刘芳旭, 董雷刚. 无线传感网络中基于 MapReduce 的组合 Skyline 查询算法[J]. 智能计算机与应用,2024,14(10):12-24.
DOI:10.20169/j.issn.2095-2163.241002

无线传感网络中基于 MapReduce 的组合 Skyline 查询算法

刘芳旭[1], 董雷刚[2]
(1 吉林化工学院 信息与控制工程学院, 吉林 吉林 132022; 2 白城师范学院 计算机科学学院, 吉林 白城 137000)

摘 要: 无线传感网络通过传感器节点能够收集到海量数据,利用组合 Skyline 查询技术可以在海量数据中获取以组合为单位的用户感兴趣的信息。然而,由于无线传感网络所处环境的不确定性及节点能量有限等问题,使得在海量数据下组合 Skyline 查询效率不太理想。针对该问题,提出一种基于 MapReduce 的组合 Skyline 查询算法(MR-GSKY 算法),首先通过预处理操作去除无用点,然后将数据集分块,充分利用 MapReduce 分布式计算的特点,在 Map 阶段并行计算每一分块的键值对,再利用删减操作去除无用候选组合,在 Reduce 阶段执行扩展操作对不同键值对的组合进行整合并计算出组合 Skyline 的中间结果,通过多次 Map 和 Reduce 操作生成 G-Skyline(n)。实验结果表明,该算法比现有算法具有更好的性能。

关键词: 无线传感网络; 组合 Skyline; MapReduce; 海量数据; 分布式计算系统
中图分类号: TP316.4 **文献标志码**: A **文章编号**: 2095-2163(2024)10-0012-13

G-Skyline query algorithm based on MapReduce in wireless sensor network

LIU Fangxu[1], DONG Leigang[2]
(1 School of Information and Control Engineering, Jilin Institute of Chemical Technology, Jilin 132022, Jilin, China;
2 School of Computer Science, Baicheng Normal University, Baicheng 137000, Jilin, China)

Abstract: The wireless sensor network can collect a large amount of data through the sensor nodes, and G-Skyline query technology can obtain the user's interest information in the massive data. However, due to the uncertainty of the environment in which the wireless sensor network is located and the limited node energy, the efficiency of G-Skyline query under massive data is not ideal. To solve this problem, G-Skyline query algorithm based on MapReduce(MR-GSKY) is proposed. Firstly, the research removes the useless points through preprocessing operations. Then, the dataset is divided into blocks, making full use of the characteristics of MapReduce distributed computing, and the key-value pairs of each block are calculated in parallel in the Map phase. Subsequently, the deletion operation is used to remove useless candidate combinations. In the Reduce phase, the expansion operation is performed to integrate the combinations of different key-value pairs, and the intermediate results of G-Skyline are calculated, and G-Skyline(n) are generated through multiple Map and Reduce operations. Experimental results show that the proposed algorithm has better performance than existing algorithms.

Key words: wireless sensor network; G-Skyline; MapReduce; massive data; distributed computer system

0 引 言

随着物联网、大数据等技术的发展,无线传感网络广泛应用于环境监测、智能家居、智能交通等多个领域。无线传感网络(Wireless Sensor Network, WSN)是由部署在监测区域内大量分布式传感器节点组成的网络,通过无线通信协作地感知、采集、处理和传输覆盖地理区域内被感知对象的信息,从而收集大量的数据信息,如何高效地处理和分析海量数据成为研究者们关注的焦点。

组合 Skyline 查询作为一种多维数据分析的重要方法,能够返回用户感兴趣的数据点集合,满足用户对数据的高级查询需求。在无线传感网络中,组合 Skyline 查询可以满足实时监测、环境感知等应用

基金项目: 吉林省自然科学基金项目(YDZJ202201ZYTS666); 吉林省教育厅科学研究项目(202301042051)。
作者简介: 刘芳旭(1999-),女,硕士研究生,主要研究方向:数据查询与优化, Email:liufangxu2022@163.com; 董雷刚(1982-),男,博士,副教授,硕士生导师,主要研究方向:数据查询与优化。
收稿日期: 2024-03-18

需求。例如，在森林火险实时监测传感网中，数千个传感器被部署在监测区域内，通过无线通信的方式获取数据信息，从而实现对森林区域的实时监测。森林防火人员可以通过得到的数据信息，判断节点所在的区域是否存在火灾隐患，从而帮助工作人员及时采取应对措施。但由于防火人员数量有限，对单个点进行排查的效率低下，因此可以使用组合 Skyline 查询获得可疑节点分组情况，使防火人员能分工巡查多个最具隐患的位置，从而实现最优化的火情排查。例如，在监测区域内分布着 10 个传感器节点，每个传感器节点采集 2 类信息：逆温度和湿度，见表 1。逆温度和湿度的值越小，意味着发生火灾的可能性越大。通过对这些传感器节点采集的信息进行 Skyline 查询，能够返回湿度最低、逆温度最低和逆温度和湿度都比较低的点，如图 1 所示，$P1$，$P3$，$P5$ 是 Skyline 点，是最有可能发生火灾的可疑点。防火人员根据 Skyline 查询结果对可疑点所在区域进行火险排查。为了实现排查效率的最优化，可以将可疑点分成若干组，每个组由一支队伍进行排查。例如，对传感器节点进行 3-组合 Skyline 查询，返回结果有（$P1$，$P3$，$P5$），（$P1$，$P5$，$P6$），（$P1$，$P2$，$P3$），（$P1$，$P3$，$P4$），（$P3$，$P4$，$P5$），（$P3$，$P5$，$P6$），消防队可以向每个组合派出一支队伍进行排查，以提高人员的工作效率。

表 1　传感器节点数据

Table 1　Sensor node data

传感器节点	逆温度	湿度	传感器节点	逆温度	湿度
$P1$	10	420	$P6$	50	200
$P2$	20	450	$P7$	58	355
$P3$	20	280	$P8$	65	370
$P4$	30	400	$P9$	75	380
$P5$	45	100	$P10$	70	340

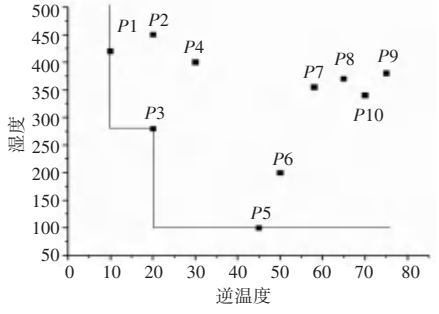


图 1　Skyline 轮廓示例

Fig. 1　Example of Skyline

然而，组合 Skyline 查询技术应用于无线传感网络时，由于网络所处环境的不确定性，节点能量和传输带宽有限等原因，使得某些节点信息收集缓慢，收集到的信息无法及时传输，影响整个网络的数据收集和传输效率，从而影响组合 Skyline 查询的效率。针对此问题，提出一种基于 MapReduce 的组合 Skyline 查询算法（MR-GSKY）。MapReduce 是由 Google 公司提出的一种分布式数据处理模型，是用户开发“基于 Hadoop 的数据分析应用”的核心框架。通过采用“分而治之”的思想，将大规模数据集分块并分配不同节点共同完成，通过整合各个节点的结果，得到最终结果。这种并行处理数据的特点，不仅能够提高数据处理的效率，而且可以减少传感器节点之间的数据传输量和通信开销。

1　研究现状

1.1　Skyline 查询

2001 年，Borzsonyi 等学者[1]首次引入了 Skyline 操作，并提出了计算 Skyline 的 2 个算法，分别是 BNL 和 D&C 算法。BNL[1]又被称为暴力算法，使用嵌套循环的方式对 2 个点进行遍历和比较。其优点是操作简单，缺点是在较大数据集中表现欠佳。D&C[1]是根据内存将数据集分为若干个子集，使每个分区都在内存之中。分别计算每个子集的 Skyline 结果，最后再将各子集的结果合并。SFS[2]首先根据单调函数对整体数据进行排序，再计算 Skyline 元组。与 BNL 相比，该算法能够提前返回一些 Skyline 点，减少后续需要比较的点的数量。Bitmap[3]是一种位图算法，是把所有数据点转化为向量，合起来就形成一个位图，再进行与运算判断是否为 SP 点。但该算法执行复杂且不适用于动态数据。2015 年，Gao 等学者[4]提出 NN 算法，采用 R 树索引结构，用最邻近搜索修剪数据元组计算 Skyline。由于需要频繁访问 R 树，特别是在高维数据中，使得其 I/O 开销很大。BBS[5]与 NN 不同，BBS 仅需遍历一次 R 树，使得 I/O 开销保持在最小。Liu 等学者[6]提出了一种查询窗口，通过改变查询窗口修剪大量无用数据点来计算 Skyline。上面提到的这些算法都是在单一服务器中进行计算的。但在大数据环境下，使用传统的单机计算 Skyline 效率低下，已经无法满足要求。因此，提出了分布式环境下的 Skyline 查询[7-9]。MapReduce 因其良好的扩展性和高效的数据处理能力等特点，在分布式 Skyline 查询中广泛应用。2011 年，Zhang 等学者[10]基于 MapReduce 框架对传统 Skyline 查询中的 BNL、SFS 和 Bitmap 进行改进，提出了相应的有

效算法。针对 MapReduce 特点及负载平衡问题提出了不同的数据划分策略,2012 年 Chen 等学者[11]提出一种在 MapReduce 框架下基于角度划分数据集的 Skyline 查询算法。但在划分之前需要执行一个复杂的转换过程。2014 年,王淑艳等学者[12]提出一种在 MapReduce 框架下基于超平面投影划分数据集的 Skyline 查询算法,与基于角度的划分方法相比,该方法在划分前的坐标转换比较简单。由于传感器节点的能量、计算、存储和通信能力都比较有限,后来,很多研究人员针对无线传感网络特性提出了一系列 Skyline 查询处理算法[13-22],如连续反向 Skyline 查询算法、分布式动态 Skyline 查询算法、概率 Skyline 查询算法和空间 Skyline 查询算法、近似 Skyline 查询算法等。

1.2 组合 Skyline 查询

目前,对于计算最优点组主要有 2 种方法。一种是基于向量的聚合算法[23-26],2012 年 IM 等学者[23]提出利用聚集函数研究组合 Skyline 问题,提出了 Glncremental 算法,相当于一种近似组合 Skyline 算法,随着组合基数的增大,在查询过程中有可能会漏掉一些组合 Skyline 结果。针对这一缺陷,提出了 G-Dynamic 算法,这是一个填充组合 Skyline 的动态算法,同样以渐进的方式生成最终结果。2012 年,Li 等学者[24]提出用输入修剪,输出压缩和搜索空间修剪快速计算组合 Skyline。2014 年,Zhang 等学者[25]提出了基于 OSM 和 WCM 的动态规划算法和迭代算法计算组合 Skyline。Chung 等学者[26]利用 k 元组对应属性上的聚合函数(*SUM*、*MAX* 或 *MIN*)计算向量的分量,再通过相应的聚合向量对元组进行比较。这类方法最大的缺陷就是用户很难选择一个适当的聚合函数,同时在执行过程中容易漏掉能够成为最终结果的重要群体。

另一种就是组合 Skyline 算法。为了返回所有可能的组合 Skyline 查询结果,2015 年 Liu 等学者[27]给出了组合 Skyline 的定义,即将 Skyline 的原始定义推广到组的 Skyline。并提出了计算组合 Skyline 的有效算法:PointWise、UnitWise、UnitWise+算法。运算后可以返回所有的 Pareto 最优组。但是计算起来也需要大量的时间。2017 年,Yu 等学者[28]利用并行搜索及子空间 Skyline 特性构造 MSL 并提出相应组合 Skyline 查询算法。2018 年,Wang 等学者[29]针对 DSG 图还存在冗余点,提出了一个最小支撑结构(Minmum Directed Graph,MDG),用于计算组合 Skyline。2019 年,Xu 等学者[30]提出了关于组合 Skyline 的新引理和分层优化的思想提高 GSKY 的查询性能。2020 年,Yang 等学者[31]提出了一种基于 Skyline 层的 Top-k 组 Skyline 查询方法,通过剪枝 Skyline 高层上的点来实现查询的优化和加速。2021 年,Liu 等学者[32]在文献[27]的基础上,提出根据支配点的数量和支配组的数量选择前 k 个代表性的 G-Skyline 组,并提出了相关的有效算法。2022 年,Han 等学者[33]基于预分类和重用元组提出 GPR 算法,以计算海量数据下的 G-Skyline[34]。针对于并行计算组合 Skyline,Wang 等学者[35]提出基于 MapReduce 分布式计算组合 Skyline。第一个 MapReduce 生成所有可能的 G-Skyline(n),第二个 MapReduce 对所有可能的 G-Skyline(n)分块,并行删减掉被支配的组合,最后再合并生成最终结果。该算法在低维中表现良好,但还需要进一步改进算法来适用于高维组合 Skyline 计算。

2 基础知识

2.1 定义及定理

定义 1 支配 给定一个 d 维空间中包含 n 个点的数据集 P,假设数据集 P 中存在 p 和 q 两点,如果点 p 在每个维度上都不比 q 差,并且点 p 至少在其中一个维度上优于点 q,那么这 2 个点之间存在支配关系[1]。用“$\prec$”表示支配,即 $p \prec q$。

定义 2 Skyline 查询 给定一个 d 维空间中包含 n 个点的数据集 P,如果 P 为 Skyline 查询结果,那么 P 中包含的 n 个点一定是不被其他任何点所支配的点[1]。

性质 1 传递性 如果存在点 p 支配点 q,点 q 支配点 s,那么点 p 一定也支配点 s。

定义 3 组合基数 一个组合中包含数据点的个数称为该组合的组合基数。

定义 4 组支配 给定 2 个组合基数大小为 n 的组 $G=(P1,P2,\cdots,Pn)$ 和 $G'=(P1',P2',\cdots,Pn')$,如果能找到 G 和 G' 中关于 n 个点的 2 个全排列 $G=(P_{u_1},P_{u_2},\cdots,P_{u_n})$ 和 $G'=(P_{v_1},P_{v_2},\cdots,P_{v_n})$,使得对所有 $i(1\leqslant i\leqslant n)$ 有 $P_{u_i}\prec P_{v_i}$,并且至少在某一点 j 上有 P_{u_j} 优于 $P_{v_j}(1\leqslant j\leqslant n)$,则称 G 组支配 G'[27]。

定义 5 组合 Skyline 查询 组合 Skyline 查询用于返回所有不被其他任意组合基数大小相同的组支配的组[27]。用 G-Skyline(n) 表示组合基数为 n 的组合 Skyline 查询结果。

定义 6　Skyline 层　给定 d 维空间中包含 n 个点的数据集 P，Skyline 层包含了 P 中所有的点且这些点分布在不同的层中。其中，第一层中的点为 Skyline 点，包含了所有在数据集 P 中不被其他任意点所支配的点，即 Layer1 = Skyline(P)。第二层中的点是在数据集 P 中除去第一层以外的点，以此类推，第 i 层中的点是在数据集 P 中除去第 $i-1$ 层以外的点[27]。

分析图 1 可知，该图为原始数据集的 Skyline 示例，其中 $P1$、$P3$ 和 $P5$ 为 Layer1 中的点。

定理 1　如果 $G=(P1,P2,P3,\cdots,Pn)$ 是一个组合基数为 n 的组合 Skyline，那么该组中的所有点一定在前 n 个 Skyline 层中[27]。

定义 7　父集　给定一个点 $p \in G$，支配点 p 的点构成的集合称作 p 的父集。其中，父集中的任意一点都是点 p 的一个父节点[27]。

定义 8　子集　给定一个点 $p \in G$，由 p 所支配的点构成的集合称作 p 的子集。其中，子集中的任意一点都是点 p 的孩子节点[27]。

定理 2　对于组合 Skyline 中的每个点，其所有父亲都要在该组合中[27]。

定义 9　Directed Skyline Graph(DSG)　DSG 图是一种数据结构，用于表示前 n 个 Skyline 层的点以及这些点的支配关系。图中每一条边代表两点之间的支配关系，每个节点的结构为[层索引，点索引，父亲，孩子]，其中层索引表示点所在的 Skyline 层，点索引表示该点所在的点的索引值，父亲包括支配该点的所有点，孩子包括由该点支配的所有点[27]。

定理 3　组合 Skyline 验证定理　P 是一组 d 维数据集中的数据点，A 是 P 的一个组合基数为 n 的组合，如果 A 中的每一个点 $a \in A$，在 A 之外都不存在支配 a 的点，那么 A 是一个组合 Skyline[27]。

该定理提供了一个有效方法来验证一个组合是否是组合 Skyline。给定一个组合 A，对每个点 $a \in A$，如果能证明支配 a 的点都在 A 中，就可以肯定 A 是一个组合 Skyline。反之亦然。

推论 1　设 P 是一组 d 维数据集中的数据点，A 是 P 的一个组合基数为 n 的组合，若 A 是一个组合 Skyline，则对于每个点 $a \in A$，P 中支配 a 的点少于 n 个[27]。

2.2　MapReduce 框架

MapReduce 是一个分布式数据处理架构。MapReduce 主要分为 2 个阶段：Map 阶段和 Reduce 阶段[36]。工作流程如图 2 所示。

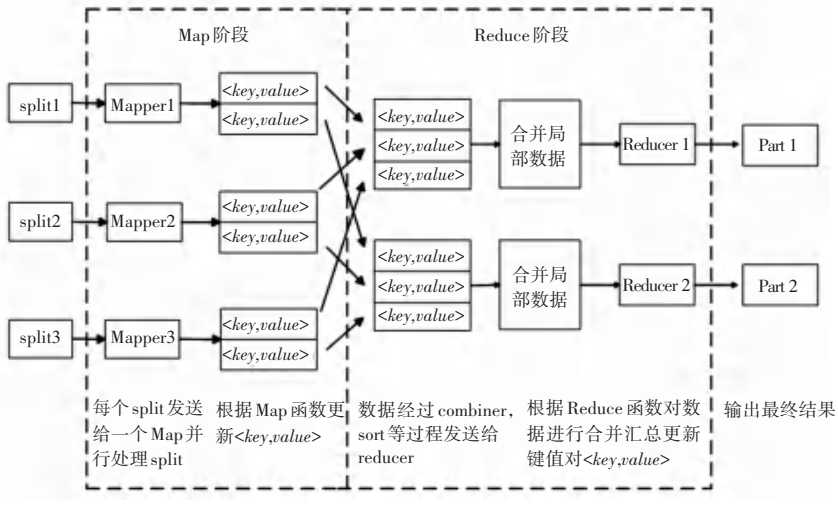


图 2　MapReduce 工作流程

Fig. 2　Workflow of MapReduce

由图 2 可知，Map 阶段负责接收每一个 split 分块，多个 mapper 并行处理，对每个 split 分块应用 *Map* 函数，对键值对 $<key,value>$ 进行更新。在 Reduce 阶段，数据经过一系列 combiner、sort 等过程发送给各个 reducer，通过对每个 reducer 应用 *Reduce* 函数再对每一个键值对结果进行整合汇总操作，输出最终结果。

3　基于 MapReduce 的组合 Skyline 算法

3.1　算法基本思想

本文提出基于 MapReduce 的组合 Skyline 查询算法(MR-GSKY 算法)，算法流程如图 3 所示。

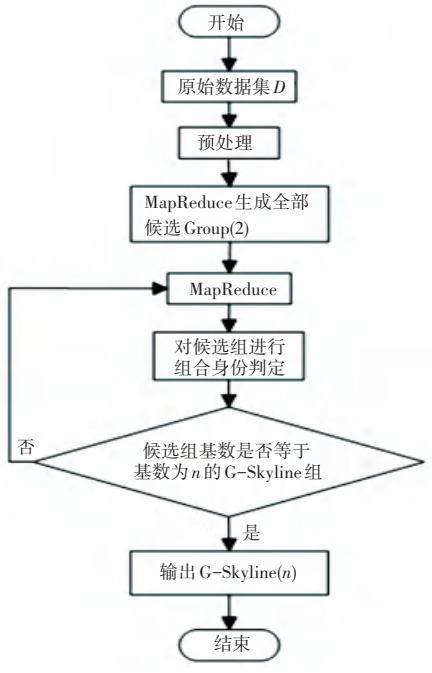


图 3 算法流程图

Fig. 3 Flow chart of the algorithm

以 MapReduce“分而治之”的思想为启发,根据框架特点将组合 Skyline 查询分为 3 个阶段:预处理阶段、计算阶段和汇总阶段。首先原始数据集 D 经过预处理阶段过滤掉明显不符合要求的数据点,减小 MapReduce 中需要处理的数据集大小。计算阶段和汇总阶段均在 MapReduce 框架中完成,用于生成候选组并对结果进行身份判定。根据组合基数 n 的大小,确定需要使用 MapReduce 的次数,最终计算出全部 G-Skyline 组。为便于说明,在后续例子中均设 $n = 3$,即求组合基数为 3 的所有 G-Skyline 组,用 G-Skyline(3)表示。

3.2 预处理阶段

由定理 1 可知,要得到 G-Skyline(n),只需要计算 Skyline 层中前 n 层的数据点即可。由定理 2 可知,给定一个数据点 P,如果 $P \in G$,$G \subset$ G-Skyline,则 P 的所有父亲也都包含在 G 中。根据推论 1,如果支配 P 的个数大于 n,则一定不是一个基数为 n 的 G-Skyline 组。基于上述定理和推论,首先对原始数据集进行预处理操作,筛选出数据集中位于 DSG 前 n 层且父亲个数不超过 n 的数据点,以提高在 MapReduce 中组合 Skyline 查询效率。

以表 1 中的数据为例,其对应的 DSG 图如图 4 所示。设组合基数 n 为 3,取 DSG 图前 3 层的数据点,分别是 $P1$、$P2$、$P3$、$P4$、$P5$、$P6$、$P7$、$P10$。其中,$P7$ 和 $P10$ 均被 $P3$、$P5$ 和 $P6$ 支配,即 $P7$ 和 $P10$ 的父亲个数均大于3。根据推论1可知,这 2 个点不会存于一个基数为 3 的 G-Skyline 组中,故可以直接排除。经过预处理后的结果为:$P1$、$P2$、$P3$、$P4$、$P5$ 和 $P6$。可以看到,经过预处理后的数据集比原数据集有明显的减小。

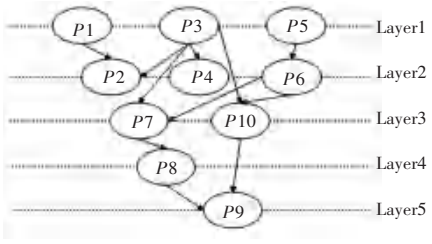


图 4 DSG 图

Fig. 4 Diagram of DSG

3.3 第一个 MapReduce 框架

图 5 为第一个 MapReduce 阶段。在 MapReduce 中,Map 任务的个数可以根据数据集的规模来确定,较大的数据集可以被分成多个小的数据分块,每个分块分别由一个 Map 任务处理生成新的键值对 $< key, value >$。其中,key 值表示该点或组合的父亲个数(单个点的 key 值表示该点的父亲个数,也就是能够支配该点的所有点的个数。组合的 key 值是指未在该组内的点的所有父节点个数);$value$ 表示对应的点或组合。例如,键值对 $< 1,(P1,P2) >$ 表示在($P1$,$P2$) 之外能支配 $P1$ 或 $P2$ 的点的个数为 1。

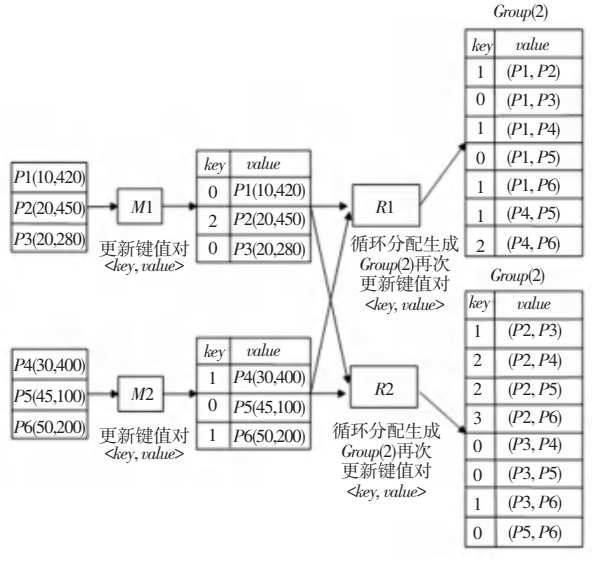


图 5 第一个 MapReduce 阶段

Fig. 5 First MapReduce phase

设 $Map = 2$,$Reduce = 2$。表示在 Map 阶段接收 2 个数据分块,每个分块作为一个 Map 任务输入,分别为 $M1$ 和 $M2$。Reduce 阶段用 $R1$ 和 $R2$ 并行整合输出所有 $Group(2)$。第一个 MapReduce 阶段的基本思想:将预处理操作后的数据集分成 2 块作为第

一个 MapReduce 的输入。在 Map 阶段中，$M1$ 和 $M2$ 分别接收每一个数据分块，应用 *Map* 函数对每个数据点的键值对 $< key, value >$ 进行更新。再分别发送给 Reduce 阶段中的 $R1$ 和 $R2$，通过循环分配并行整合数据生成全部 $Group(2)$。

(1) Map 阶段。预处理后的数据集会被分成多个分块并生成默认键值对 $< key, value >$。每个分块作为一个 Map 任务输入，对每个分块应用自定义 *Map* 函数，*Map* 函数的作用就是将初始键值对进行更新，在新的键值对中 *key* 值表示点的父亲个数，*value* 表示对应的数据点。

以图 5 中 $M1$ 分块为例，$M1$ 中包含 $P1$、$P2$、$P3$，与其相应的父亲个数分别为 0,2,0，更新每个数据点的键值对 $< key, value >$，可得新的键值对分别为 $< 0, P1(1,3) >$，$< 2, P2(2,3) >$，$< 0, P3(2,2) >$。$M2$ 和 $M3$ 执行与 $M1$ 相同操作。

(2) Reduce 阶段。为避免数据倾斜和负载不均衡的问题，在 Reduce 阶段采用了循环分配策略[35]。循环分配是按照一定的规则将数据均匀地分配给各个 Reducer，直到所有数据都分配完毕。确保每个 Reducer 都相对均匀地处理一部分数据，从而高效地发挥并行处理的效果，提高整个 MapReduce 的执行效率。

以图 5 中 $R1$ 为例，从图 5 中可以看到，在 $M1$ 中以 $P1$、$P2$、$P3$ 为前缀生成的 $Group(2)$ 占据较大比重，一共可生成 12 个 $Group(2)$。$M2$ 中以 $P4$、$P5$、$P6$ 为前缀生成的 $Group(2)$ 仅有 3 个。通过对比可以明显看到，如果 $R1$ 直接接收由 $M1$ 中的数据生成的 $Group(2)$，会产生严重负载不均的情况。因此设置 $R1$ 接收 $P1$ 和 $P4$，通过循环分配生成 $(P1,P2)$，$(P1,P3)$，$(P1,P4)$，$(P1,P5)$，$(P1,P6)$，$(P4,P5)$，$(P4,P6)$ 共 7 个组合。$R2$ 接收 $P2$，$P3$，$P5$ 和 $P6$，通过循环分配生成 $(P2,P3)$，$(P2,P4)$，$(P2,P5)$，$(P2,P6)$，$(P3,P4)$，$(P3,P5)$，$(P3,P6)$，$(P5,P6)$ 共 8 个组合。

3.4　第 *m* 个 MapReduce 阶段

在第一个 MapReduce 阶段求得全部 $Group(2)$ 之后，从第二个 MapReduce 阶段开始，每一阶段都采用相同的过滤删减策略和扩展策略进行计算，直到获得全部 G-Skyline 组。在第 m 个 MapReduce 阶段中，首先将上一阶段的计算结果划分为若干块，然后针对所有的块同时进行 Map 操作，并行对每个分块执行过滤删减，删除被支配的 $Group(m)$ 及 *key* 值大于 $n-m$ 的 $Group(m)$。接下来，将各分块处理后的候选 $Group(m)$ 发送到 Reduce 阶段生成全局候选集，对全局候选集再次执行过滤删减操作，最终执行扩展操作生成 $G-Skyline(n)$。图 6 为当 $n=3$ 时，第 2 个 MapReduce 阶段执行过滤删减操作和扩展操作生成的全部 G-Skyline 组，即 G-Skyline(3)。

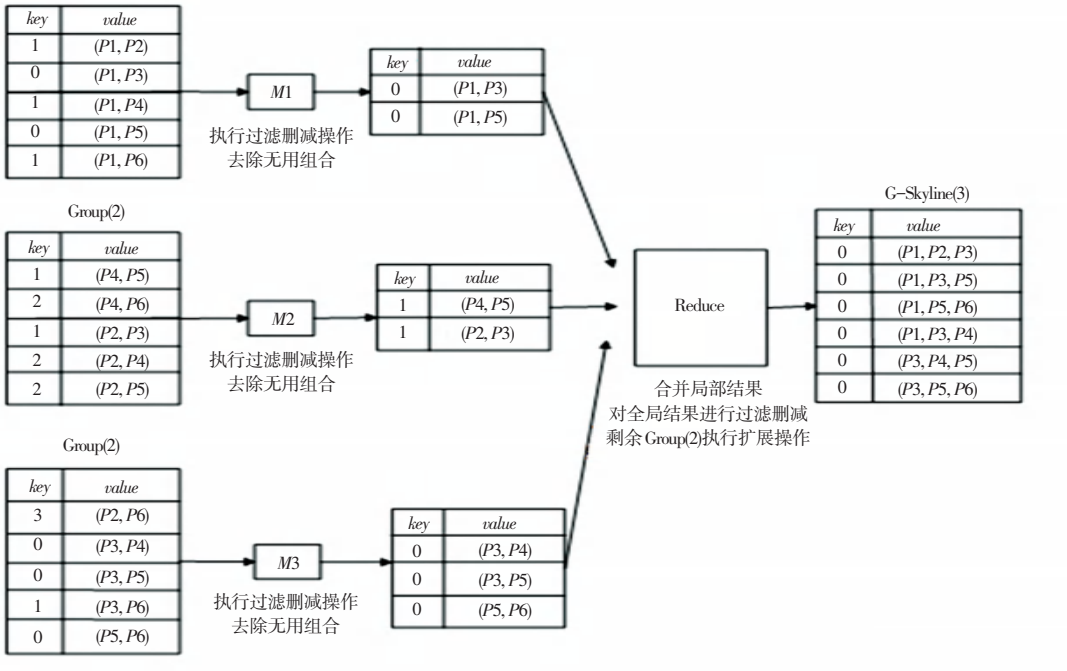


图 6　第二个 MapReduce 阶段

Fig. 6　Second MapReduce phase

表 2 为组合基数 n 与使用 MapReduce 次数之间的关系。由于在第 1 个 MapReduce 阶段中一定会得到 *key* 值为 0 的 *Group*(2)。为保证结果准确性，*Group*(2) 在每个阶段的扩展操作只能加入一个点，因此，要想获得所有基数为 n 的 G-Skyline 组，需要使用 $n-1$ 次 MapReduce。

表 2 组合基数 n 与 MapReduce 次数之间的关系

Table 2 Relationship between the number of points n in a combination and MapReduce times

	第一个 MapReduce	第二个 MapReduce	第三个 MapReduce	…	第 $n-1$ 个 MapReduce
$n=3$	Group(2)	G-Skyline(3)			
$n=4$		G-Skyline(3) G-Skyline(4)	G-Skyline(4)		
…		…	…	…	…
n		G-Skyline(n) G-Skyline($n-1$) G-Skyline($n-2$) …G-Skyline(3)	G-Skyline(n) G-Skyline($n-1$) …	…	G-Skyline(n)

在表 2 中可以看到，当 $n=3$ 时，共有 2 个 MapReduce 阶段。第 1 个 MapReduce 阶段用于生成所有的组合基数为 2 的候选组，用 *Group*(2) 表示。第 2 个 MapReduce 阶段生成 G-Skyline(3)。当 $n=4$ 时，共有 3 个 MapReduce 阶段。这里，第 1 个 MapReduce 阶段生成所有候选 *Group*(2)，第 2 个 MapReduce 阶段生成 G-Skyline(3) 和 G-Skyline(4)，G-Skyline(4) 可直接作为结果提前输出。接着在第 3 个 MapReduce 阶段进一步将 G-Skyline(3) 生成 G-Skyline(4)。

(1) Map 阶段。在 Map 阶段对每个分块中的候选 *Group*(m) 并行实施过滤删减操作，提出 2 个剪枝策略，如下所示。仍要指出的是，Group(m) 只要满足任何一个剪枝策略就会被过滤。

① 剪枝策略 1：对于一个 *Group*(m)，若对应的 *key* 值大于 $n-m$，则该组合一定不会生成一个基数为 n 的 G-Skyline 组，可以将 *Group*(m) 过滤掉。

证明 由定理 2 可知，给定一个数据点 P，如果点 P 在一个 G-Skyline 组中，则 P 的所有父亲也一定包含在该 G-Skyline 组中。对于($P1,P2,\cdots,Pm$)，这里的 $m<n$，若要生成一个基数为 n 的 G-Skyline 组，还需加入 $n-m$ 个点。设 $|\text{Parents}(P1\cup P2\cup\cdots\cup Pm)-(P1,P2,\cdots,Pm)|=key, key>n-m$，即未在组内的父亲个数大于需要加入组内点的个数，则($P1,P2,\cdots,Pm$) 无法生成一个基数为 n 的 G-Skyline 组，可以将该组合直接过滤掉。

② 剪枝策略 2：对于一个 *key* 值小于 $n-m$ 的 *Group*(m)，若被某个组合 G 支配，则由 *Group*(m) 扩展生成的一个基数为 n 的 G-Skyline 组，一定可以由 G 扩展得到，可以将 *Group*(m) 过滤掉。

证明 由组支配定义可知，对于基数相同的 2 个组 $G=(Pu1,Pu2,\cdots,Pum)$ 和 $G'=(Pv1,Pv2,\cdots,Pvm)$，若 $G<G'$ 则至少在某一点上存在严格支配的关系，即 $Pum<Pvm$，Pum 是 Pvm 的父亲。G' 中的 *key* 取值范围是 $0<key<n-m$，如果向 G' 中随意加入一个点生成基数更大的组合，要想为一个 G-Skyline 组可能还需要加入其他点，这对于快速生成一个 G-Skyline 组来说并不高效。因此以最快生成一个 G-Skyline 组为前提，对加入到组内的点进行优先级排序，G' 一定优先通过加入未在组内的点的父亲生成一个 G-Skyline 组，G 可以通过加入组内点的子集(加入的点满足其父亲均已在组内)或 Skyline 点生成一个 G-Skyline 组。通过比较可知，支配能力更强的组 G 不仅能够生成由 G' 扩展生成的相同组合，并且还能生成其他可能的 G-Skyline 组。因此，可将 G' 直接过滤掉。

以图 6 中的 $M1$ 为例，$M1$ 接收的分块中有 5 个 *Group*(2)，其中 ($P1,P2$) 和 ($P1,P4$) 被 ($P1,P3$) 支配，($P1,P6$) 被 ($P1,P5$) 支配，通过执行过滤删减操作，最终剩下 ($P1,P3$) 和 ($P1,P5$)。$M2$ 和 $M3$ 与 $M1$ 同时执行相同操作。可以明显看到，经过过滤删减操作后候选 *Group*(2) 的数量明显减少。

(2) Reduce 阶段。由于在 Map 阶段只能对每个分块中的 *Group*(m) 进行局部过滤删减，还有可能存在无用组合。因此，在 Reduce 阶段将剩余候选 *Group*(m) 合并成全局候选集后，再一次执行过滤删减操作，极大地减少需要执行扩展操作的候选 *Group*(m) 的数量。根据 *Group*(m) 中 *key* 值的不同，提出一种扩展操作。通过扩展操作会生成基数不同的候选组，对候选组进行组合身份判定，一部分候选组还需要继续执行扩展操作，一部分候选组有可能直接生成基数为 n 的 G-Skyline 组。

扩展操作就是通过向 $Group(m)$ 中加入未在组内的点的所有父亲或子集(保证其父亲都在组中)或 Skyline 点,生成所有基数为 n 的 G-Skyline 组、即 G-Skyline (n)。根据 $Group(m)$ 中 key 值的不同,对于扩展点的优先级也不同。

① 情况一:对于一个 key 值不为 0 的 $Group(m)$,优先考虑将未在该组内的点的所有父亲加入到组中生成基数更大的组合。当 $Group(m)$ 的 key 值为 $c(0 < c \leqslant n - m)$,可直接将 c 个点加入到 $Group(m)$ 中生成 $Group(c + m)$。若 $n = c + m$,则 $Group(c + m)$ 为一个基数为 n 的 G-Skyline 组,可直接作为结果输出;若 $n > c + m$,$Group(c + m)$ 还需要在下一阶段的 Reduce 继续执行扩展操作,直到得到一个基数为 n 的 G-Skyline 组。

证明　根据前述的定理 2 可知,一个 G-Skyline 组中点的所有父亲均在该组内。对于一个 $Group(m)$,$m < n$,该组合 key 值为 c,若 $c < n - m$,即未在组内的父亲个数小于需要加入组内点的个数,则 $Group(c + m)$ 一定不是一个基数为 n 的 G-Skyline 组,需要再次进行扩展操作。反之,若 $c = n - m$,即未在组内的父亲个数等于需要加入组内点的个数,$Group(c + m)$ 可直接作为一个基数为 n 的 G-Skyline 组输出。

② 情况二:对于一个 key 值为 0 的 $Group(m)$,一种方法是加入该组内点的孩子节点生成基数更大的组合,加入的点要满足其所有父亲也在该组内。另一种方法是直接加入一个 Skyline 点生成基数更大的组合。若 $m + 1 < n$,则 $Group(m + 1)$ 还需在下一阶段的 Reduce 继续执行扩展操作,直到得到一个基数为 n 的 G-Skyline 组。

证明　根据定理 2 可知,一个 G-Skyline 组中点的所有父亲均在该组内。通过加入一个孩子节点或一个 Skyline 点扩展生成 $Group(m + 1)$,若 $n > m + 1$,则 $Group(m + 1)$ 一定不是一个基数为 n 的 G-Skyline,需要再次执行扩展操作,直到得到一个基数为 n 的 G-Skyline 组。

表 3 为 $n = 3$ 时,在不同 key 值下候选 $Group(2)$ 执行的扩展操作。其中在($P4$,$P5$) 中,$P3$ 是 $P4$ 的父亲,根据扩展操作的情况一,可直接生成($P3$,$P4$,$P5$)。在($P1$,$P3$) 中,$P1$ 和 $P3$ 都已经是 Skyline 点,通过扩展操作的情况二,加入当前组内点的子集中的一个点(父亲均在当前组内)、比如 $P2$ 或 $P4$,或者加入一个 Skyline 点、比如 $P5$,生成($P1$,$P2$,$P3$)、($P1$,$P3$,$P4$) 和($P1$,$P3$,$P5$)。

表 3　*key* 值不同情况下的扩展操作

Table 3　Expansion operations in the case of different *key*

	$Group(2)$	未在组内的点的所有父亲	孩子节点	Skyline 点	G-Skyline 组
key 值不为 0	($P4$,$P5$)	$P3$			($P3$,$P4$,$P5$)
					($P1$,$P2$,$P3$)
key 值为 0	($P1$,$P3$)		$P2$,$P4$	$P5$	($P1$,$P3$,$P4$)
					($P1$,$P3$,$P5$)

MR-GSKY 算法的伪代码描述如下。

算法　基于 MapReduce 的组合 Skyline 查询算法(MR-GSKY)

输入　原始数据集 D,组合基数 n,DSG 图

输出　G - Skyline(n)

1. 对原始数据集 D 进行预处理,预处理后的数据集表示为 $D1$

2. Block($D1$) → Ci　// 将 $D1$ 中的数据分成 i 个分块

/ * 第一个 Map,并行更新所有分块中数据点的键值对 < key,$value$ > * /

3. for each part $p \in Ci$ do

4.　　p < key,$value$ > = p < parent number(p),$value$ >

5.　　输出 p < key,$value$ >　/ * 新的 key 值为 p 的所有父亲个数,新的 $value$ 为对应的数据点 * /

6. end

/ * 第一个 Reduce,生成全部候选 $Group(2)$ 并更新键值对 * /

7. for each $p \in Ci$ do

8.　　用循环分配策略生成所有候选 $Group(2)$

9.　　更新所有候选 $Group(2)$ 的键值对 / * 新的 key 值为未在组内的点的所有父亲个数,新的 $value$ 为对应的 $Group(2)$ * /

10.　　输出所有候选 $Group(2)$

11. end

12. $Block(Group(m))$ → Cj　//将上一个 MapReduce 阶段得到的候选 $Group(m)$ 分为 j 块

/* 第 m 个 Map 阶段,并行对每一分块中的 $Group(m)$ 执行过滤删减操作 */

13. for each part $Group(m) \in Ci$ do

14. 按照剪枝策略进行过滤删减操作

15. 输出每一分块中剩余候选 $Group(m)$

16. end

/* 第 m 个 Reduce 阶段,对全局候选 $Group(m)$ 执行过滤删减操作,最终执行的是扩展操作 */

17. integrate each part $Group(m) \rightarrow D2$/* 将每一分块中剩余候选 $Group(m)$ 合并为全局候选集,用 $D2$ 表示 */

18. for each $Group(m) \in D2$ do

19. 对全局候选集 $D2$ 进行过滤删减操作

20. 输出剩余全局候选 $Group(m)$ 结果

21. end

22. for each $Group(m)$ do

23. if $key! = 0$ then

24. 向 $Group(m)$ 中加入未在组内的点的所有父亲

25. 输出 $Group(key + m)$ 并更新键值对

26. else $key == 0$

27. 向 $Group(m)$ 中加入组内点的孩子节点(该点的所有父亲均在组内)或 Skyline 点

28. 输出 $Group(m + 1)$

29. end

30. end

/* 对第 m 个 MapReduce 阶段得到的全部候选组进行组合身份判定 */

31. for each $Group(key + m)$ do

32. if $key + m == n$ then

33. $Group(key + m)$ 直接作为一个基数为 n 的 G-Skyline 组输出

34. else $key + m < n$

35. 继续在下一 MapReduce 的 Reduce 执行扩展操作

36. end

37. end

38. for each $Group(m + 1)$ do

39. if $m + 1 == n$ then

40. $Group(m + 1)$ 直接作为一个基数为 n 的 G-Skyline 组输出

41. else $m + 1 < n$

42. 继续在下一 MapReduce 阶段执行扩展操作

43. end

44. end

45. 输出所有基数为 n 的 G-Skyline 组,用 G-Skyline(n) 表示

其中,在第 1~2 行将预处理后的数据集 D2 分块。第 3~6 行表示第一个 MapReduce 的 Map 阶段,接收每个分块并行更新每个分块中数据点的键值对。第 7~11 行表示第一个 MapReduce 的 Reduce 阶段,用于生成全部候选 $Group(2)$ 并对每个 $Group(2)$ 的键值对进行更新。从第二个 MapReduce 阶段开始,每一阶段都用相同的过滤删减策略和扩展策略计算。第 12 行对上一 MapReduce 阶段得到的候选 $Group(m)$ 分块。第 13 ~ 16 行表示第 m 个 MapReduce 的 Map 阶段,在这一阶段对每一分块执行过滤删减操作,删除掉无用候选 $Group(m)$,减少需要进行扩展操作的候选组数量。第 17~29 行表示第 m 个 MapReduce 阶段的 Reduce 阶段,合并每一分块中剩余 $Group(m)$ 并再次执行过滤删减操作,再对剩余的全局候选 $Group(m)$ 执行扩展操作。为得到全部基数为 n 的 G-Skyline 组,在第 30 ~ 44 行设置对第 m 个 MapReduce 阶段得到的所有候选组进行组合身份判定,若当前候选组不满足条件,需在下一 MapReduce 的 Reduce 阶段继续执行扩展操作。第 45 行表示输出 G-Skyline (n)。

4 实验

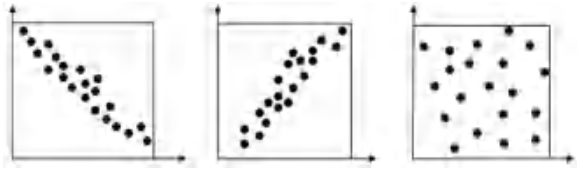
4.1 实验设置

本实验在 6 台 PC 机的集群上执行。每台机器配备的都是 Inter Core2 E8400 3 GHz 处理器和 4 GB RAM。所有算法都用 Java 编写,程序运行环境是 Hadoop1.2.1 和 JDK 1.6。为了验证算法的有效性和高效性,在实验中分别采用合成数据集和真实传感器数据集对算法进行评估。

其中,合成数据集采用了与文献[35]相似的数据集,合成数据集的数据分布包括相关数据集(COR)、反相关数据集(ANTI-COR)和独立数据集(IND)。相关数据集的数据分布如图 7(a)所示,一般在一个维度上占优势的数据在另一维度上也占优势。对于二维数据,2 个坐标所表现出的趋势一般是相同的。反相关数据集的数据分布如图 7(b)所示,在一个维度上占优势的数据在另一个维度上表现是较差的。独立数据集的数据分布如图 7(c)所示,在各个维度

上的变化趋势都是独立的，不存在任何关系。

以上数据集也都是 Skyline 查询中常用的数据集。真实传感器数据集来自一个森林环境监测项目，其中包括温度、湿度、日降雨量、日蒸发量和日温度范围。实验相关参数设置见表 4。



(a) 数据分布 1　(b) 数据分布 2　(c) 数据分布 3

图 7　合成数据集示例

Fig. 7　Example of a synthetic dataset

表 4　实验相关参数设置表

Table 4　Experiment-related parameter setting table

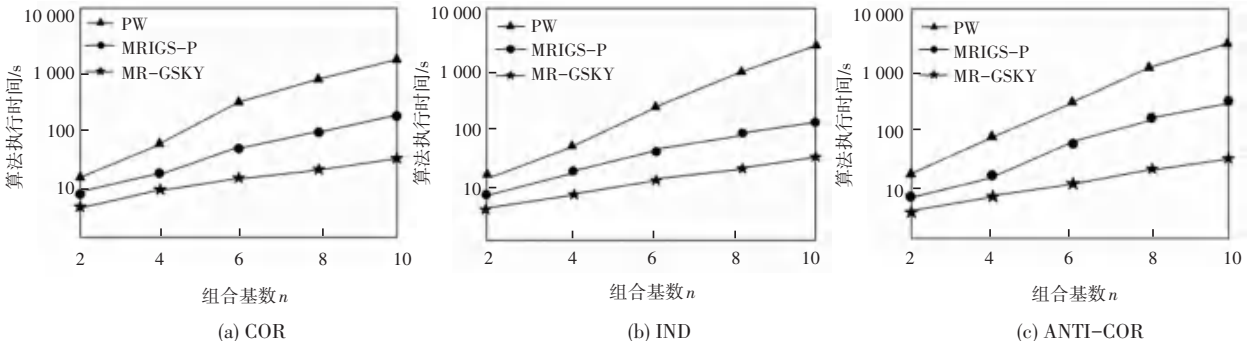
实验参数	取值范围
组合基数 n	2,4,6,8,10
数据集大小 $\|D\|$	100,200,300,400,500
维度大小 d	2,3,4,5,6

为了验证该算法的高效性，选择了文献[27]中的 PW(Point-Wise)算法和文献[35]中的 MRIGS-P 算法参与比较。文献[35]中分别有 MRGS、MRIGS 和 MRIGA-P 算法。MRGS 是一个基于 MapReduce 的组合 Skyline 基本算法。针对 MRGS 算法节点负载不平衡这一缺陷，进一步提出了 MRIGS 算法。MRIGS-P 算法在 MRIGS 算法基础上进行优化，提出级联剪枝删除掉一些点。通过实验可以看到，MRIGS-P 算法在各个方面都更具优势。所以最终选择 MRIGS-P 算法与 MR-GSKY 算法进行比较，以凸显出 MR-GSKY 在执行效率上的优势。

4.2　合成数据集

在本小节中，通过合成数据集来测试 PW、MRIGS-P 和 MR-GSKY 算法的性能，通过比较验证 MR-GSKY 的有效性和高效性。

(1)组合基数 n 的可扩展性。在这个实验中，研究了组合基数 n 的大小对算法执行时间的影响。图 8 表示在合成数据集中(COR、IND 和 ANTI-COR)组合基数 n 与算法执行时间的关系。其中，数据集大小 $|D|$ 固定为 300，维度 d 固定为 2，组合基数 n 从 2 到 10。对于相关数据集和独立数据集，该实验的执行时间是对数标度。



(a) COR　(b) IND　(c) ANTI-COR

图 8　组合基数 n 与算法执行时间的关系

Fig. 8　Relationship between the number of points n in a combination and algorithm execution time

在 3 种数据分布中，可以发现 MR-GSKY 的表现均优于 PW 和 MRIGS。当组合基数较小时，由于生成的候选组也相对较少，MR-GSKY 算法的优势并不明显。随着组合基数越来越大，生成的候选组数量也在增加，MR-GSKY 算法优势明显。这是因为 MR-GSKY 算法能够在查询过程中提前删除掉无法生成 G-Skyline(n) 的候选组并提取支配能力更强的候选组，同时部分候选组能够通过扩展操作提前生成最终 G-Skyline 结果。因此，在每一种情况下 MR-GSKY 都要优于 PW 和 MRIGS-P。

(2)数据集大小 $|D|$ 的可伸缩性。仿真实验中，研究了数据集大小 $|D|$ 对算法执行时间的影响。图 9 表示在合成数据集中(COR、IND 和 ANTI-COR)数据集大小 $|D|$ 与算法执行时间的关系。其中，组合基数 n 固定为 4，维度 d 固定为 2，数据集大小 $|D|$ 从 100 到 500。

在这 3 种数据分布中可以看到，在所有情况下，MR-GSKY 都要优于其他 2 个算法。当数据集的数量 $|D|$ 较小时，MR-GSKY 中预处理和过滤删减操作的执行优势不明显，候选集个数也不会显著减少。当数据集越来越大时，通过减少大量的无用候选组，提高查询效率，从而有效减少算法执行时间。MRIGS-P 仅能在数据集较小时发挥其优势，但与 MR-GSKY 相比，算法执行时间较高。

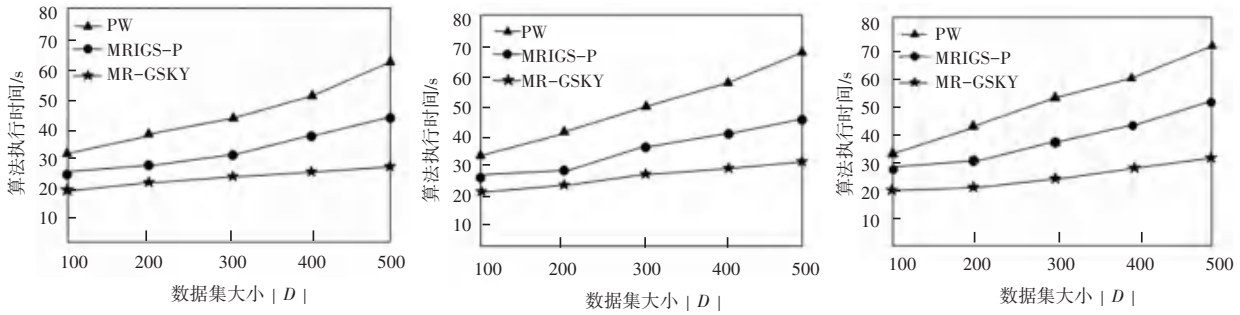


图 9 数据集大小|*D*|与算法执行时间的关系

Fig. 9 Relationship between dataset size |*D*| and algorithm execution time

(3)维度大小 d 的可伸缩性。在这个实验中,研究了维度大小 d 对算法执行时间的影响。图 10 表示在合成数据集中(COR、IND 和 ANTI-COR)维度大小 d 与算法执行时间的关系。组合基数 n 固定为 4,数据集大小 | D | 固定为 300,维度大小 d 从 2 到 5。

随着维度 d 的增加,每个算法的执行时间都明显增多。在相同维度下,MR-GSKY 执行时间最短。

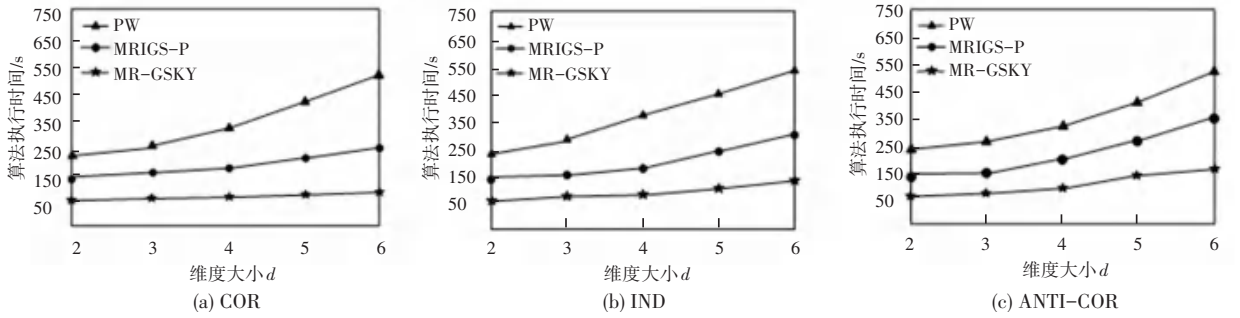


图 10 维度大小 *d* 与算法执行时间的关系

Fig. 10 Relationship between dimension size *d* and algorithm execution time

4.3 真实传感器数据集

在本小节中,通过真实传感器数据集来测试 PW、MRIGS-P 和 MR-GSKY 算法的性能,展示算法处理真实传感器数据集的能力和效果,通过比较验证 MR-GSKY 的有效性。

图 11 显示了在相同维度 (d = 2),相同数据集大小(| D | = 300) 及不同组合基数 n 下的算法执行时间。

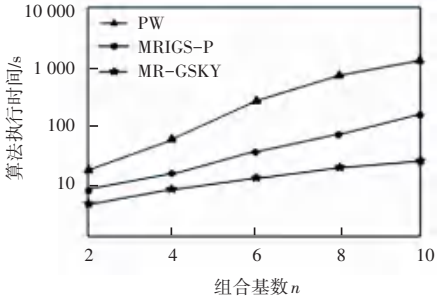


图 11 组合基数 *n* 与算法执行时间关系

Fig. 11 The relationship between the number of *n* in a combination and algorithm execution time

研究发现,算法执行时间随组合基数的增加显著加大。这是因为随着 n 的增加,需要进行检查和扩展的点会越来越多。值得注意的是,MR-GSKY 的执行时间要比 PW 和 MRIGS-P 少得多。其原因是在 Map 和 Reduce 阶段执行的过滤删减操作,极大地减少了数据点和候选组数量,这样将花费更少的时间对剩余候选组进行扩展操作,同时在每一 MapReduce 阶段均有可能提前生成部分 G-Skyline 组。

图 12 显示了在相同维度 (d = 2),相同组合基数 (n = 3) 及不同数据集大小 | D | 下的算法执行时间。

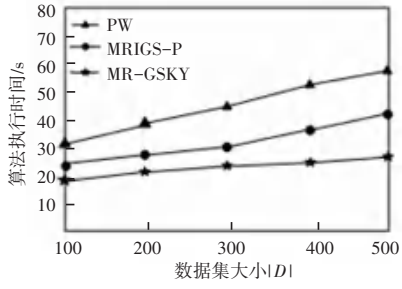


图 12 数据集大小|*D*|与算法执行时间关系

Fig. 12 Relationship between dataset size | *D* | and algorithm execution time

分析可知，随着输入数据集的增加，各算法的执行时间均显著增加。同时也能观察到，MR－GSKY 的优势相对其他算法表现得也越来越明显。因为该算法中所提出的预处理和过滤删减策略能够最大限度地将无用数据点及无用候选组删除掉，通过执行这些操作，在查询过程中不需要用到全部数据集就能得到 G － Skyline(n)，缩短了算法执行时间。

图 13 显示了在相同组合基数（n = 3），相同数据集大小（| D | = 300）及不同维度 d 下的算法执行时间。

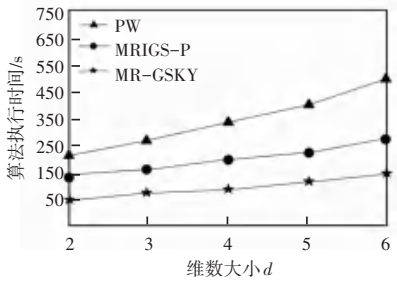


图 13　维度大小 d 与算法执行时间关系

Fig. 13　Relationship between dimension d and algorithm execution time

研究发现，随着维度的增加，每个算法执行时间也都明显增大。这是因为随着 d 的增大，点的支配能力越弱，更多的点位于 DSG 图前 n 层，在查询过程中计算复杂度增大，算法执行时间也有所增长。

5　结束语

针对无线传感网络所处环境的不确定性和传输带宽有限等对组合 Skyline 查询效率的影响，本文基于并行编程框架 MapReduce，提出适用于无线传感网络的并行计算组合 Skyline 查询算法，即 MR－GSKY 算法。该算法首先对原始数据集进行预处理操作，去除掉不可能存在于组合 Skyline 中的点。接着对 MapReduce 中的 2 个主要阶段进行改进，提出在 Map 阶段的过滤删减策略和在 Reduce 阶段的扩展策略。实验结果表明，MR－GSKY 有着优于 PW 和 MRIGS－P 的所有性能指标，证明了 MR－GSKY 算法的有效性和高效性。在未来会考虑对无线传感网络中离散数据的组合 Skyline 查询展开研究。

参考文献

[1] BORZSONYI S, STOCKER K, KOSSMANN D. The Skyline operator[C]//Proceedings of 17th International Conference on Data Engineering. Heidelberg, Germany: IEEE, 2001: 421－430.

[2] LIN Xuemin, YUAN Yidong, ZHANG Qing, et al. Selecting Stars: The k most representative Skyline operator[C]//IEEE 23rd International Conference on Data Engineering. Istanbul, Turkey: IEEE, 2007: 86－95.

[3] TIAN Xia, ZHANG Donghui, TAO Yufei. On Skylining with flexible dominance relation[C]//2008 IEEE 24th International Conference on Data Engineering. Cancún, Mexico: IEEE, 2008: 1397－1399.

[4] GAO Yunjun, LIU Qing, CHEN Lu, et al. Efficient algorithms for finding the most desirable skyline objects[J]. Knowledge－Based Systems, 2015, 89: 250－264.

[5] MAN L, MAMOULIS N. Multi－dimensional top－k dominating queries[J]. The VLDB Journal, 2009, 18(3): 695－718.

[6] LIU Xin, YU Jing, LIU Guohua. Algorithm for skyline queries based on window query[J]. Journal of Yanshan University, 2005, 29(5): 398－402.

[7] WU Ping, ZHANG Caijie, FENG Ying, et al. Parallelizing skyline queries for scalable distribution[C]//International Conference on Extending Database Technology. Berlin/Heidelberg: Springer, 2006: 112－130.

[8] VLACHOU A, DOULKERIDIS C, KOTIDIS Y, et al. SKYPEER: Efficient subspace Skyline computation over distributed data[C]//2007 IEEE 23rd International Conference on Data Engineering (ICDE 2007). Istanbul, Turkey: IEEE, 2007: 416－425.

[9] WANG Shiyuan, OOI B C, TUNG A K H, et al. Efficient Skyline query processing on peer－to－peer networks[C]// IEEE 23rd International Conference on Data Engineering(ICDE 2007). Istanbul, Turkey: IEEE, 2007: 1126－1135.

[10] ZHANG Boliang, ZHOU Shuigeng, GUAN Jihong. Adapting skyline computation to the mapreduce framework: Algorithms and experiments[C]//International Conference on Database Systems for Advanced Applications. Berlin/Heidelberg: Springer, 2011: 403－414.

[11] CHEN Liang, HWANG K, WU Jian. MapReduce Skyline query processing with a new angular partitioning approach[C]// 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. Shanghai, China: IEEE, 2012: 2262－2270.

[12] 王淑艳，杨鑫，李克秋. MapReduce 框架下基于超平面投影划分的 Skyline 计算[J]. 计算机研究与发展，2014，51(12)：2702－2710.

[13] 齐玉东，何诚，司维超. MapReduce 框架下的 Skyline 云资源选择算法[J]. 计算机科学，2018，45(S1)：411－414.

[14] 王澍. MapReduce 环境下基于支配层次树的 k－支配 skyline 查询方法研究[J]. 沈阳：辽宁大学，2019.

[15] KOH J L, CHEN C C, CHAN C Y, et al. MapReduce skyline query processing with partitioning and distributed dominance tests[J]. Information Sciences: An International Journal, 2017, 375: 114－137.

[16] YIN Bo, ZHOU Siwang, ZHANG Shiwen, et al. On Efficient processing of continuous reverse skyline queries in wireless sensor networks[J]. KSII Transactions on Internet & Information Systems, 2017, 11(4): 1931－1953.

[17] WANG Yan, SONG Baoyan, WANG Junju, et al. Geometry－based distributed spatial skyline queries in wireless sensor networks[J]. Sensors, 2016, 16(4): 454.

[18] AHMED K, NAFI N S, GREGORY M A. Enhanced distributed dynamic skyline query for wireless sensor networks[J]. Journal of Sensor and Actuator Networks, 2016, 5(1): 2.

[19] WANG Zhiqiong, XIN Junchang, WANG Pei. Alternative Tuples based probabilistic skyline query processing in wireless sensor networks[J]. Mathematical Problems in Engineering, 2015, 2015:1-10.
[20] 信俊昌,王国仁. 无线传感器网络中 Skyline 节点连续查询算法[J]. 计算机学报, 2012, 35(11): 2415-2430.
[21] 尹波. 分布式环境下 skyline 查询处理技术研究[D]. 长沙:湖南大学,2019.
[22] 潘立强,李建中,骆吉洲. 无线传感器网络中一种近似 Skyline 查询处理算法[J]. 软件学报, 2010,21 (5): 1020-1030 .
[23] IM H, PARK S. Group skyline computation[J]. Information Sciences, 2012,188:151-169.
[24] LI C, RAJASEKARAN S, NAN Z, et al. On Skyline groups [J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 26(4): 2119-2123.
[25] ZHANG Nan, LI Chengkai, HASSAN N, et al. On Skyline groups [J]. IEEE Transactions on Knowledge & Data Engineering, 2014, 26(4): 942-956.
[26] CHUNG Y, SU I, LEE C. Efficient computation of combinatorial skyline queries[J]. Information Systems, 2013, 38(3): 369-387.
[27] LIU Jinfei, XIONG Li, PEI Jian, et al. Finding Pareto optimal groups: group-based skyline[J]. Proceedings of the VLDB Endowment, 2015,8(13): 2086-2097.
[28] YU Wenhui, QIN Zheng, LIU Jinfei, et al. Fast algorithms for Pareto optimal group-based Skyline[C]// ACM International Conference on Information and Knowledge Management. Singapore:ACM, 2017:417-426.
[29] WANG Changping, WANG Chaokun, GUO Gaoyang, et al. Efficient computation of G-Skyline groups[J]. IEEE Transactions on Knowledge and Data Engineering, 2018,30(4): 674-688.
[30] XU Zhou, LI Kenli, YANG Zhibang, et al. Progressive approaches for Pareto optimal groups computation[J]. IEEE Transactions on Knowledge and Data Engineering, 2019,31(3): 521-534.
[31] YANG Yuntian, LU Wenbo, TANG Cong. A fast top-k group skyline query method based on skyline layer [C]//2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI). Sanya, China :IEEE,2020:146-151.
[32] LIU Jinfei, XIONG Li, PEI Jian, et al. Group-based Skyline for Pareto optimal groups[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(7) :2914-2929.
[33] HAN Xixian, WANG Jinbao, LI Jianzhong, et al. Efficient computation of G-Skyline groups on massive data [J]. Information Sciences, 2022, 587: 300-322.
[34] 李光辉,李艳红,杨洋,等. 在正交查询范围内解决 G-Skyline 查询中的[J]. 中南民族大学学报(自然科学版), 2023, 42(5):678-688.
[35] WANG Changping, WANG Chaokun, GUO Gaoyang, et al. Efficient computation of G-Skyline groups [J]. IEEE Transactions on Knowledge and Data Engineering, 2018,30(4): 674-688.
[36] 石碧瑶. Hadoop MapReduce 海量数据处理方法分析与研究[J]. 西安交通工程学院学术研究, 2022(1): 56-59,63.
[37] LIU Jia, CHEN Wei, CHEN Ziyang, et al. Optimized query algorithms for Top-K group Skyline [J]. Wireless Communications and Mobile Computing,2022,2022:1-11. DOI: 10.1155/2022/3404906.