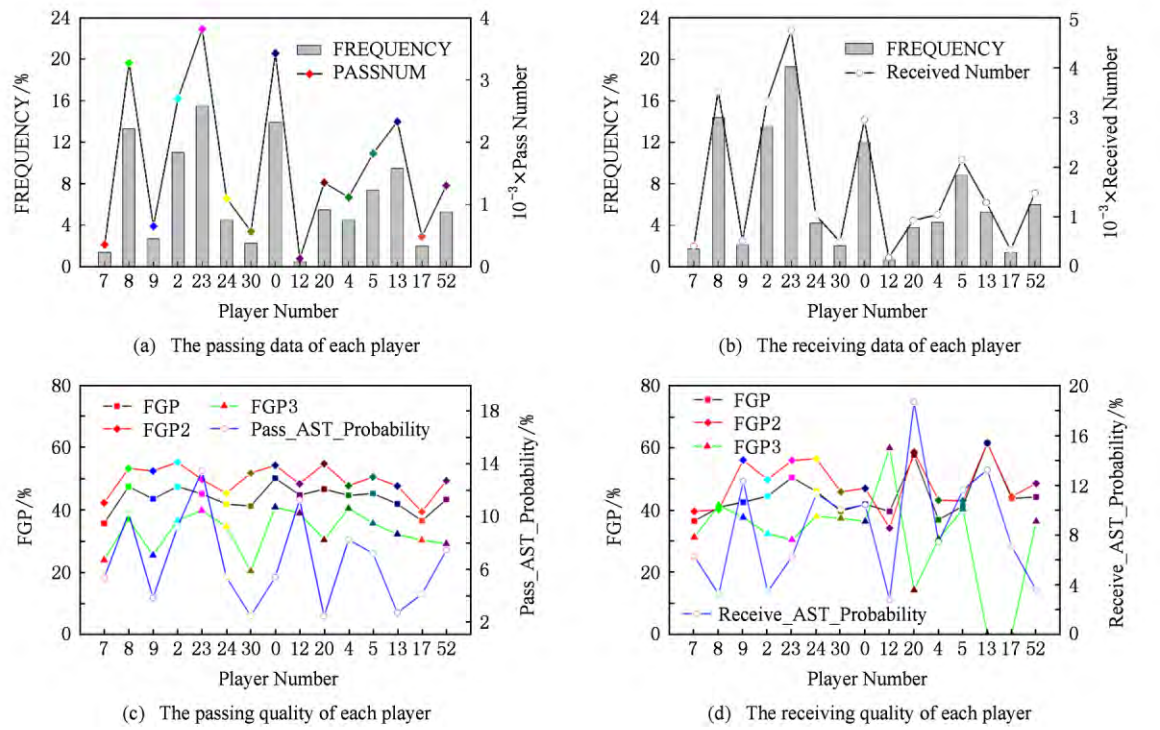


Fig. 9 Comparison of players passing and receiving data in 2015—2016 regular season of Cavaliers.

图 9 2015—2016 赛季常规赛骑士队球员传球及接球数据对比

4.2.2　基于传球网络的战术人员选择

篮球进攻战术中无论是的挡拆战术、传切配合还是三角进攻,都是建立在有效的移动与传球基础上.那么,教练在制定战术过程中就需要通过球员角色、移动数据、传球数据及投篮命中率等数据综合考量并选择战术的执行者.由于通过 3.4 节中构建的传球网络(如图 5,6 所示)中包含了球员角色、任意 2 个球员之间的传球数量及质量相关的数据、投篮命中率等数据,可以很好地协助教练基于数据选择合适的战术执行者.

例如挑选传切配合执行者时,教练可首先基于传球网络进行传球数据的筛选,选择条件为:配合熟练度高(相互传球数较多)、传球质量高(助攻数较多),并且战术执行成功率可能性高(投篮命中率高)的球员对.以骑士队为例,在图 9(c)的基础上,教练能够通过数据挑选出传球质量高的前 6 为球员:James,McRae,Dellavedova,Irving,Shumpert,Smith 作为传切配合的传球人.而通过图 9(d)教练能够挑选出助攻转化率高且投篮命中率高的球员,如果关注进攻成功率,应当以投篮命中率为球员选择标准,可以挑选出球员:Thompson,Mozgov,James,Jefferson,Irving;但是由于传切有球员角色限制,而 Thompson,Mozgov 都为中锋,所以去掉他们;最终得到适合切入的球员有:James,Jefferson,Irving.在传球人名单及切入人名单确定后,教练再根据每个球员的特点,将球员划分为组,分别进行战术训练;最后采集训练过程中战术执行的成功率数据,最终确定战术的执行球员.

4.2.3　基于传球网络的临场战术制定

篮球比赛在暂停结束后、一节比赛只剩一次进攻时间、甚至比赛时间所剩无几却落后时,教练需要重新布置一次进攻战术.特别是最后一次投篮决定比赛胜负时,教练的战术制定就显得尤为重要.无论战术选择投篮为 2 分球还是 3 分球,都可结合图 9(c)选择合适的传球人,并通过图 9(d)选择接球并执行投篮的人.当然,教练的人员选择需要与当场比赛球员的状态进行结合.实际上,当前的 NBA 教练选择执行绝杀的球员往往只是根据当场球员的状态决定,这样使得对方教练很容易猜出执行绝杀的球员,制定针对性的防守策略,从而大大减小绝杀成功的可能性.

假设教练需要制定 3 分绝杀战术,结合历史传球数据,可通过图 9(d)选出适合执行绝杀的球员名单:McRae(60%),Dellavedova(41.6%),Smith(40.3%),Jefferson(38%),Frye(37.7%),Love(36.4%),其中 McRae 仅有 10 次投篮,所以数据真实性较低,而 James 与 Irving 分别只有 30.5%与 32.4%的命中率.所以,根据历史数据,可将 James 与 Irving 作为绝杀战术中的传球执行人(对方教练的防守重心在他们身上),将 Smith 或 Dellavedova 作为绝杀执行者.

如图 10 所示为一种基于传球数据的 3 分绝杀战术示例,首先由 Thompson 边线发球给 James,Irving 沿着跑动路线所示从 3 分顶部跑到底线附近吸引防守压力,同时 Smith 沿着跑动路线指示迅速跑动到投篮点,并借助 Love 的档人与防守队员拉开空间,James 传球给 Smith 并完成绝杀,绝杀成功率在 40%左右.

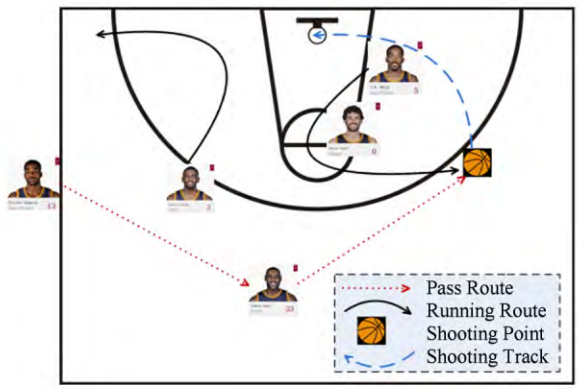


Fig. 10　The 3 points tactical lore based on passing data analysis.

图 10　基于传球数据的 3 分绝杀战术

4.3　传球网络子图应用

GraphX 中 Structural Operators 主要有 *reverse*,*subgraph*,*mask*,*groupEdges* 四种函数,其中对于传球网络分析最有用的操作是 *subgraph*.*subgraph* 可以在传球网络的基础上抽取出特定条件的子图,例如可通过如下代码抽取出 Stephen Curry 常规赛期间的传球情况:

核心代码 8.网络子图抽取代码.

```
val subgraph_Curry=graph.subgraph(epred=
    e=>e.srcId==201939)
```

如图 11 所示为库里的子图抽取结果,从图 11 中可以很直观地看出库里传球的分布情况.在子图的基础上可以针对子图及边的属性,挖掘特定球员的传球数据内涵.通过查询子图的边属性并过滤场均传球次数大于 1 的球员,将得到的数据除以 Stephen Curry 比赛的场次,可以得到 Stephen Curry 传球数据如表 13 所示:

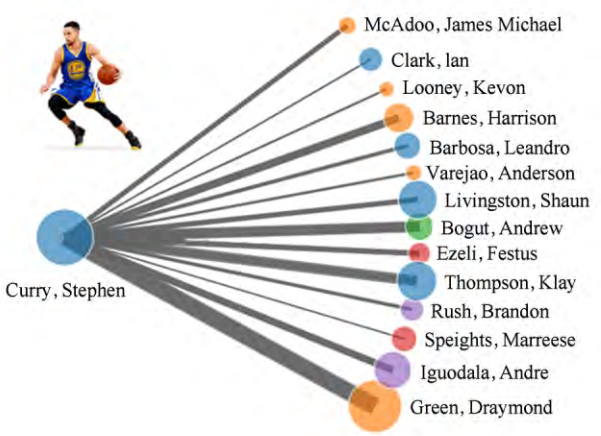


Fig. 11 Subgraph extraction results for Stephen Curry.

图 11 球员库里的子图抽取结果

一方面,Stephen Curry 本人及教练可根据表 13 数据改进比赛战术,Stephen Curry 有 34.8%的球传给了 Green,但是却只有 7.61%的助攻转化率,而传给 Thompson 时的助攻转化率达到了惊人的 19.44%,这就要求 Stephen Curry 当 Thompson 出现投篮机会后,需要多传球给 Thompson;而另一方面,其他球队可根据表 13 制定防守 Stephen Curry 的策略,可尽量让他传给不在攻击范围内且命中率较低,或助攻转化率低的球员(如 Barbosa, Iguodala, Rush),尤其要防止传给 Thompson 或处在内线的 Bogut 及 Ezeli 等内线球员.

而表 14 所示为库里常规赛期间的场均接球数据,结合表 13 数据发现 Stephen Curry 与 Green 之间的传接球(传球率为 34.8%,接球率为 41.6%)最为密切. 所以,其他球队与勇士队比赛时,重点需要切断 Stephen Curry 与 Green 之间的配合. 并且需要注意内线的 Bogut 传球给外线的 Stephen Curry 时,命中率达到惊人的 58.1%,这些数据及分析方法能帮助球队制定针对性的防守策略并提高防守效率.

Table 13 Per Game Passing Data Summary for Stephen Curry

表 13 球员库里的场均传球数据

Player	FREQUENCY/%	PASSNUM	AST	AST Probability/%	FGP/%
Green	34.8	19.7	1.5	7.61	46.8
Thompson	19.1	10.8	2.1	19.44	47
Bogut	11.8	6.7	0.7	10.45	59.4
Barnes	9.5	5.4	0.7	12.96	45.2
Iguodala	7.4	4.2	0.3	7.14	50
Livingston	3.6	2	0.2	10.00	82.1
Rush	3.3	1.9	0.2	10.53	43.9
Ezeli	3.1	1.8	0.3	16.67	47.8
Barbosa	2.6	1.5	0.1	6.67	38.9
Speights	1.9	1.1	0.2	18.18	44.4

Table 14 Per Game Received the Pass Data Summary for Stephen Curry

表 14 球员库里的场均接球数据

Player	FREQUENCY/%	PASSNUM	AST	AST Probability/%	FGP/%
Green	41.6	28.8	2	6.94	49.5
Bogut	16.7	11.5	0.8	6.96	58.1
Barnes	9.1	6.3	0.4	6.35	44.9
Iguodala	7.4	5.2	0.6	11.54	56.6
Thompson	6.2	4.3	0.3	6.98	50
Ezeli	4.6	3.2	0.1	3.13	46.8
Livingston	4.1	2.8	0.2	7.14	49.3
Rush	3.7	2.5	0.1	4.00	50
Speights	1.8	1.3	0.1	7.69	45.5
Barbosa	1.8	1.2	0.1	8.33	46.9

如果将传球网络子图中节点的大小替换为该球员传球时的助攻转化率,如图 12 所示为 Stephen Curry 的传球助攻转化效率的图示. 图 12 一方面可帮助球员明确与特定队友之间的配合效率,协助球员在球场上做出更优化的传球选择;另一方面可帮助教练了解任意 2 个球员之间的配合默契程度,为教练战术人员选择提供帮助. 同样,也可以将传球失误率等作为子图节点大小,让球员明确需要与哪些队友的配合需要改进,并在训练中得以加强,可以减小比赛中的失误次数.

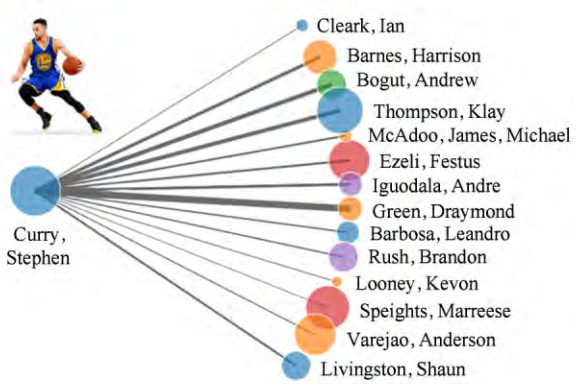


Fig. 12　The pass assist transformation efficient for Stephen Curry.

图 12　球员库里的传球助攻转化效率

4.4　传球网络在游戏中的应用

即便在顶级的篮球游戏 NBA2K16 中,游戏中的球员传球与真实的 NBA 比赛中的球员传球存在着巨大的差距. 主要存在 3 个问题:1)游戏中缺乏对传球数据的统计;2)游戏中球员传球的场均次数与真实比赛中的传球次数有差距;3)游戏中任意 2 个球员之间的传球频率与真实比赛中的传球频率有较大偏差、并且游戏中对于球员之间的传球失误场景的模拟,真实性较差. NBA2K16 是对真实的 NBA 比赛的模拟,存在的这些问题会导致游戏模拟的真实性降低,从而降低游戏者的体验.

但是,在传球网络建立的基础上,并通过 4.3 节的建立网络子图,对于任意的球员、在游戏中的任意时刻,可利用网络子图中的边属性 FREQUENCY%,建立球员的传球概率模型. 设 P 表示游戏中场上的任意一个球员,并设 P 场上的其他 4 个队友为 P_1, P_2, P_3 及 P_4,而 $Pr_{P\to P_n}(n\in[1,4])$表示游戏中球员 P 传球给其他 4 位队友的概率. $Pr_{P\to P_n}$ 的值为

$$Pr_{P\to P_n}=\frac{F_{P\to P_n}}{\sum_{n=1}^{4}F_{P\to P_n}}\times 100\%, \tag{7}$$

其中 $F_{P\to P_n}$ 表示传球网络中连接球员 P 与 P_n 之间边的属性 FREQUENCY%的值.

例如,以 Stephen Curry 为例,表 13 中的 FREQUENCY 字段是 Stephen Curry 在真实的 NBA 赛场中的传球概率,假设游戏中 Stephen Curry 担任控球后卫,并且其他 4 位队友为:中锋 Bogut、大前锋 Green、小前锋 Iguodala 及得分后卫 Thompson. $F_{P\to P_n}$ 的值如表 15 所示:

Table 15　Stephen Curry's Pass Frequency

表 15　库里传球概率表

Variable	Probability Value	Description
$F_{P\to P_1}$	34.8	The probability of Curry passing the ball to Green in passing network
$F_{P\to P_2}$	19.1	The probability of Curry passing the ball to Thompson in passing network
$F_{P\to P_3}$	11.8	The probability of Curry passing the ball to Bogut in passing network
$F_{P\to P_4}$	7.4	The probability of Curry passing the ball to Iguodala in passing network

那么根据式(7)可计算出在游戏中 Stephen Curry 的传球概率分布情况,如表 16 所示:

Table 16　Stephen Curry's Pass Frequency in the Game

表 16　库里在游戏中传球概率表

Variable	Probability Value	Description
$Pr_{P\to P_1}$	47.6	The probability of Curry passing the ball to Green in the game
$Pr_{P\to P_2}$	26.1	The probability of Curry passing the ball to Thompson in the game
$Pr_{P\to P_3}$	16.2	The probability of Curry passing the ball to Bogut in the game
$Pr_{P\to P_4}$	10.1	The probability of Curry passing the ball to Iguodala in the game

表 16 中的数据表示的是 Curry 与 Green 等其他 4 名球员之间的总体传球概率分布. 无论是在真实的 NBA 赛场还是游戏中,球员的任意一次传球都是属于随机事件,与接球人的位置、防守人的位置、离篮筐的距离、接球人的角色、战术种类等都有很大关系. 而表 16 中的概率值只能作为游戏中球员传球决策计算中的一个参数,并且,传球网络除了可以辅助计算球员在游戏中的传球决策外,在记录游戏中的传球数据的基础上还可以将传球网络嵌入到游戏中,在赛后将传球网络图呈现给游戏者,从而增加游戏者的体验.

5 结论及下一步工作

大数据时代并不代表所有应用数据量都大,大数据最核心的问题怎样挖掘大数据的价值.随着以MapReduce计算模型为核心的大数据技术的不断成熟,其在社交网络、电子商务、金融、公共安全、健康医疗等领域不断发挥着巨大的价值;但是在竞技体育方面,大数据的相关应用研究还处于探索阶段.本文研究范畴属于图的应用,图广泛应用于社交网络、生物信息学、化学信息学、智能交通、舆情监控等等领域.

本文发现篮球比赛中除得分、篮板、助攻、抢断、盖帽、失误、犯规、投篮命中率、出场时间等常规技术统计外,缺乏对传球数据的记录,更缺乏对传球数据的统计分析、数据挖掘及应用方法的研究.本文将球员之间的传球关系进行关联,发现随着传球数据量的不断增加,最终形成一张稳定的传球网络图.所以本文围绕传球数据形成的传球网络进行研究.1)经过传球数据的获取、数据清洗及格式转化、Vertex及Edge表构建的基础上,通过GraphX构建传球网络图,为传球网络的应用打下基础.2)在传球网络可视化研究方面,由于与社交网络的可视化存在诸多的不同,本文研究了区分球员在传球网络中的角色、重要度、传球次数等信息的方法.3)在构建GraphX传球网络图及可视化研究的基础上,本文分析了传球数据对比赛结果的影响、球队传球数据分析、战术人员选择、临场战术制定、传球网络子图及游戏体验等方面的应用.

下一步工作主要集中在4个方面:

1)高效的传球数据获取方法的研究.由于已有的传球数据只能由人工现场统计或录像分析的方法得到,效率低下并且不能避免人为错误.所以,需要与图像识别及视频处理等技术相结合,研究自动从比赛录像中提取传球数据的方法.

2)本地化的基于传球网络的应用开发.中国的CBA甚至是中国男篮,对常规技术统计数据的价值挖掘及应用都停留在初级阶段.所以,研究基于传球网络的应用软件,从而简化传球数据的获取到数据价值转化的过程,是推动篮球竞技数据化、智能化的关键.

3)传球数据与其他技术统计的关联分析.需在已有的传球数据中扩展传球类型、速度、失误概率、传球距离等信息,并挖掘传球数据与球员位置、得分、失误、助攻等其他技术统计数据之间的关联关系.

4)将传球网络的研究与应用扩展到其他球类运动.本文的传球数据来自于篮球赛场,可将传球网络的研究与应用扩展到其他球类运动,如足球、排球、橄榄球、水球、手球等.

参 考 文 献

[1] International Data Corporation. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east [EB/OL]. [2015-12-15]. http://www.emc.com/collateral/analyst-reports/idc-the-digitaluniverse-in-2020.pdf

[2] Meng Xiaofeng, Ci Xiang. Big data management: Concepts, techniques and challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-149 (in Chinese)
(孟小峰, 慈祥. 大数据管理: 概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-149)

[3] Ghemawat S, Gobioff H, Leung S T. The Google gile system [C] //Proc of the 19th ACM Symp on Operating System Principles. New York: ACM, 2003: 29-43

[4] Dean J, Ghemawat S. MapReduce: Simplifed data processing on large clusters [C] //Proc of the 6th Symp on Operating System Design and Implementation. New York: ACM, 2004: 137-150

[5] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets [C] //Proc of the USENIX Conf on Hot Topics in Cloud Computing. Berkeley: USENIX Association, 2010: 1765-1773

[6] Xin R S, Gonzalez J E, Franklin M J, et al. GraphX: A resilient distributed graph system on Spark [C] //Proc of the Int Workshop on Graph Data Management Experiences and Systems. New York: ACM, 2013: 1-6

[7] Gonzalez J E, Xin R S, Dave A, et al. Graphx: Graph processing in a distributed dataflow framework [C] //Proc of the 11th USENIX Symp on Operating Systems Design and Implementation (OSDI'14). Berkeley: USENIX Association, 2014: 599-613

[8] Inokuchi A, Washio T, Motoda H. An apriori-based algorithm for mining frequent substructures from graph data [C] //Proc of the 4th European Conf on Principles of Data Mining and Knowledge Discovery. Berlin: Springer, 2010: 13-23

[9] Kuramochi M, Karypis G. Frequent subgraph discovery [C] //Proc of the 1st IEEE Int Conf on Data Mining. Piscataway, NJ: IEEE, 2001: 313-320

[10] Yan Xifeng, Han Jiawei. Gspan: Graph-based substructure pattern mining [C] //Proc of the 2nd IEEE Int Conf on Data Mining. Piscataway, NJ: IEEE, 2002: 721-724

[11] Huan Jun, Wang Wei, Prins J. Efficient mining of frequent subgraphs in the presence of isomorphism [C] ///Proc of the 2nd IEEE Int Conf on Data Mining. Piscataway, NJ: IEEE, 2003: 549-552

[12] Bhuiyan M A, Hasan M A. An iterative MapReduce based frequent subgraph mining algorithm [J]. IEEE Trans on Knowledge & Data Engineering, 2013, 27(3): 608-620

[13] Lu Wei, Chen Guang, Tung A K H, et al. Efficiently extractiong frequent subgraphs using mapreduce [C] //Proc of the 1st IEEE Int Conf on Big Data. Piscataway, NJ: IEEE, 2013: 639-647

[14] Liao Bin, Yu Jiong, Zhang Tao, et al. Energy-efficient algorithms for distributed file system HDFS [J]. Chinese Journal of Computers, 2013, 36(5): 1047-1064 (in Chinese)
(廖彬, 于炯, 张陶, 等. 基于分布式文件系统 HDFS 的节能算法[J]. 计算机学报, 2013, 36(5): 1047-1064)

[15] Liao Bin, Zhang Tao, Yu Jiong, et al. Energy consumption modeling and optimization analysis for MapReduce [J]. Journal of Computer Research and Development, 2016, 53(9): 2107-2131 (in Chinese)
(廖彬, 张陶, 于炯, 等. MapReduce 能耗建模及优化分析[J]. 计算机研究与发展, 2016, 53(9): 2107-2131)

[16] Righi R D R, Gomes R D Q, Rodrigues V F, et al. MigPF: Towards on self-organizing process rescheduling of bulk-synchronous parallel applications [J/OL]. Future Generation Computer Systems, 2016, 32(5): 1-18 [2016-08-01]. http://www.sciencedirect.com/science/article/pii/S0167739X16301145

[17] Righi R R D, Pilla L L, Carissimi A, et al. MigBSP: A novel migration model for bulk-synchronous parallel processes rescheduling [C] //Proc of the IEEE Int Conf on High Performance Computing and Communications. Piscataway, NJ: IEEE, 2009: 585-590

[18] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A system for large-scale graph processing [C] //Proc of the ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 135-146

[19] Salihoglu S, Widom J. Optimizing graph algorithms on pregel-like systems [J]. Proceedings of the VLDB Endowment, 2014, 7(7): 577-588

[20] Han M, Daudjee K, Ammar K, et al. An experimental comparison of pregel-like graph processing systems [J]. Proceedings of the VLDB Endowment, 2014, 7(12): 1047-1058

[21] Sengupta D, Song S L, Agarwal K, et al. GraphReduce: Processing large-scale graphs on accelerator-based systems [C] //Proc of the 15th Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2015: 1-12

[22] Garimella K, Morales G D F, Gionis A, et al. Scalable facility location for massive graphs on pregel-like systems [C] //Proc of the 24th ACM Int on Conf on Information and Knowledge Management. New York: ACM, 2015: 273-282

[23] Low Y, Bickson D, Gonzalez J, et al. Distributed GraphLab: A framework for machine learning and data mining in the cloud [J]. Proceedings of the VLDB Endowment, 2012, 5(8): 716-727

[24] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing [C] //Proc of the USENIX Conf on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2012: 141-146

[25] Han M, Daudjee K. Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems [J]. Proceedings of the VLDB Endowment, 2015, 8(9): 950-961

[26] Yan Yuliang, Dong Yihong, He Xianmang, et al. FSMBUS: A frequent subgraph mining algorithm in single large-scale graph using Spark [J]. Journal of Computer Research and Development, 2015, 52(8): 1768-1783 (in Chinese)
(严玉良, 董一鸿, 何贤芒, 等. FSMBUS: 一种基于 Spark 的大规模频繁子图挖掘算法[J]. 计算机研究与发展, 2015, 52(8): 1768-1783)

[27] Cervone D, Amour A, Bornn L, et al. POINTWISE: Predicting points and valuing decisions in real time with NBA optical tracking data [C/OL] //Proc of the 8th MIT Sloan Sports Analytics Conf. Cambridge, MA: MIT, 2014: 1-9 [2016-08-01]. http://www.lukebornn.com/papers/cervone_ssac_2014.pdf

[28] Maheswaran R, Chang Y H, Su J, et al. The three dimensions of rebounding [C/OL] //Proc of the 8th MIT Sloan Sports Analytics Conf. Cambridge, MA: MIT, 2014: 1-7 [2016-08-01]. http://www.sloansportsconference.com/wp-content/uploads/2014/02/2014_SSAC_The-Three-Dimensions-Of-Rebounding.pdf

[29] Maymin P. Acceleration in the NBA: Towards an algorithmic taxonomy of basketball plays [C/OL] //Proc of the 7th MIT Sloan Sports Analytics Conf. Cambridge, MA: MIT, 2013: 1-7 [2016-08-01]. http://www.sloansportsconference.com/wp-content/uploads/2013/Slides/RP/Acceleration%20in%20the%20NBA.pdf

[30] Goldman M, Rao J M. Live by the three, die by the three? The price of risk in the NBA [C/OL] //Proc of the 7th MIT Sloan Sports Analytics Conf. Cambridge, MA: MIT, 2013: 1-15 [2016-08-01]. http://www.sloansportsconference.com/wp-content/uploads/2013/Live%20by%20the%20Three,%20Die%20by%20the%20Three%20The%20Price%20of%20Risk%20in%20the%20NBA.pdf

[31] Franks A, Miller A, Bornn L, et al. Counterpoints: Advanced defensive metrics for NBA basketball [C/OL] //Proc of the 9th MIT Sloan Sports Analytics Conf. Cambridge, MA: MIT, 2015: 1-8 [2016-08-01]. http://www.lukebornn.com/papers/franks_ssac_2015.pdf

[32] Wiens J, Balakrishnan G, Guttag J. To crash or not to crash: A quantitative look at the relationship between offensive rebounding and transition defense in the NBA [C/OL] //Proc of the 7th MIT Sloan Sports Analytics Conf. Cambridge, MA: MIT, 2013: 1-7 [2016-08-01]. http://www.sloansportsconference.com/wp-content/uploads/2013/To%20Crash%20or%20Not%20To%20Crash%20A%20quantitative%20look%20at%20the%20relationship%20between%20offensive%20rebounding%20and%20transition%20defense%20in%20the%20NBA.pdf

[33] Ren Lei, Du Yi, Ma Shuai, et al. Visual analytics towards big data [J]. Journal of Software, 2014, 25(9): 1909-1936 (in Chinese)
(任磊, 杜一, 马帅, 等. 大数据可视分析综述[J]. 软件学报, 2014, 25(9): 1909-1936)
[34] Herman I, Melancon G, Marshall M S. Graph visualization and navigation in information visualization: A survey [J]. IEEE Trans on Visualization and Computer Graphics, 2000, 6(1): 24-43
[35] Zhang Xi, Yuan Xiaoru. Treemap visualization [J]. Journal of Computer-Aided Design & Computer Graphics, 2012, 24(9):1113-1124 (in Chinese)
(张昕, 袁晓如. 树图可视化[J]. 计算机辅助设计与图形学学报, 2012, 24(9): 1113-1124)
[36] Balzer M, Deussen O. Voronoi treemaps [C] //Proc of the IEEE Symp on Information Visualization. Piscataway, NJ: IEEE, 2005: 49-56
[37] Gou Liang, Zhang Xiaolong. Treenetviz: Revealing patterns of networks over tree structures [J]. IEEE Trans on Visualization and Computer Graphics, 2011, 17(12): 2449-2458

Zhang Tao, born in 1988. PhD candidate in the School of Information Science and Engineering, Xinjiang University. Her main research interests include big data and cloud computing, etc.

Yu Jiong, born in 1964. Professor and PhD supervisor in computer science at the School of Information Science and Engineering, Xinjiang University. His main research interests include grid computing, parallel computing, etc.

Liao Bin, born in 1986. PhD and associate professor in the School of Statistics and Information, Xinjiang University of Finance and Economics. His main research interests include database theory and technology, big data and green computing, etc.

Guo Binglei, born in 1991. PhD candidate in the School of Information Science and Engineering, Xinjiang University. Her main research interests include cloud and green computing, etc.

Bian Chen, born in 1981. PhD candidate in the School of Information Science and Engineering, Xinjiang University. His main research interests include parallel computing, distributed system, etc.

Wang Yuefei, born in 1991. PhD candidate in the School of Information Science and Engineering, Xinjiang University. His main research interests include big data and green computing, etc.

Liu Yan, born in 1990. Master candidate in the school of Software, Tsinghua University. His main research interests include cloud and green computing, etc.