武漢大學

研究生学位论文开题报告登记表

学　　院：测绘遥感信息工程国家重点实验室

专　　业：地图制图学与地理信息工程

学　　号：2016206190076

姓　　名：王源

导师姓名：吴华意

导师职称：教授

2018 年　12 月　10 日

武汉大学关于研究生学位论文开题报告的有关规定

根据《中华人民共和国学位条例》及其《暂行实施办法》和《武汉大学学位授予工作细则》的精神，为做好研究生学位论文的开题报告，保证学位论文质量，特作如下规定：

第一条 学位论文开题报告是研究生写作论文的必经过程，所有研究生（含：博士生、硕士生）在修完学位课程，写作学位论文之间都必须作开题报告。

第二条 开题报告主要检验研究生对专业知识的独立驾驭能力和研究能力，考察写作论文准备工作是否深入细致，包括选题是否恰当，资料占有是否翔实、全面，对国内外的研究现状是否了解，本人的研究是否具有开拓、创新性。

第三条 学位论文开题报告前，研究生必须根据专业培养目标，结合导师、教研室（或研究室）所承担的国家、省部委等有关部门下达的研究项目或课题以及本人的研究特长，与导师协商，确定选题，广泛查阅文献，深入调研，收集资料，制定学术研究方案，在此基础上撰写开题报告。

第四条 研究生进行开题报告，必须提交“开题报告”的书面材料，内容包括：（1）论文选题的理由或意义；（2）国内外关于该课题的研究现状及趋势；（3）本人的研究计划，包括研究目标、内容、拟突破的难题或攻克的难关、自己的创新或特色、实验方案或写作计划等；（4）主要参考文献目录。开题报告的书面材料不得少于 3000 字。

第五条 研究生进行学位论文开题报告要向导师提出申请，申请获准后，博士生在博士生指导小组范围内作开题报告，硕士生在导师所在教研室或教学小组作开题报告。参加开题报告的教师，包括导师在内，一般不得少于 3 人。无论博士生还是硕士生，在作开题报告时，本学科专业的研究生一般必须参加，跨学科或相近专业的研究生亦可旁听。

第六条 参加研究生学位论文开题报告的教师应当对开题报告进行评议，主要评议论文的选题是否恰当，研究设想是否合理、可行，研究内容与方法是否具有开拓性、创新性，研究生是否可以开始进行论文写作等。评议结果分“合格”与“不合格”二种。评议结束后，由研究生指导教师在《研究生学位论文开题报告登记表》“评语”栏中填写评语。学位论文开题报告通过后，研究生方可进行论文撰写工作。

第七条 开题报告结束后，研究生应将登记表复印一份连同登记表原件和开题报告等一并交所在院、系研究生干事将登记表复印件加盖公章后报送研究生院培养教育处，其他材料留存院、系查备查。研究生院培养教育处将不定期抽查研究生开题报告材料。

第八条 本规定自 2008 年级研究生开始实行。

第九条 本规定由研究生培养教育处负责解释。

武 汉 大 学 研 究 生 院

研究生学位论文开题报告表

姓　　名	王源	院、系（所）	测绘遥感信息工程国家重点实验室
学 科 专 业	地图制图学与地理信息工程	攻读学位	硕士
研 究 方 向	时空大数据计算与可视化分析	指导教师	吴华意
拟定学位论文题目：面向海量点模式分析的时空 Ripley’s K 函数优化与加速			
参加开题报告教师人数		参加旁听学生人数	

	姓　　名	职　　称	所 在 工 作 单 位
开题报告组成人员			

一、研究背景与意义

点模式分析，旨在一定空间范围内研究点数据的分布模式，以便于更深刻地了解点所代表的现象本身以及产生过程[1]，点模式分析方法在大数据时代蕴含着巨大的应用价值。得益于传感器技术的广泛应用、通讯技术的飞跃发展以及信息基础设施的逐渐成熟，人们能够收集和积累大量数据，并通过这些数据来认知自然景观、社会变化、经济发展、生态环境等现象的产生机制与发展过程。显然这些数据绝大部分离不开 Where 和 When 两个要素，而点是现实世界各类实体与现象的常用表征形式，因此针对海量点的时空分布模式分析，是数据驱动的空间分析与挖掘研究的重要组成部分，也是大数据为 GIS 带来的挑战之一[2]。

Ripley's K 函数（以下正文中简称为“K 函数”）是点模式分析方法的典型代表，已经广泛应用于生物学[3]、传染病学[4]、犯罪学[5-6]、交通学[7]、经济学[8]、社会学[11-12]等领域。相比其他点模式分析方法，K 函数的最大特点在于其将尺度作为输入变量，无需预先选取固定值，从而能够研究点数据在不同尺度下的分布模式，且不会受到可变面元问题（Modifiable Area Unit Problem，MAUP）的影响。自 K 函数被提出以来，在理论上也得到许多拓展，包括维度方面（如空间到时空[13]）、距离度量方面（如欧式距离到路网距离[14]）、应用对象方面（如一元点模式到二元点模式[15]）、点过程假设方面（如均匀点过程到非均匀点过程[16]）等等。这些扩展之间并无排斥，研究人员可根据具体场景因地制宜地设计 K 函数。然而，由于 K 函数计算量较大，在大数据时代，又因为数据规模上升而继续增大，这在一定程度上抬高了 K 函数的应用门槛。

K 函数庞大的计算量一方面来源于自身$O(n^2)$级别的时间复杂度（即计算密集型问题），另一方面也来源于海量点模式分析场景下数据规模的上升（即数据密集型问题）。计算密集型问题可通过算法流程的改进来降低算法复杂度，也可利用并行计算技术，将 K 函数计算任务分解给多核 CPU、众核 GPU，提升计算效率。但是并行计算技术往往需要价格昂贵的高端 CPU 或 GPU，否则一般配置下效果欠佳，因而使用环境受限。数据密集型问题可通过分布式计算技术来解决，将 K 函数划分为不同数据分区，以传输指令而非传输数据的方式实现“分而治之”。随着云计算技术的发展与推广，Hadoop、Spark 等分布式系统使用门槛逐渐降低，这些系统对硬件要求通常不高，借助其实现 K 函

数可在海量数据场景下保持高效率、高伸缩性以及高可靠性。

除 K 函数计算流程与实现方式以外，K 函数的输入参数对其计算效率同样存在影响，包括样本规模、尺度上限、研究区域边界、边界校正方法、模拟点方法等。这些因素在影响 K 函数效率的同时，也会影响 K 函数的计算结果，进而影响到基于 K 函数的分析结论（即 K 函数分析效用）。在实际应用中，需要合理选取 K 函数输入参数，从而在效率与效用间取得平衡。

综上所述，本文以海量点数据为研究对象，从时空模式分析的应用需求出发对 K 函数进行流程优化，并基于 Spark 分布式计算框架进行分布式 K 函数的设计与实现，同时研究 K 函数输入参数对其效率与效用的影响，使 K 函数在海量点时空模式分析应用中使用门槛更低，产生价值更大。

二、国内外研究现状及趋势

2.1 面向计算加速的 Ripley's K 函数算法流程改造与优化

目前针对 K 函数效率提升的工作可分为流程优化与并行加速两部分。其中，流程优化主要从遍历方式与权重复用两方面进行设计，并行加速则是利用 OpenMP、MPI、CUDA 等并行计算框架进行 K 函数实现。

K 函数的核心流程是一个双层遍历，外层遍历选取所有研究区域内的点为中心（下文称为“中心点”），内层遍历将中心点与自身以外的其它点（下文称为“邻近点”）进行距离计算，再根据当前尺度阈值与距离的大小关系过滤邻近点。显然由于尺度阈值这一限制条件，内层遍历存在很多不必要的计算。一个直接有效的方式是对点数据进行预排序，比如先按 X 坐标将点数据分为多个桶，再对每个桶内点按照 Y 坐标排序[17]。在进行内层遍历时，先选取 Y 轴的一个方向与桶内点进行距离计算，一旦 Y 轴距离超过尺度阈值就更换遍历方向，当两个方向都已遍历时，终止桶内遍历。接着，中心点与邻近桶内的点进行距离计算，遍历方式与桶内相同，只是方向由 Y 轴改为 X 轴。通过该优化，可在一定程度上减少内层遍历次数，具体程度与尺度阈值大小有关。

K 函数的另一个主要计算任务是边界校正权重，其复杂度与边界形状、校正方式有关。比如常用的 Isotropic Correction，是一个由中心点位置与距离决定的权重，这意味着当中心点与邻近点的距离相等时，校正权重也相等。那么，以键值对<距离，权重>的形式来记录计算过的权重，保存在各中心点，在遇到距离相等的邻近点时就无需再计算权重。并且，在模拟过程中，如果涉及相同

的中心点，也可以继续复用这些权重[17]。通过该优化，在一定程度上减少了权重计算次数，具体程度与点数据分布状况以及参考点模拟方式有关。

K 函数的并行化基础在于其双层遍历过程是相互独立的，每一次权重的计算不依赖于点对以外的其他输入，于是能够通过 OpenMP、MPI、CUDA 等框架将 K 函数计算分配给多核 CPU、众核 GPU 执行。其中，OpenMP 基于共享内存模型设计，适用于单机多核 CPU；MPI 则通过消息传递模型实现了多机 CPU 的协同计算；CUDA 专为 NVIDIA 硬件设计，通过多级内存结构发挥众核 GPU 的规模优势。OpenMP 和 MPI 已被应用于 K 函数外层遍历和模拟过程的并行[17]，而 CUDA 也被应用于 K 函数内外双层遍历和模拟过程的并行（外层遍历分解至 Block 级别，内部遍历分解至 Thread 级别，利用 Block 内部的 Shared Memory 实现快速数据交换与同步[18]）。

但是，上述优化方法仍存在改进空间。预排序的方式会使遍历过程由于点数据分布状况而表现不稳定（等距桶的点数量可能极度不均），对此可将内层遍历转换为距离查询问题，利用空间/时空索引[19-22]加速遍历过程；现有权重复用的实现方式绑定于观测点模式，其适用的边界校正方法与模拟点方法较为局限，对此可独立设计权重缓存，以点对时空位置为 Key，权重值为 Value，从而适用于更多的 K 函数应用场景。此外，基于并行计算的 K 函数加速方法主要从计算角度发挥硬件效能，却难以应对海量点数据带来的挑战，尤其是当点数据存储于多个结点时，并行计算框架将涉及大量数据传输，拷贝副本的方式也会因数据规模的增大而失效。因此，在流程优化的基础上，利用分布式计算框架设计并实现 K 函数，将能更好地应对大数据时代的挑战。

2.2 基于分布式计算框架的空间数据处理

在大数据时代，以 Hadoop、Spark 为代表的分布式系统已被广泛用于海量数据的收集、清洗、存储、管理、计算、分析以及应用。在此基础上，也出现了许多专为空间数据处理设计的分布式地理计算框架。

基于 Hadoop 的分布式地理计算框架主要包括 SpatialHadoop[23]、HadoopGIS[24]、Parallel-Secondo[25]等。其中，SpatialHadoop 基于 Hadoop 原生 API 实现了空间数据结构（点、线、面、集合等），支持多种空间分区方法（规则格网、R 树、四叉树、KD 树、Hilbert 曲线等），并提供空间索引及可视化功能，此外 SpatialHadoop 的 SQL 扩展 Pigeon 支持 SQL/MM-Part3 标准下的

空间 SQL 查询；HadoopGIS 主要通过基于 SATO（类似 KD 树）的空间分区与本地索引实现高效空间查询，并集成 Hive 来支持命令式空间查询，但是 HadoopGIS 未提供复杂空间数据结构（比如凸/凹多边形），且不支持标准化 SQL；Parallel-Secondo 主要将 Hadoop 与一种支持非标准化数据的可扩展数据库 SECONDO 集成，利用 Hadoop 管理分布式任务，从而实现多结点空间数据库操作，但是该系统仅支持规则分区，无法处理空间数据倾斜问题。上述系统均以 Hadoop 的 MapReduce 框架为基础实现，因此在获得分布式处理能力的同时，也会受到 MapReduce 自身缺陷的影响，即大量的磁盘读写消耗。

基于 Spark 的分布式地理计算框架主要包括 GeoSpark[26]、SpatialSpark[27]、GeoMesa[28]、Magellan[29]、Simba[30]等。其中，GeoSpark 分为 Core、SQL、Viz 三个模块，分别提供 Spark RDD 级别的空间数据结构与操作、Spark DataFrame 级别的标准化 SQL 空间查询以及基于空间操作或查询结果的可视化图片生成，其核心是将 Spark RDD 扩展至 SpatialRDD，通过空间分区与空间索引加速操作，并封装了范围查询、KNN 查询、空间连接等方法；SpatialSpark 主要基于 Spark RDD 实现了空间范围查询与空间连接查询，并通过 R 树分区和 R 树索引来加速查询过程；GeoMesa 是一个开源时空数据库，其中包含的 GeoMesaSpark 模块提供了 Spark RDD 与 DataFrame 级别的接口进行空间数据处理，同样采用了 R 树分区降低额外计算消耗，但是本地索引只是简单的网格索引，于是分区内的处理效率会受到空间数据倾斜的影响；Magellan 主要基于 Spark DataFrame API 扩展了 SparkSQL，支持空间范围查询与空间连接查询，其特别之处在于采用了 Z-Order 曲线索引加速查询，但在空间连接上表现并不理想[31]；Simba 同样是 Spark DataFrame 层面的空间扩展，提供范围查询、距离连接查询、KNN 查询、KNN 连接查询，并利用 R 树分区与 R 树索引加速查询过程。上述系统均基于 Spark RDD 或 DataFrame 实现，其中间结果存储于内存，相比 Hadoop MapReduce 读写消耗小很多，更适合大数据分析应用的开发。

但是目前针对空间数据处理的分布式地理计算框架主要关注底层的空间数据支持以及基础性空间操作的封装，对类似 K 函数的具体空间分析方法实现研究较少。并且，目前已有的数据分区与索引主要围绕空间维度设计，关于时空维度的研究仍在探索中[32-34]。为了时空 K 函数的分布式实现，需要考虑

顾及时空分布的数据分区方法，以保持各分区任务均衡。此外，在数据分区与具体计算过程中，各分区、各结点都无法避免数据传输与交换，而序列化与反序列化是数据传输的必要环节[35-37]，因此还应当针对时空点对象以及时空索引的核心信息定制序列化方法，以减少序列化与反序列化的 CPU 消耗以及数据传输量，从而提高分布式集群的总体计算效率。

2.3 Ripley's K 函数输入参数影响分析

K 函数的输入参数主要包括点数据、边界数据、尺度上限、边界校正方法、模拟点方法。这些参数将同时影响 K 函数的计算效率与计算结果，因此在使用 K 函数时，需要综合考虑输入参数的影响。

点数据代表了研究所面向的对象，其数据量可视为固定值。当数据量过小时（比如小于 100），K 函数的计算结果会存在较大偏差，因为过小的数据规模难以支撑置信度较高的模拟过程[38]。当数据量过大（比如大于数十亿），以至于难以在可容忍时间内完成 K 函数计算，显然应当考虑对数据进行裁剪，具体方式可以是空间/时空抽样、维度过滤、形态化简等等。但是裁剪前后的分析结果是否一致，还需进一步讨论，目前几乎没有相关研究。

边界数据代表了研究区域范围，主要从点密度与边界校正权重两方面影响 K 函数计算效率，此外模拟点方法的处理过程也将受边界影响。常见选择是行政边界或点数据的最小外包矩形，CrimeStat 软件还提供了圆形边界选项[38]。显然边界越精确，K 函数准确性越高，但计算复杂度也越高。基于行政边界与最小外包矩形的 K 函数结果可能存在明显区别，但不同详细程度的行政边界会对 K 函数结果产生何等程度的影响，尚且需要探究。

尺度上限指定了 K 函数所分析的尺度范围，它与尺度步长共同决定 K 函数所计算的具体尺度集合，而集合总量与 K 函数计算复杂度线性相关。尺度上限的选取依据暂无定论，R 中常用的 spatstat 包默认采用 Ripley's rule of thumb（取最小外包矩形较短边的四分之一）与 Large sample rule（取 $\sqrt{1000/(\pi \times \lambda)}$,其中λ为点平均密度）之间的较小值作为 K 函数尺度上限[39]，但也并非适用于所有情况。虽然 K 函数同时将点模式的一阶效应（点的集中趋势，以点平均密度衡量）与二阶效应（点对距离关系，以邻近点期望数目衡量）纳入理论公式，但主要关注的还是点模式的二阶效应，如果点数据的一阶效应过于显著，二阶效应会被掩盖。因此，适当地控制尺度上限将更好地发挥

K 函数分析效用。

边界校正方法决定了 K 函数对边金额效应的校正效果，也决定了 K 函数权重计算的复杂度。常用的边界校正方法有 Border Correction、Isotropic Correction、Translation Correction 等[40]。Border Correction 直接跳过研究区域边缘点的内层遍历，以避免引入估计偏差；Isotropic Correction 将点对所在圆与研究区域相交部分的圆弧比例倒数作为校正权重，该权重只与点对所在圆有关，而与方向无关；Translation Correction 将研究区域按点对向量反向平移的重叠面积比例作为校正权重，该权重只与点对向量有关，与中心点位置无关。三种方式对点数据的认知基础不同，得到的 K 函数结果也将存在差异。目前大多数研究采用 Isotropic Correction，对于几种校正方式的对比研究较少。

模拟点方法确定了观测点模式的参照对象，由于聚集或离散是相对概念，那么不同的参考点模式与观测点模式的对比结果也就不同。K 函数的一个基本理论假设是点产生过程符合 CSR（完全空间随机），于是 K 函数存在一个对应 CSR 的理论值。在实际应用中，通常会采用 Monte Carlo 方式模拟点，保持其数目与观测点相等，对模拟点计算 K 函数值，模拟一定次数后，按一定置信水平选取模拟点 K 函数值上下限，在与观测点 K 函数值比较。Bootstrapping 是另一种常用模拟方式，由于模拟点均取自观测点，因此可以充分复用观测点权重，但使用场景需要进行仔细考虑。比如产业聚集研究中，对不同行业观测点的 K 函数分析，可考虑对全行业观测点进行 Bootstrapping 模拟，在一定程度上能获得更加贴近现实情况的分析结论[41]。此外，当分析维度从空间扩展至时空时，可用的模拟方法还有基于时空不可分辨假设的 Random Labeling。上述不同模拟方法得到的结论显然会有所不同，而对于时空维度的模拟方法对比研究也相对较少。

综合来看，目前 K 函数优化与加速的研究工作中，通常会包含数据量与尺度上限对 K 函数计算效率影响的实验[17-18]，但并考虑这些参数对计算结果的影响；目前 K 函数理论及应用的研究工作中，通常会涉及边界校正方法、模拟点方法的对比或修改，并讨论它们对分析结论的影响[41-42]，但鲜有研究考虑过这些方法在 K 函数计算效率上带来的差异。在海量点模式分析应用中，既只不能关注计算效率的提升幅度而忽视计算结果，也无必要为细微的计算结果差异牺牲计算效率，两者需要取舍与平衡。因此，顾及计算效率与计算结

果的 K 函数输入参数影响分析将有助于实际应用工作开展。

基于上述国内外研究现状及不足，本文将通过时空索引与权重缓存降低 K 函数时间复杂度，利用时空分区与定制序列化提升分布式 K 函数整体计算效率，同时研究输入参数对 K 函数效率与效用的影响，为 K 函数在海量点模式分析中的应用提供参考。

三、研究目标、内容和拟解决的关键问题

3.1 研究目标

本文以时空 Ripley's K 函数估计与模拟算法为理论基础，以 Apache Spark 分布式计算框架为技术支撑，针对海量点模式分析场景下，时空 Ripley's K 函数面临的计算密集型问题与数据密集型问题，从算法流程优化与分布式实现两个角度研究时空 Ripley's K 函数的效率提升算法，为高效海量点模式分析提供支持。同时综合考虑 Ripley's K 入参对于函数计算效率与分析效用的影响，讨论 Ripley's K 函数入参选取策略，供不同应用场景参考借鉴。

3.2 研究内容

本文主要研究内容包含三部分，各部分关联关系如图 1 所示：

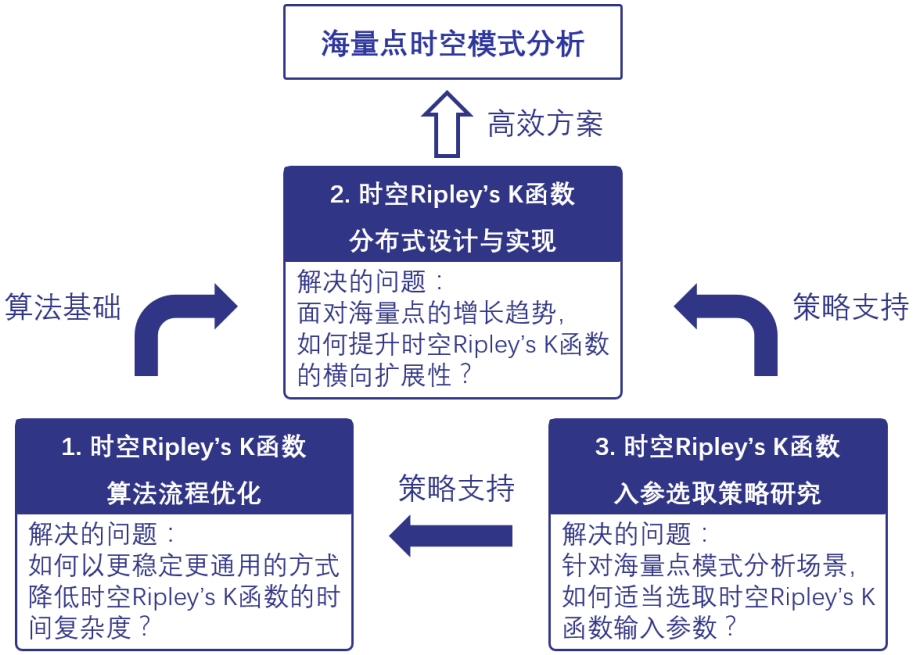

图 1 研究目标与研究内容

1）结合时空索引与权重缓存的时空 Ripley's K 函数算法流程优化

K 函数$O(n^2)$级别的时间复杂度主要来自于研究区域内的所有点对计算，同时 K 函数在不同尺度下的估计与模拟过程涉及大量重复计算。为此，本文将从点对过滤方式与中间值复用两个角度进行流程优化。其中，利用时空索引可将点对距离计算转换为索引结点查询判断（比如最小外包矩形的相交判断），从而大大减少计算负担；通过权重缓存可避免不同尺度下 K 函数估计与模拟过程的重复计算，以一定的空间代价换取时间上的提升。结合以上两部分优化，将降低 K 函数的时间复杂度，从而应对 K 函数的计算密集问题。

2）基于时空分区与定制序列化的时空 Ripley's K 函数分布式设计与实现

在优化 K 函数流程的基础上，本文将进一步研究分布式 K 函数的设计与实现。在分布式环境下，K 函数的整体计算效率受到每一个分区及结点的影响，本文从数据分区与定制序列化两个角度对分布式 K 函数进行优化。其中，基于时空索引思想对海量点数据分区，可充分考虑点数据的时空分布特征，使时空邻近的点数据尽可能均衡地分配在各分区，从而减少分区间的网络消耗以及负载不均带来的整体消耗；根据时空点对象以及时空索引，定制相应的序列化方法，以减少 CPU 消耗以及各分区、各结点之间的数据传输量。结合以上两部分优化，将提升分布式 K 函数的整体计算效率，从而应对 K 函数的数据密集问题。

3）顾及效率与效用的时空 Ripley's K 函数入参选取策略研究

除去计算密集与数据密集问题，K 函数算法的耗时会受到许多因素影响，这包括 K 函数的各类输入设置，即样本规模、尺度上限、研究边界、校正方法、参考点模拟方法。这些因素同时也会影响 K 函数的分析效用，因此不能单纯地为了提升效率而对以上因素进行简化。为了平衡 K 函数的效率与效用，本文将对以上因素开展对比实验，综合考虑入参对计算效率与计算结果的影响，总结入参选取策略，为海量点模式分析的开展提供参考。

3.3 拟解决的关键问题

本文研究的关键问题包括：

1） 基于时空索引的距离查询方法以及面向时空点对的权重缓存设计

内层遍历与权重计算是时空 K 函数算法流程的主要耗时操作，本文针对这两类操作分别进行优化：通过将点对比较转换为基于索引的距离查询问题，

有效减少内层遍历次数；通过将计算过的权重进行缓存，避免相同权重的重复计算。但如何选取时空索引，以尽可能低的额外消耗降低距离查询时间复杂度，是内层遍历优化的难点；如何设计面向时空点对的权重缓存，适应不同模拟点方法，是权重计算优化的难点。

2） 顾及时空分布的数据分区以及面向时空点对象及索引的序列化方法

数据倾斜与网络消耗是分布式算法实现需要应对的主要挑战，本文针对时空 K 函数分布式实现存在的性能瓶颈进行两方面优化：一是顾及点数据的时空分布特征进行数据分区，以平衡各分区任务负载，减少分区网络通信次数；二是面向时空点对象及索引设计序列化方法，减少 CPU 消耗与数据传输量。但如何选取数据分区依据，使分区结果与尺度上限相适应，是数据分区的难点；如何设计更简洁的二进制序列，尽可能地降低序列化与反序列化时间复杂度，并减少数据传输量，是序列化方法的难点。

3） 时空 K 函数计算效率与计算结果的综合评价方法

为了研究 K 函数入参对计算效率与计算结果两方面的影响，需要设计一种综合评价方法，来衡量 K 函数计算效率与计算结果。但 K 函数计算结果的有效性或价值一般难以量化，因此如何设计合适的规则对计算结果进行定性评价，以及如何与计算效率结合，得到综合评价结果是该研究内容的难点。

四、拟采取的技术路线及可行性分析

4.1 技术路线

本文以时空 K 函数的优化与加速为主线，先从内层遍历与权重计算两个角度，优化时空 K 函数算法流程；在此基础上，进行时空 K 函数分布式设计与实现，利用时空分区与定制序列化，进一步提升分布式时空 K 函数的横向扩展性；同时开展控制变量实验，综合考虑时空 K 函数入参对计算效率与计算结果的影响，总结入参选取策略，为时空 K 函数在海量点模式分析中的应用提供参考。总体技术路线如图 2 所示。

4.1.1 时空 Ripley’s K 函数算法流程优化

为降低时空 K 函数时间复杂度，从理论上提升时空 K 函数的计算效率，本文针对时空 K 函数最为耗时的内层遍历与权重计算进行流程优化。对于内层遍历，本文拟采用基于时空索引的距离查询代替点对距离计算，使得内层遍

历无需按点执行，而是基于时空索引的树结构进行范围比较，大大减少遍历次数。对于距离查询得到的邻近点，需要进行权重计算。为避免重复计算，设计双层哈希表缓存对权重进行复用。以 Isotropic Correction 为例，外层哈希表以中心点空间/时间位置为 Key，以内层哈希表为 Value，而内层哈希表以空间/时间距离为 Key，空间/时间权重为 Value。由于哈希表的查询时间复杂度为常数级，将显著减少权重计算耗费时间。同时由于双层哈希表与观测点相互独立，因而可用于 Monte Carlo、Bootstrapping、Random Labeling 等多种模拟点方法。

4.1.2 时空 Ripley's K 函数分布式设计与实现

为应对海量点数据的规模增长，在时空 K 函数的优化流程基础上，进行分布式设计与实现。在分布式计算框架中，任务主要以分区为单位执行。如果各分区任务负载不均，将会拖累整个集群的效率，同时如果分区任务需要其他分区的数据，将出现较大的网络通信消耗。为提高集群总体计算效率，本文从数据分区与序列化两个角度对分布式实现进行改进。时空 K 函数计算任务与点数目密切相关，且计算主要发生在邻近点对上，因此通过顾及时空分布的数据分区，将时空邻近的点数据均衡地划分在不同分区，将有效提升时空 K 函数总体计算效率。利用叶子结点元素相近的时空索引，可获得所需要的时空分区。对于无法避免的数据交换，本文将对时空 K 函数所需的时空点对象及时空索引定制序列化方法，简化二进制序列，只保留必要信息，以降低序列化与反序列化过程的 CPU 消耗，同时也减少数据传输量。

4.1.3 时空 Ripley's K 函数入参选取策略研究

为了研究时空 K 函数不同入参对计算效率与计算结果的影响，本文将开展一系列控制变量实验。当点规模、边界数据、时空尺度上限、边界校正方法、点模拟方法其中之一为自变量时，其余参数保持不变，因变量包括时空 K 函数计算耗时以及时空 K 函数结果图。在时空 K 函数结果差异较为细微时，认为计算耗时越短的参数设置越推荐作为参考值。如果时空 K 函数结果差异较大，比如点模式表现出显著时空聚集的范围或程度存在明显不同，则根据实际数据状况及其他必要知识判断何种参数设置更加合理。基于控制变量实验的结果，本文将总结出一定规则，作为入参选取策略，供时空 K 函数在海量点模式分析中的应用参考。

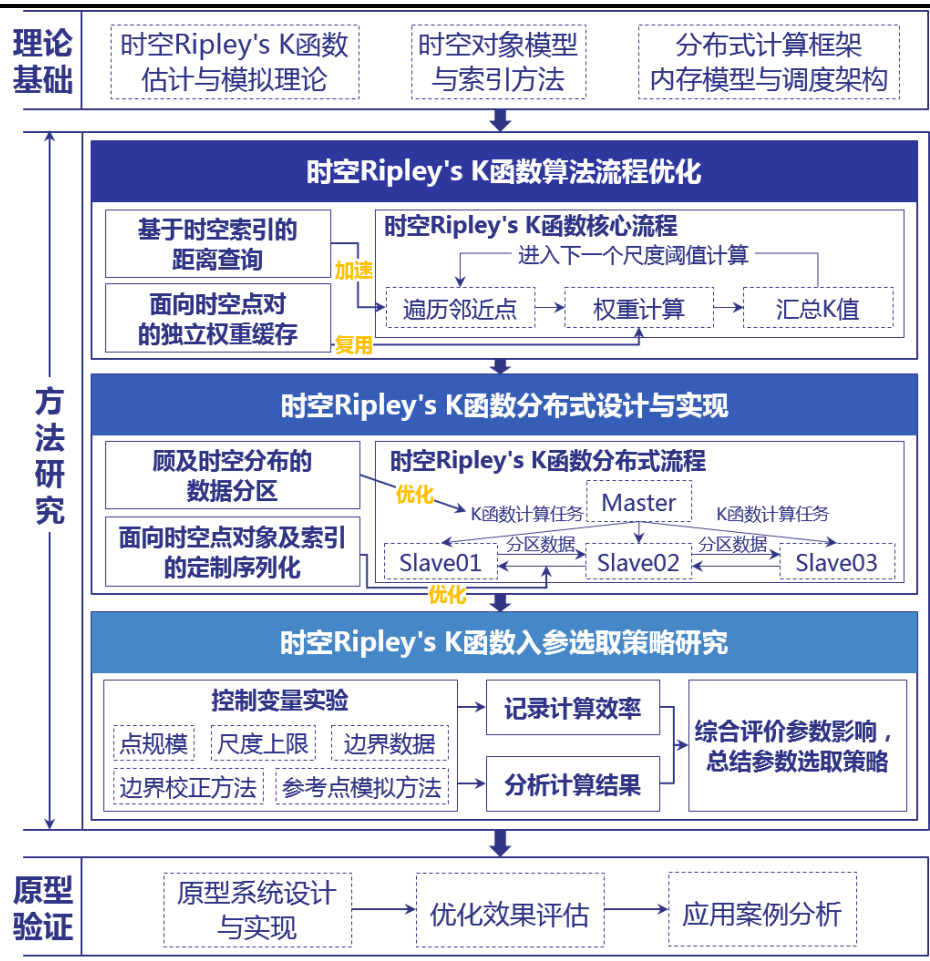

图 2 总体技术路线

4.2 实验方案

4.2.1 实验数据

本文以全国工商企业登记数据作为实验数据，覆盖全国 32 个省级行政区（港澳台地区数据缺失），时间跨度为 1990 年至 2016 年，共 1600 万条记录，包含企业名称、注册日期、经纬度、企业类型、经营范围等字段。实验过程中，主要选取企业经纬度、注册日期、企业类型字段用于 K 函数计算，用来分析不同时空尺度下不同行业企业点的分布模式。

4.2.2 数据预处理

为便于 K 函数实验开展，本文对实验数据进行以下预处理：一是经纬度地理坐标系统统一，原始数据包含的经纬度，是通过对企业注册地址进行地理编码后获得，其坐标系受到地理编码 API 影响，本文将所有经纬度统一为

WGS-84 坐标系，以便于投影坐标计算以及可视化工作；二是时间格式统一，原始数据包含的注册日期存在粒度差异，为便于后续开展不同时间尺度下的 K 函数实验，统一为“YYYY-MM-DD HH:MM:SS”格式，细粒度时间信息缺失的数据采用默认值“2001-06-30 00:00:00”；三是行业类别统一，原始数据所包含的行业类别存在缺失，为便于后续开展不同行业点数据的 K 函数实验，统一按照《国民经济行业分类与代码（GB/T 4754-94）》进行修补，实验将主要使用一级行业类别。

4.2.3 实验设计

1）Ripley’s K 函数算法流程优化对比实验

本实验主要用于衡量时空索引与权重缓存给 K 函数带来的效率提升，以及带来的额外消耗，因此本实验将以消耗时间与内存占用为主要衡量指标。为更好体现 K 函数时间复杂度的降低程度，实验采用数据规模作为自变量，以上述衡量指标为因变量，以优化方法为控制变量，进行原始 K 函数、时空索引优化的 K 函数、权重缓存优化的 K 函数以及结合两项优化的 K 函数之间的对比。根据该实验结果，可验证 K 函数流程优化方法的有效性，并量化评价优化方法的效率提升以及额外消耗。

2）分布式 Ripley’s K 函数设计优化对比实验

本实验主要用于衡量时空分区与调度策略给分布式 K 函数带来的效率提升，以及带来的额外消耗，故而本实验同样以消耗时间与内存占用为主要衡量指标。为更好体现分布式 K 函数的横向扩展性，实验将采用数据规模与节点规模作为自变量，以衡量指标为因变量，以优化方法为控制变量，进行原始分布式 K 函数、时空分区优化的分布式 K 函数、调度策略优化的分布式 K 函数以及结合两项优化的分布式 K 函数之间的对比。根据实验结果，可验证分布式 K 函数优化方法的有效性，并量化评价优化方法的效率提升以及额外消耗。

3）Ripley’s K 函数输入参数对比实验

本实验主要分析 K 函数的各类输入参数对 K 函数计算效率与分析效用的影响，以便总结出 K 函数的配置策略参考。由于不同输入参数的影响不同，实验设计也有相应的差异。若影响因素为数值类型（样本规模、尺度上限与步长），则以影响因素为自变量，K 函数耗时与计算结果作为因变量；若影响因素为枚举类型（区域边界、边界效应校正方法、参考点模拟方法），则以数据

规模为自变量，K 函数耗时与计算结果为因变量，影响因素为控制变量。根据实验结果，讨论 K 函数输入参数在不同应用场景下的选取策略。

4.3 可行性分析

4.3.1 理论方法可行

目前基于时空索引的范围查询或距离查询已经在许多时空数据研究中得到验证，而时空 K 函数内层遍历过程与距离查询存在较大相似性；基于哈希表的缓存设计也已经广泛应用于各种性能提升场景，包括分布式环境，这些场景与时空权重是吻合的；空间分区在分布式地理计算领域已有广泛研究，有效处理了空间维度的数据倾斜问题，针对时空 K 函数的时空分区具有类似作用；定制序列化是分布式计算框架调优理论的重要组成，已在多种分布式应用中表现出显著性能提升，对时空 K 函数的分区数据交换具有参考意义。综上所述，基于时空索引和时空权重缓存优化时空 K 函数流程，利用时空分区和定制序列化加速时空 K 函数分布式实现，在理论上是可行的。

4.3.2 研究方案与技术路线可行

针对本文的研究方案与技术路线，已在前期调研中查阅了中英文文献以及技术资料，学习了空间数据处理库、时空索引实现参考、分布式计算框架扩展方式等内容，对部分研究内容进行了实现，初步证明其可行性。后续将根据代码编写与实验开展情况，对技术路线进一步细化与完善。

4.3.3 实验数据与使用环境可行

实验数据方面，目前已完成全国工商企业数据的空间坐标系统一、时间格式统一、类别信息补全，足够支持不同规模、不同类型的时空 K 函数实验。实验环境方面，目前已在私有云集群完成 Spark 分布式计算框架的初步搭建，实例硬件配置可根据实验情况进行动态调整，满足分布式时空 K 函数的横向扩展性实验需求。

五、创新点及特色

1）结合时空索引与独立权重缓存，优化了时空 K 函数算法流程

现有 K 函数流程改造方式存在局限性，基于预排序的内层遍历会由于点分布状况表现不稳定，绑定观测点的权重复用难以适用不同模拟点方法。本文提出的基于时空索引的内层遍历能够适应点数据的时空分布特征，保持较稳

定的遍历效率优化；面向时空点对的权重缓存将时空维度的权重与观测点独立，便于多种模拟点的复用，使时空 K 函数的优化效果适用更多应用场景。经过两项优化，时空 K 函数的时间复杂度得到了降低，在理论上达到了更高的计算效率。

2）基于时空分区与定制序列化，提升了分布式时空 K 函数的横向扩展性

现有的 K 函数并行加速方法存在较严重的数据冗余或数据传输，而现有的分布式地理计算框架主要关注空间数据与基本操作的实现，无法完全满足时空 K 函数的需要。本文提出的顾及时空分布的数据分区能够更均衡地划分海量点数据，同时减少分区间网络通信的可能性，提升了集群整体效率；面向时空点数据及索引的定制序列化方法降低了数据传输时的 CPU 消耗，减少了数据传输量，将无法避免的数据交换带来的影响进一步缩小。经过两项改进，时空 K 函数的分布式实现拥有了更强的横向扩展性，将更好地适应点数据规模的增长。

3）顾及计算效率与计算结果，讨论了时空 K 函数入参选取策略

现有 K 函数研究大多只关注计算效率或计算结果，而较少兼顾两者，这使得 K 函数在海量点的应用缺少参考。本文对时空 K 函数入参开展了一系列对比实验，综合考虑了入参对计算效率与计算结果的影响，并总结了时空 K 函数入参选取策略，以便于时空 K 函数在海量点模式分析中发挥更大价值。

六、论文大纲

一、绪论

1.1 研究背景及意义

1.2 国内外研究现状

1.3 研究目的与内容

1.4 论文的组织结构

二、Ripley’s K 函数算法理论

2.1 空间 Ripley’s K 函数估计与模拟

2.2 时空 Ripley’s K 函数估计与模拟

2.3 时空 Ripley’s K 函数复杂度分析

2.4 本章小结

三、结合时空索引与权重缓存的时空 Ripley’s K 函数算法流程优化

6.2 社交媒体信息时空传播模式可视化分析

6.3 城市交通事件时空聚集模式可视化分析

6.4 本章小结

七、总结与展望

7.1 总结

7.2 展望

参考文献

致谢

七、研究进展安排

起止时间	工作安排	具体内容
2018.09~2018.12	论文调研	查阅文献资料
		提炼研究问题
		梳理研究内容
2018.12~2019.01	论文开题	撰写开题报告
		制作开题 PPT
		完成开题答辩
2019.01~2019.02	开展研究	完成代码编写
		开展对比实验
		整理实验结果
2019.02~2019.03	论文撰写	深化文献综述
		总结方法思想
		分析实验结果
2019.04~2019.05	论文修改与完善	反复讨论修改论文
2019.05	论文答辩	整理成果，完成答辩

八、主要参考文献

[1] 周成虎. 点模式分析[J]. 地理科学进展, 1989, 8(2): 8-11.

[2] 李清泉, 李德仁. 大数据 GIS[J]. 武汉大学学报(信息科学版), 2014, 39(06): 641-644.

[3] Kuuluvainen T, Penttinen A, Leinonen K, et al. Statistical opportunities for comparing stand structural heterogeneity in managed and primeval forests: an example from boreal spruce forest in southern Finland[J]. 1996.

[4] Hohl A, Delmelle E, Tang W, et al. Accelerating the discovery of space-time patterns of infectious diseases using parallel computing[J]. Spatial and spatio-temporal epidemiology, 2016, 19: 10-20.

[5] Pandit K, Bevilacqua E, Mountrakis G, et al. Spatial Analysis of Forest Crimes in Mark Twain National Forest, Missouri[J]. Journal of Geospatial Applications in Natural Resources, 2016, 1(1): 3.

[6] 徐晓, 朱菁玮, 吴玲, 等. 城市入室盗窃时空点模式分析[J]. 测绘与空间地理信息, 2014, 37(8): 15-18.

[7] Ouni F, Belloumi M. Spatio-temporal pattern of vulnerable road user's collisions hot spots and related risk factors for injury severity in Tunisia[J]. Transportation Research Part F: Traffic Psychology and Behaviour, 2018, 56: 477-495.

[8] 葛莹, 蒲英霞, 赵慧慧, 等. 基于边际 K 函数的长三角地区城市群经济空间划分[J]. 地理学报, 2015, 70(4): 528-538.

[9] 刘春霞. 产业地理集中度测度方法研究[J]. 经济地理, 2006, 26(5): 742-747.

[10] 张琳彦. 产业集聚测度方法研究[J]. 技术经济与管理研究, 2015 (6): 113-118.

[11] 郭洁, 吕永强, 沈体雁. 基于点模式分析的城市空间结构研究[J]. 经济地理, 2015, 35(8): 68-74.

[12] 闫庆武, 卞正富, 王桢. 基于空间分析的徐州市居民点分布模式研究[J]. 测绘科学, 2009, 34(5): 160-163.

[13] Diggle P J, Chetwynd A G, Häggkvist R, et al. Second-order analysis of space-time clustering[J]. Statistical methods in medical research, 1995, 4(2): 124-136.

[14] Okabe A, Yamada I. The K‐function method on a network and its computational implementation[J]. Geographical Analysis, 2001, 33(3): 271-290.

[15] Ripley B D. Modelling spatial patterns[J]. Journal of the Royal Statistical Society. Series B (Methodological), 1977: 172-212.

[16] Baddeley A J, Møller J, Waagepetersen R. Non‐and semi‐parametric estimation of interaction in inhomogeneous point patterns[J]. Statistica Neerlandica, 2000, 54(3): 329-350.

[17] Zhang G, Huang Q, Zhu A X, et al. Enabling point pattern analysis on spatial big data using cloud computing: optimizing and accelerating Ripley's K function[J]. International Journal of Geographical Information Science, 2016, 30(11): 2230-2252.

[18] Tang W, Feng W, Jia M. Massively parallel spatial point pattern analysis: Ripley's K function accelerated using graphics processing units[J]. International Journal of Geographical Information Science, 2015, 29(3): 412-439.

[19] Robinson J T. The KDB-tree: a search structure for large multidimensional dynamic indexes[C]//Proceedings of the 1981 ACM SIGMOD international conference on Management of data. ACM, 1981: 10-18.

[20] Leutenegger S T, Lopez M A, Edgington J. STR: A simple and efficient algorithm for R-tree packing[C]//Data Engineering, 1997. Proceedings. 13th international conference on. IEEE, 1997: 497-506.

[21] Giao B C, Anh D T. Improving sort-tile-recusive algorithm for r-tree packing in indexing time series[C]//Computing & Communication Technologies-Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on. IEEE, 2015: 117-122.

[22] 郑祖芳. 分布式并行时空索引技术研究[D]. 中国地质大学, 2014.

[23] Eldawy A, Mokbel M F. Spatialhadoop: A mapreduce framework for spatial data[C]//Data Engineering (ICDE), 2015 IEEE 31st International Conference on. IEEE, 2015: 1352-1363.

[24] Aji A, Wang F, Vo H, et al. Hadoop gis: a high performance spatial data warehousing system over mapreduce[J]. Proceedings of the VLDB Endowment, 2013, 6(11): 1009-1020.

[25] Lu J, Guting R H. Parallel secondo: boosting database engines with hadoop[C]//Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on. IEEE, 2012: 738-743.

[26] Yu J, Zhang Z, Sarwat M. Spatial data management in apache spark: the GeoSpark perspective and beyond[J]. GeoInformatica, 2018: 1-42.

[27] You S, Zhang J, Gruenwald L. Large-scale spatial join query processing in cloud[C]//2015 31st IEEE International Conference on Data Engineering Workshops (ICDEW). IEEE, 2015: 34-41.

[28] Hughes J N, Annex A, Eichelberger C N, et al. Geomesa: a distributed architecture for spatio-temporal fusion[C]//Geospatial Informatics, Fusion, and Motion Video Analytics V. International Society for Optics and Photonics, 2015, 9473: 94730F.

[29] Sriharsha R. Magellan: geospatial analytics on spark[J]. 2015.

[30] Xie D, Li F, Yao B, et al. Simba: Efficient in-memory spatial analytics[C]//Proceedings of the 2016 International Conference on Management of Data. ACM, 2016: 1071-1085.

[31] Eldawy A, Alarabi L, Mokbel M F. Spatial partitioning techniques in SpatialHadoop[J]. Proceedings of the VLDB Endowment, 2015, 8(12): 1602-1605.

[32] Alarabi L, Mokbel M F, Musleh M. St-hadoop: A mapreduce framework for spatio-temporal data[J]. GeoInformatica, 2018, 22(4): 785-813.

[33] Hagedorn S, Götze P, Sattler K U. The STARK framework for spatio-temporal data analytics on spark[J]. Datenbanksysteme für Business, Technologie und Web (BTW 2017), 2017.

[34] Hagedorn S, Räth T. Efficient spatio-temporal event processing with STARK[C]//EDBT. 2017: 570-573.

[35] Opyrchal L, Prakash A. Efficient object serialization in Java[C]//Electronic Commerce and Web-based Applications/Middleware, 1999. Proceedings. 19th IEEE International Conference on Distributed Computing Systems Workshops on. IEEE, 1999: 96-101.

[36] Greanier T. Discover the secrets of the Java Serialization API[J]. Sun Developer Network: http://java. sun. com/developer/technicalArticles/P rogramming/serialization/, July, 2000.

[37] Kryo: a fast and efficient Object Graph Serialization Framework for Java[EB/OL]. https://github.com/EsotericSoftware/kryo.

[38] Levine N. CrimeStat IV: A Spatial Statistics Program for the Analysis of Crime Houston (TX): Ned Levine & Associates/Washington, DC: National Institute of Justice, 2013.

[39] Baddeley A, Turner R. Spatstat: an R package for analyzing spatial point patterns[J]. Journal of statistical software, 2005, 12(6): 1-42.

[40] Baddeley A, Rubak E, Turner R. Spatial point patterns: methodology and applications with R[M]. Chapman and Hall/CRC, 2015.

[41] Kosfeld R, Eckey H F, Lauridsen J. Spatial point pattern analysis and industry concentration[J]. The Annals of Regional Science, 2011, 47(2): 311-328.

[42] Zhu G, Ge Y, Wang H. A modified Ripley's K function to detecting spatial pattern of urban system[C]//Geoinformatics (GEOINFORMATICS), 2013 21st International Conference on. IEEE, 2013: 1-5.

研究生开题报告及指导教师所提问题回答的内容记录：

老师的问题及意见：

1、

2、

3、

开题报告记录人签名：

年　　月　　日

<table>
<tr><td colspan="3">指导教师意见：

指导教师签名：
年　　月　　日</td></tr>
<tr><td rowspan="2">开
题
报
告
评
语</td><td>评议结果</td><td></td></tr>
<tr><td colspan="2">

参加开题报告的教师（3~5 人）签名：

年　　月　　日</td></tr>
</table>

注：评议结果分“合格”或“不合格”。