无线传感器网络中的Skyline查询处理技术

王海翔[1] 郑吉平[1,2] 宋保利[1]
(南京航空航天大学计算机科学与技术学院 南京 210016)[1]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 Skyline查询作为多目标决策的重要手段之一,在无线传感器网络应用中发挥着越来越重要的作用。对无线传感器网络中的Skyline查询处理技术进行了论述。首先讨论集中数据库中的Skyline查询算法。其次,讨论无线传感器网络中Skyline查询的典型应用。进而,根据无线传感器网络能量、存储和处理能力有限等特点,全面论述了无线传感器网络中的Skyline查询方法,并指出了今后的研究方向。

关键词 无线传感器网络,Skyline查询,过滤,能量高效,数据划分
中图法分类号 TP392 **文献标识码** A

Skyline Query Processing in Wireless Sensor Networks

WANG Hai-xiang[1] ZHENG Ji-ping[1,2] SONG Bao-li[1]
(Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[1]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)[2]

Abstract As one of the important means of multi-criteria decision making problems, the skyline query processing technology plays an increasingly important role for applications of the wireless sensor networks. This paper summarized the studies of skyline query processing techniques in wireless sensor networks. Firstly, it described algorithms for answering skyline queries in centralized databases. Secondly, it listed typical applications of the skyline queries in the wireless sensor networks. Thirdly, it thoroughly discussed the skyline query methods of the wireless sensor networks according to limited energy and storage as well as processing abilities. Finally, this paper proposed several directions for further research.

Keywords Wireless sensor networks, Skyline queries, Filtering, Energy-efficient, Data partitioning

1 引言

感知技术、嵌入式技术和通信技术的进步,推动了低功耗、多功能传感器的发展,使其集成了数据采集、数据传输和数据处理等多种功能。传感器节点感知、采集和处理网络中监测的信息,如光照、温度和湿度等,并通过无线通信经路由将信息传送至远程用户。传感器节点因为体积小、成本低等诸多优点得到了广泛的应用,其在国防军事、环境监测、交通管理、医疗卫生等诸多领域具有十分广阔的应用前景。然而,传感器节点在实际应用中还存在一些限制和约束:节点都是由电池供电,它的能量极其有限,这就导致节点往往因电源能量的问题而失效或废弃。实际应用中的传感器节点一般部署在环境恶劣的地方,因此通过人为的方式更换电池是不现实的,电源能量约束成为阻碍传感器网络应用的严重问题,在应用中需要高效运用它们的能量资源以最大限度地提高系统的生命周期。因此,如何在实际应用中尽可能地节省能量,是传感器网络研究面临的巨大挑战[1]。

加州大学伯克利分校的TinyDB[2]和康奈尔大学的Cougar[3]是两种典型的感知数据查询系统,它们实现了MAX、MIN等聚合操作,对于无线传感器网络中的其它查询,如Join运算[4,5]、kNN查询[6]、基于过滤的、近似及精确的top-k查询[7-9]、时空连续查询[10]以及Skyline查询[11-13]等在近年来得到了广泛研究。Skyline查询从多维元组集合中返回具有优势的元组,是多目标决策的重要手段之一。Borzsonyi等人[14]最早提出Skyline查询,为描述Skyline引入了一个典型的例子:假设去Nassau海滩旅游,向数据库查找一个既便宜又靠近海滩的旅馆。一般情况下越靠近海滩的旅馆价格越高,这时候数据库系统不能返回用户一个最好的结果,但它能反馈出一些用户感兴趣的旅馆,这些旅馆在价格和距离两个方面都不比其它旅馆差,这就引出了Skyline。具体来讲,Skyline查询就是从给定的N维空间的元组集合T中选择那些不被任何元组支配的元组。假设有两个元祖$t_i(t_{i1},t_{i2},\cdots,t_{iN})$和$t_j(t_{j1},t_{j2},\cdots,t_{jN})$,若$t_i$的所有维度上的属性值都不比$t_j$差,并且至少在一维上的属性值优于$t_j$,那么$t_i$支配$t_j$,即$t_i \succ t_j$。"优于"可能是指小于也可能是指大于,这要根据上下文判断,本文假设取小为优。图1中x轴代表旅馆价格,y轴

到稿日期:2012-11-15 返修日期:2013-03-20 本文受教育部高等学校博士学科点专项科研基金项目(20103218110017),江苏高校优势学科建设工程资助项目,南京航空航天大学青年科技创新基金(NS2010116,NN2012102),南京航空航天大学研究生开放基金(KFJJ120222)资助。
王海翔(1989-),女,硕士,主要研究方向为信息物理融合,E-mail:wanghaixiang@nuaa.edu.cn;郑吉平(1979-),男,博士,副教授,CCF高级会员,主要研究方向为传感器数据管理、粒子滤波、蒙特卡罗方法等;宋保利(1987-),男,硕士,主要研究方向为不确定数据管理。

代表旅馆与海滩的距离，黑点代表各个对应的旅馆，那么根据定义，折线上的点即为 Skyline 结果集。

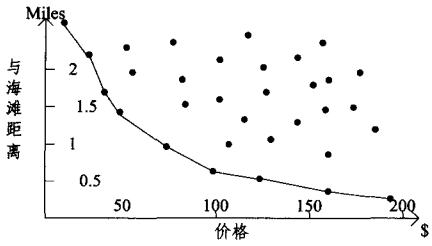


图 1　Nassau 海滩旅馆的 Skyline

对无线传感器网络中的 Skyline 查询处理算法的研究是传感器网络数据处理的重要内容之一。由于传感器节点的能量、计算、存储和通信能力都有限，Borzsonyi 等人[14]提出了 Skyline 查询以及后续的集中数据库环境中的 Skyline 查询处理技术，但其并不适用于无线传感器网络，很多研究人员针对传感器网络的特性提出了一系列 Skyline 查询处理算法。本文对国内外无线传感器网络中的 Skyline 查询处理技术的研究进行了归纳总结。第 2 节介绍了集中式数据库中的 Skyline 查询处理技术，并简要总结了其它应用环境中的 Skyline 查询方法；第 3 节介绍了 Skyline 查询在传感器网络中的应用场景；第 4 节通过几个算法介绍了无线传感器网络环境中基于过滤思想的 Skyline 查询处理技术；最后作出总结，提出了 Skyline 查询在无线传感器网络中的几点后续研究方向。

2　集中式环境中的 Skyline 查询处理技术

Borzsonyi 等人[14]在数据库查询领域引入 Skyline 操作后，研究人员对于集中数据库中的 Skyline 查询处理提出了许多改进算法，其在块嵌套循环（Block-nested-loops，BNL）算法、分而治之（Divide and Conquer，D&C）算法以及 B 树等索引技术的基础上提出了改进思想。从 Skyline 查询处理过程是否借助于索引的角度，将其分为不带索引的 Skyline 查询算法和带索引的 Skyline 查询算法两大类。

2.1　不带索引的算法

不带索引的算法中 BNL 算法和 D&C 算法最具代表性。BNL 算法通过反复读取元组计算 Skyline，算法的思想是在内存中保持一个可比较的元组窗口，当从输入中读取元组 p 时，p 和窗口中的所有元组进行比较。基于这个比较，元组 p 面临以下 3 种情况：a）被去除；b）被写入到窗口；c）被写入到临时文件。所有元组比较完后，输出窗口中的元组作为 Skyline 结果的一部分，重复这一过程计算 Skyline 直至临时文件内无待处理元组为止。D&C 算法将数据集划分成几个部分，使得每部分都可以放入内存，然后分别计算每一部分的 Skyline 点，最后通过合并每部分的 Skyline 去除其中被支配的数据点来得到最终的 Skyline 数据集。

Tan 等人[15]提出了 Bitmap 的方法，使得数据不用分块而采用位图结构就能快速识别出一个点是否为用户感兴趣的点。每一条记录映射为一个 n 位的向量，n 是所有维度的属性个数。算法无须遍历整个数据集就能判断出一个点是否在 Skyline 中，从而快速返回查询结果。然而 Bitmap 方法采用的映射方式会导致位串长度随各维上不同取值的增大而急剧增加，从而占用大量存储空间。Chomicki 等人[16]提出了排序过滤 Skyline 算法（Sort Filter Skyline，SFS），其是在 BNL 算法的基础上提出来的。SFS 算法的主要思想是：对输入文件中的元组先按照 Skyline 规则进行排序，即预排序（Presorting），然后再按照 BNL 算法进行处理。SFS 算法中输入的元组是排序过的，因此找到一个 Skyline 点后，不能被其后面的元组支配，这样就不用跟窗口中的元组进行替换，可以直接输出窗口中的元组作为 Skyline 集的一部分，从而减少了开销。显然，SFS 算法在关系操作中性能较好。Godfrey 等人[17]在 SFS 算法的基础上提出了 Skyline 的线性排序算法（Linear Elimination Sort for Skyline，LESS），它在预处理的过程中提前去除一些不可能属于 Skyline 的点，进一步提高了 SFS 算法的性能。LESS 通过一个 Skyline 过滤器（Skyline Filter，SF）过滤掉一些记录，预先加入一个消除过滤窗口对外部排序的数据，提前删除一些不可能是 Skyline 的点，从而提高了算法性能。

2.2　带索引的算法

Borzsonyi 等人[14]提出了使用了 B 树和 R 树索引的方法来计算 Skyline。B 树索引算法主要针对二维空间的 Skyline 查询问题，用 B 树对所有元组建立一个有序索引，然后扫描整个索引，得到按序排列的元组。B 树索引算法仅需要比较当前元组和已处理元组中最近的 Skyline 点，就可以判断该元组是否为 Skyline 中的点。B 树索引算法仅在数据集小并且能很快找到第 1 个匹配点的情况下才具有较高的效率。而 R 树索引算法针对任意空间上的 Skyline 查询问题，用 R 树对所有元组建立一个有序索引，然后深度优先遍历 R 树，每当遇到一个新的元组时就进行剪枝。可以看出，当 R 树中记录了 Skyline 查询语句中所有维度上的属性信息时，R 树索引算法比较适用。

Tan 等人[15]提出了 Index 索引算法，它采用一个转换机制将高维数据转换为低维数据，然后使用 B^+ 树对转换后的数据建立索引。Index 算法的基本思想是：将各个维度上的属性值按升序排列，扫描各个元组得到第一个 Skyline 点。这样可以很容易删除每个维度值都排在这个点后的元组，从而节省 I/O 开销。Kosmann 等人[18]提出了最近邻算法（Nearest Neighbor，NN），即第一个用户友好的 Skyline 计算方法。NN 算法的基本思想是：利用搜索到的最近邻居来递归地划分数据空间，它根据用户自定义的距离公式，用 R 树对所有数据建立一个索引。每一次找到与原点距离最小的点后，用这个点将数据空间划分成若干子部分。显然，这个点是一个 Skyline 点，可以被立即输出给用户。NN 将在每一子部分上递归调用这一过程直到子部分只含有一个点为止。Papadias 等人[19,20]在 NN 的基础上提出了分支界限（Branch and Bound Skyline，BBS）的思想，算法基于最近邻搜索并采用 R 树对数据建立索引。R 树中的每个上层节点对应下层的一个最小边界矩形（Minimum Boundary Rectangle，MBR），叶子节点则对应一个数据点，计算每个 MBR 到原点的最小距离。BBS 算法首先将 R 树根节点的所有子 MBR 按其到原点的最小距离插入一个最小堆中，每一次从最小堆中取出根节点，并扩展出它的所有子 MBR，一一插入堆中。当最小堆的根节点不包含子 MBR 时，计算出此节点里所含的数据，求出新的 Skyline 点。Lee 等人[21,22]提出了 Z-SKY 框架，其根据 Z 顺序曲线（Z-order Curve）特性提出了一个新的 Skyline 查询框架，该框架由 *ZBtree* 索引结构构成，用来组织源数据库和 Skyline 候

选集。另外 Z-SKY 框架中还提出了一系列算法，*ZSearch* 处理 Skyline 查询，*ZInsert*、*ZDelete* 和 *ZUpdate* 递增地维护 Skyline 查询结果，*ZRand* 返回最优的 Skyline 点，*k-ZSearch* 评估 k 支配的 Skyline 查询等等。此外，对元组数的有效评估和查询代价分析可以提高 Skyline 查询处理效率。Chaudhuri 等人[23]提出了鲁棒技术，其用日志采样(Log Sampling，LS)来评估元组数和 Skyline 查询处理代价，并在 Microsoft SQL Server 中实现了 Skyline 查询。LS 方法仅适用于非独立数据的独立维度上的理论结果，这可能导致大量评估错误。为此，Zhang 等人[24]提出了基于内核(Kernel-Based，KB)的方法，使用非参数方法(Nonparametric Methods)来近似 Skyline 查询中的元组数，在各种真实数据库中，KB 方法都具有较高可靠性。另外，还将 LS 和 KB 方法扩展到 k 支配的 Skyline 查询中，这在高维数据的 Skyline 查询中比传统 Skyline 查询更适用。

除了集中式环境，Skyline 查询在分布式、P2P、Web 以及不确定等环境下也得到了广泛应用。Katja 等人[25]对分布式环境中的 Skyline 查询处理算法进行了详细的论述。Wu 等人[26]研究了并行 Skyline 查询执行的问题，提出使用分布的 Skyline 查询处理算法(Distributed Skyline，DSL)逐步发现 Skyline 点。DSL 算法有两种机制，一种是递归的区域划分机制，另一种是动态的区域编码机制。Chen 等人[27]在 P2P 网络中提出 Skyline 计算问题，采用 iMinMax(θ)转换方法将高维的点映射成一维的值。所有转换的点分布到一个有结构的 P2P 网络 BATON 中，将所有的端组织为一个平衡二叉树，这种方法称为 iSky。由于映射的属性不适用于动态空间，因此 iSky 不能支持动态 Skyline 查询。Wang 等人[28]在 P2P 系统中提出了处理 Skyline 查询的 Skyframe 框架，它能够快速响应查询，并且具有较低的通讯代价。Skyframe 包含两种查询方法，一个关注优化网络通讯，另一个关注减少查询响应时间。Skyframe 通过加载节点间分离、合并的数据空间以及动态加载迁移来实现查询负载均衡。Tao 等人[29]主要研究了数据流环境中的 Skyline 查询处理方法，提出了滑动窗口 Skyline 的维护策略，它持续监控新到来的数据，使用最新的滑动窗口内的元组来计算 Skyline 并逐步维护结果。对流数据环境中的 Skyline 计算提出两种方法，一种是“懒”策略(Lazy Method)：在某个 Skyline 点即将失效时才进行大多数计算工作；另一种是“渴望”策略(Eager Method)：利用预计算来最小化主存开销。Balke 等人[30]根据 Web 资源分布独立的特点，首次提出了一种有效的基于 Web 的分布式 Skyline 算法(Web based distributed Skyline algorithm)。算法的基本流程是：首先，按照元组在各个维度上的属性值降序分别建立有序队列，依次从队列中检测元组 T_{new}，建立一个中间队列 Q 并确定一个所有维上属性值已知的元组 T_{known}；然后，对于有序队列中 T_{known} 之后的元组，比较其各维属性值与 T_{known} 的大小，若都较小则终止检测；最后去除中间队列被支配的元组，得到 Skyline 结果。文献[31，32]对不确定数据的 Skyline 查询处理进行了研究。Pei 等人[31]提出了不确定数据的概率 Skyline 查询，即 p-skyline。一个不确定的对象有存在于 Skyline 结果集中的可能性，p-skyline 包含所有 Skyline 概率至少是 p 的对象。它提出两种算法来计算不确定数据集的 Skyline 概率，一种是自底向上算法：选择不确定对象的一些实例计算 Skyline 概率，然后使用这些实例有效地删除掉其它实例和一些不确定对象；另一种是自顶向下算法：算法将不确定对象实例递归地分成几个子集，然后逐步删除一些子集和对象。两种算法在计算不确定数据 Skyline 上互相补充。Lian 等人[32]研究了不确定数据的反 Skyline 查询，它形式化了不确定数据上的概率反 Skyline 查询，提出了两种剪枝策略，即空间剪枝和概率剪枝来减少查询空间，并且联系剪枝策略实现了查询程序。另外基于 Skyline 的最好最大支配语义，提出了 PRFS 查询。Khalefa 等人[33]研究了不完整数据的 Skyline 查询处理，完整数据的支配关系是传递的，不完整数据的支配关系不具有这一特性，这可能导致循环的支配行为。它提出了替换(Replacement)和水桶(Bucket)算法，先使用传统的 Skyline 算法进行计算，然后使用 ISkyline 算法来处理那些不完整的数据，ISkyline 算法采用虚拟点和阴影 Skyline 两种优化策略来避免产生循环支配关系。此外，文献[34，35]研究了交通网的 Skyline 查询处理问题。Sharifzadeh 等人[34]对交通网中多源 Skyline 查询提出了解决方法，它是一个交通网中两点之间网络距离需要联机计算辅助的相对 Skyline 查询。绝对 Skyline 查询是根据数据集中点的静态数据值的最小值来计算 Skyline，相对 Skyline 查询是根据给定数据集中点和用户给定查询点之间差异的最小值来计算 Skyline。提出 3 种不同的处理算法来减少计算网络距离的开销：协同扩张算法(the Collaborative Expansion，CE)、欧式距离约束算法(the Euclidean Distance Constraint，EDC)和下界约束算法(the lower bound constraint，LBC)，其中 LBC 算法实例较优。

可以看出，已有的集中式环境下 Skyline 查询处理算法以及分布式、Web、数据流和交通网等环境下的查询处理算法在无线传感器网络环境中不可用。传感器网络中无线通信对传感器节点的能量消耗起主要影响作用，传感器节点将感知数据全部收集到基站进行 Skyline 查询处理不太现实。进而借助索引结构如 R 树来处理传感器网络中的 Skyline 也不可行。因此，针对无线传感器网络中节点的存储和计算处理能力有限的特点，研制能量高效的无线传感器网络中的 Skyline 查询处理算法是面临的重要问题。

3 Skyline 查询在无线传感器网络中的应用

针对无线传感器网络的数据处理，TinyDB[2]和 Cougar[3]仅仅支持类似于 SUM、MIN 和 MAX 等基本的聚集操作。随着传感器节点性能的提升，对无线传感器网络进行 Skyline 查询的需求日益增强。无线传感器网络中 Skyline 查询的典型应用包括：

1. 森林火灾监控

传感器节点可以感知其附近的温度、湿度和烟雾密度，当火灾发生时，火苗会使其附近的温度变高、湿度变低、烟雾密度变大，传感器节点对应的感知数据会发生相应变化；同时，温度较低但是烟雾密度较大的地方发生火灾的机率也较高。在森林中部署一个无线传感器网络来监测森林火灾，向无线传感器网络发起一个 Skyline 查询，查询那些温度较高、湿度较低同时烟雾密度较大或者湿度低、烟雾密度大的区域，系统会返回满足查询要求的 Skyline 结果集。这对森林火灾的监控非常有效，可以使消防人员很快识别出危险的地方并且及早采取行动。

2. 环境污染监测

将传感器节点布置在潜在的污染源附近，如核能发电厂、化工厂等。简单地说，一个传感器节点对应一个潜在的污染

源，可以用污染程度 dp 和污染源距离居民区的距离 $dist$ 来代表每个传感器节点。在一个大规模区域中部署无线传感器网络来监测环境污染状况，向无线传感器网络发送 Skyline 查询，查询那些距离居民区较近并且污染较为严重的点，系统返回结果即为急需治理的地方，这样就能优先治理这些潜在的污染源。

可以部署一个无线传感器网络来监测空气污染，诸如 CO 或 SO_2 等有毒气体的浓度较高时，这些区域空气污染严重，需要及时治理。用传感器节点感知 CO 等有毒气体的浓度，对部署的传感器网络发送 Skyline 查询，查询那些有毒气体浓度较高的区域，系统返回的结果可以为人们治理空气污染提供可靠依据。

此外，对于河流水质监测也可以使用 Skyline 查询。在河流中部署一个无线传感器网络，使用传感器节点感知水中大肠杆菌数量、PH 值、盐度和浑浊度等。向无线传感器网络发送一个 Skyline 查询，查找那些可能大肠杆菌量超标、PH 值过低的地方，以及那些盐度过多、浑浊度过大的地方，这样就能针对不同区域进行水质治理。

3. 野生动物行为检测

户外生物学家可以通过无线传感器网络采集鸟类的信息以分析其行为习惯，如果一个生物学家研究区域内鸟类群体之间的生态关系，那么他希望观测区域中鸟的种类较多且数量较大。可以在森林中部署一个无线传感器网络来感知这些条件，向网络发送 Skyline 查询，返回两个方面或某个方面的值较大的区域，以为生物学家研究动物行为提供参考。

此外，对野生鱼类进行研究时，如果科学家需要寻找生活在一定水温、一定水流速度并具有一定种群数量的鱼群作为研究对象，可以在水域中部署一个无线传感器网络来监测水流速、温度以及鱼的数量，向传感器网络发送 Skyline 查询，查找最接近于这一查询条件的区域，返回的结果可为研究人员提供参考，从中选择一个作为研究区域。

4. 泥石流灾害监控

泥石流是指在山区或者其它沟谷深壑，地形险峻的地区因暴雨暴雪等而引发的山体滑坡并携带有大量泥沙以及石块的特殊洪流。也就是说易发生泥石流的地方必须有以下 3 个条件：大量水、大量泥沙、险峻的地形，可以用传感器节点来感知这些条件。在可能发生泥石流的区域内部署一个无线传感器网络来监控泥石流灾害，向无线传感器网络发送 Skyline 查询，查询以上 3 个方面感知数据都较大的地方，系统返回的 Skyline 结果集即为需要特别关注的危险区域，及早对这些区域进行治理可以有效防止泥石流灾害发生。

5. 园艺环境监测

可以用传感器网络来监测园艺环境，为园丁提供有用的服务支持。如果园丁想要提高兰花开花率，那么兰花的发芽期是需要关注的，兰花一般在环境温度较低、湿度较高的情况下易萌芽。然而现实情况是，一般温度较高的地方水分蒸发在空气中，这样才会有一个较高的湿度，所以温度较低但湿度又高的地方需要认真检测才能找到，这就会耗费大量的时间和精力。可以在公园中部署一个无线传感器网络来采集温度、湿度、光照等条件，向传感器网络中发送一个 Skyline 查询，查询那些符合条件的区域，系统返回的 Skyline 结果能够为园丁提供一个参考区域。

由此可见，无线传感器网络中的 Skyline 查询在人们的生产生活中应用极为广泛。研究者们也从各个方面给出了无线传感器网络中的 Skyline 查询处理方法。

4 传感器网络中的 Skyline 查询处理技术

为了有效应答传感器网络中的 Skyline 查询，基本出发点就是过滤与查询无关的元组，从而避免不必要的无线通信以减少传感器节点的能量消耗。现有国内外研究主要是针对过滤思想从不同角度处理无线传感器网络中的 Skyline 查询。

4.1 基于过滤元组的策略

Huang 等人[36]对于移动自组织网络中的 Skyline 计算提出了基于过滤元组的方法，这可以简单地扩展到无线传感器网络环境中，旨在降低移动传感器节点之间的通信代价并减少在每一个节点处的执行时间。直接策略是将所有移动节点的本地 Skyline 汇集于查询节点之后计算 Skyline 结果，然而这些本地 Skyline 中有很大一部分元组并不包含于最终 Skyline 中，传输这些元组会消耗大量能量。为了实现网络的能量高效性，提出了基于过滤元组的策略。

1. 直接策略

假设移动节点 M_{org} 发起一个 Skyline 查询请求：$Q_{ds}=(id, pos_{org}, d)$，其中 id 为节点 M_{org} 的标识，pos_{org} 为节点 M_{org} 的位置，d 为节点 M_{org} 与感兴趣区域的距离。M_{org} 在发送查询请求后计算自身的 Skyline，当移动传感器节点 M_i 收到一个查询请求时，在它自身的关系 R_i 上进行一个 Skyline 查询并返回 M_{org} 最终结果。最后 M_{org} 以一个增量的模式将接收到的数据和之前的结果进行合并得到最终的 Skyline。

2. 基于过滤元组的策略

基于过滤元组的策略中除了全局的 Skyline 集 SK 之外，每个传感器节点 N_i 保存一个本地 Skyline 集 SK_i。由于在 SK_i 中，可能存在不满足 SK 的元组，那么阻止这些元组的传输可以大大降低通信代价。如果 SK_i 中的元组 tp_i 没有出现在最终的 SK 中，那么至少有一个在 SK 中而不在 SK_i 中的元组 tp_j 支配 tp_i。

因为在移动传感器节点 M_{org} 上也进行了 Skyline 的查询，所以可以在 SK_{org} 中选择一个元组 tp_j。对于元组 $tp_j(p_{j1}, p_{j2}, \cdots, p_{jn})$，其支配其它元组的能力由一个超矩形决定，这个超矩形的对角线是 tp_j 的坐标与数据空间最大角之间的连线，如图 2 所示。假设在 p_k 维度上的数据范围是$[s_k, b_k]$，那么支配区域 $VDR_j=\prod(b_k-p_{jk})$，k 从 1 到 n。选择使 VDR_j 达到最大的 tp_j 作为过滤元组 tp_{flt}。在 M_{org} 发送查询请求的同时发送过滤元组，每个节点 M_i 在进行本地 Skyline 查询处理时使用 tp_{flt} 过滤掉不满足条件的元组。

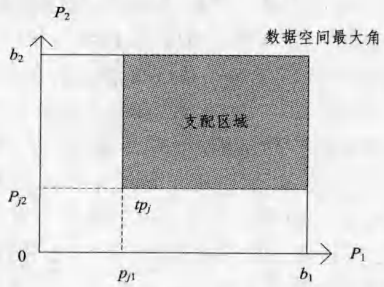


图 2 元组的支配区域

以 Nassau 海滩查找旅馆为例，假设节点 M_1 和 M_2 对应的关系如表 1(a)和表 1(b)所列，M_1 发起查询，查找距离近且

价格低的旅馆。M_1 的 Skyline=$\{h_{11},h_{12},h_{13}\}$，M_2 的 Skyline=$\{h_{21},h_{22},h_{23}\}$，如果不使用过滤元组，M_2 需要发送 3 个元组至 M_1。假设价格和距离的最大约束是 200 和 10，从 M_1 的本地 Skyline 中选择一个过滤元组，计算 $VDR_{11}=(200-60)*(10-3)=980$，$VDR_{12}=960$，$VDR_{13}=540$，所以选择 h_{11} 为过滤元组。这样在 M_2 可以过滤掉 h_{22} 和 h_{23}，从而避免了这些元组的传输，减少了能量的消耗。

表 1　节点 M_1 和 M_2 对应的关系

(a) M_1 的关系 R_1

旅馆	价格	距离
h_{11}	60	3
h_{12}	80	2
h_{13}	140	1
h_{14}	150	4

(b) M_2 的关系 R_2

旅馆	价格	距离
h_{21}	30	8
h_{22}	60	5
h_{23}	80	3
h_{24}	100	4

4.2　基于簇的网内聚集方法 MFTAC

KWON 等人[11]针对无线传感器网络中的 Skyline 查询提出了网内处理算法，即使用基于簇的网内聚集的过滤算法(MFT-Applied Aggregation Method Based on Cluster，MFTAC)来避免非 Skyline 元组在传感器节点之间的传输，算法减少了通信代价并且平衡了负载。

MFTAC 算法的剪枝策略是使用最小得分的过滤元组(Min-Score Filter Tuple，MFT)来过滤掉不满足 Skyline 查询语义的元组，以避免其向查询节点传输。MFT 过滤掉的非 Skyline 元组越多，网内需要传输的数据就越少，因此 MFT 的选择直接影响了算法的性能。越靠近原点的元组，支配其它元组的可能性越大，所以 MFTAC 算法选择最靠近原点的元组作为 MFT。使用 MFTAC 算法进行 Skyline 查询处理包含两个阶段。

1) 每个传感器节点首先计算其本地 MFT。当一个 Skyline 查询到达根节点时，根节点将查询连同其本地 MFT 经路由树发送至其孩子节点。MFT 根据其过滤能力动态变化。中间节点收到 Skyline 查询时，比较本地 MFT 与收到的 MFT 的过滤能力，如果本地的 MFT 过滤掉的不满足条件的元组较多，则将本地 MFT 向下传递。最终，叶子节点获得最具有支配能力的 MFT。

2)由第一阶段获得的 MFT 从叶子节点向根节点汇聚。每个中间节点首先使用 MFT 来过滤掉本地的点，然后计算其本地 Skyline 并将结果传送到父节点。最后，根节点收到其孩子节点发送的本地 Skyline 后计算整个数据集的最终 Skyline。

图 3 为 MFTAC 算法实例，无线传感器网络中有 5 个节点，分为两个簇，如图 3 虚线圈所示，其中 S_1 和 S_2 为簇首节点。当传感器节点 S_1 接收到基站 BS 发起的 Skyline 查询后，S_1 选择 P_{12} 作为 MFT_1，并将 Skyline 查询和 MFT_1 发送至 S_2 和 S_3。传感器节点 S_2 使用 P_{22} 更新 MFT_2，并将 Skyline 查询和 MFT_2 发送至 S_4 和 S_5。节点 S_4 和 S_5 使用 MFT_2 过滤掉非 Skyline 点，计算本地 Skyline。因为 S_4 中所有点都被 MFT_2 支配，所以不上传任何点，同理 S_5 上传点 P_{52} 至 S_2。节点 S_2 计算出此时本地 Skyline 为 P_{22} 和 P_{52}，并将结果上传至 S_1。因为节点 S_3 中其它点都被 MFT_1 支配，所以 S_3 只上传 P_{31} 至 S_1。最后 S_1 通过合并孩子节点上传数据过滤掉 P_{13}，得到最终 Skyline 集$\{P_{12}, P_{22}, P_{31}, P_{52}\}$。

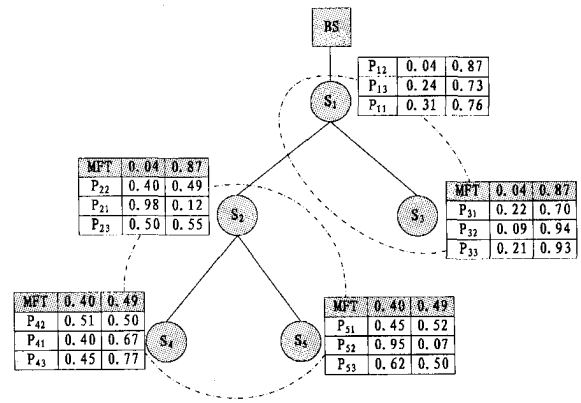


图 3　MFTAC 算法实例

MFTAC 算法的缺点是连同查询一同发送 MFT 增加了传感器网络的传输代价，随着传感器节点数量的增加，传输代价会增大。所以在无线传感器网络中，MFTAC 算法对于减少通信代价并不高效。

4.3　MinMax 阈值方法

为了减少整个网络的通信流量，Chen 等人[12]首次研究了无线传感器网络中的连续 Skyline 查询问题，提出了基于阈值的分层方法及其改进方法 MinMax。

1. 基于阈值的分层方法

阈值是从本地数据和孩子节点发送的数据中选取的一个元组，基于阈值的分层方法是使用一个阈值在中间节点去掉那些被支配的元组。阈值的选取需要满足两条规则：a)父节点阈值支配或等于子节点的阈值；b)除自身阈值外节点还需维护其孩子节点的阈值。基于分层阈值的方法包含两个阶段：向上报告和向下查询。

1)向上报告阶段：叶子节点报告不被阈值支配的点；中间节点计算自身及其孩子节点报告数据的 Skyline，并将结果报告其父节点，同时存储未报告的点；根节点合并所有孩子节点上传的数据得到 Skyline 集。

2)向下查询阶段：当孩子节点的阈值不能被根节点 Skyline 中某些点支配时，根节点向其孩子节点发送本地 Skyline 集。中间节点通过接收父节点的 Skyline 集和本地未报告的数据来计算新的 Skyline，并按照根节点规则向下发送新的 Skyline 结果，所有孩子节点响应后，更新本地的 Skyline 集。叶子节点如果之前被上报或者被支配，则不上报。

假设无线传感器网络中有 5 个节点，树状结构如图 4 所示，S_i 代表传感器节点，P_i 代表 S_i 的值，F_i 代表 S_i 本地过滤元组。向上报告阶段，叶子节点 S_3、S_4 和 S_5 中 P_3 被 F_3 支配、P_4 被 F_4 支配均不上报，F_5 不能支配 P_5，所以 S_5 向 S_2 发送 P_5；中间节点 S_2 中因为 P_2 和 P_5 都不被 F_2 支配且不可比较，所以 S_2 将 P_2 和 P_5 发送至 S_1；根节点 S_1 比较 P_1、P_2、P_5，得出此时 Skyline 结果集为 P_2 和 P_5。向下查询阶段，Skyline 结果集不能支配 S_2 的过滤元组 F_2，所以 S_1 向 S_2 发送 P_2 和 P_5；中间节点 S_2 接收到根节点的 Skyline 结果集，与本地未报告数据合并形成新的 Skyline，因为 S_2 本地无未报告数据，所以 S_2 对 Skyline 结果无影响。过滤元组 F_4 不能被结果集支配，所以 S_2 向 S_4 发送 P_2 和 P_5，因为 P_4 也不被支配，所以它也是 Skyline 的一部分，将新的结果返回给 S_2 节点。因此，最终的 Skyline 结果集为 P_2、P_4 和 P_5。

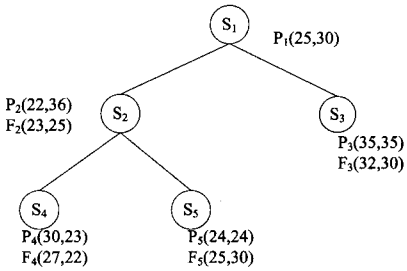


图 4　无线传感器节点树状结构

2. MinMax 方法

MinMax 方法主要改进了基本阈值方法中过滤元组的选择策略。在每个传感器节点计算本地每个元组维度最大的属性值，在这些值中选择最小的那个值，假设最小值为 min，d 为元组的维度，则过滤元组为 $(\min,\min,\cdots,\min)_d$。假设传感器节点 S_i 有 2 维数据的 4 个元组 P_i，如表 2 所列，则 P_1 到 P_4 的维度最大的属性值分别为 40、32、30、42，由此得 min 值为 30，所以过滤元组为(30,30)。如图 5 所示，灰色区域为支配区域。根据这个方法对其它的节点计算过滤元组。最后，根节点在收到所有孩子节点的数据后合并所有的结果产生 Skyline 结果集。

表 2　传感器节点 S_i 中的元组

元组	d_1	d_2
P_1	15	40
P_2	20	32
P_3	30	23
P_4	42	18

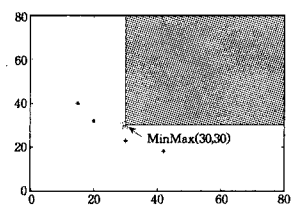


图 5　MinMax 阈值

MinMax 阈值处理的方法可以明显减少网络上的传输元组从而减少通信代价。但是因为每个传感器节点的阈值都是本地计算的，所以会导致许多本地 Skyline 元组不一定是全局 Skyline，无法避免这些元组的传输，从而增加了这些元组的传输代价。

4.4　滑动窗口 Skyline 监控算法

Xin 等人[13]提出了一个滑动窗口 Skyline 监控的算法(Sliding Window Skyline Monitoring Algorithm，SWSMA)来持续维护传感器网络中的滑动窗口 Skyline，SWSMA 算法采用基于过滤的思想。在传感器网络中每个节点按照固定的周期采集数据，只要节点没有失效，这是一个反复进行的过程。节点数据实时产生，将所有的数据收集到基站再进行 Skyline 查询处理不能实现能量高效性，同时在整个数据流上计算 Skyline 也难以实现，因此，只计算滑动窗口内也就是最新产生的一些数据的 Skyline，便可很好地进行数据监控。SWSMA 算法中主要使用基于元组过滤的滑动窗口 Skyline 查询处理算法，来减少一些非 Skyline 结果的元组的传输，从而减少传感器节点间数据的传输量，节省能量消耗。

为减小传输数据量的大小，SWAMA 引入了两种过滤方法，即元组过滤方法和网格过滤方法来去除那些被支配的元组。同时为适应不同的数据分布，提出了自适应过滤方法。

1. 元组过滤方法

根节点决定支配最多元组个数的元组为过滤器 T_f，并将 T_f 广播到整个网络。每个传感器节点使用 T_f 来去掉那些被其支配的元组，中间节点合并自身及孩子节点的 Skyline，并将结果发送给父节点。那些没有被支配的元组通过路由树传到基站，最后基站合并得到最终 Skyline 结果集。如图 6 所示，假设将过滤元组 T_f 设置为(0.22,0.3)，那么落入图中灰色区域的点都会被 T_f 支配，直接丢掉，不向其父节点发送。其余没有被 T_f 支配的点，经过路由树汇聚于基站，基站对这些点做进一步处理。

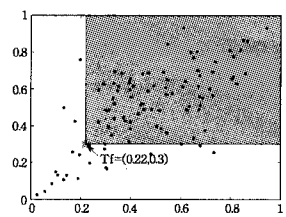
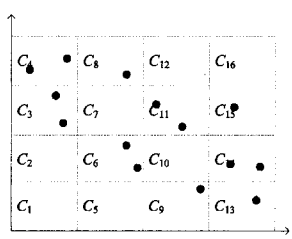


图 6　元组过滤方法

2. 网络过滤方法

用一个网格来划分数据空间，每一维被分成 s 个片段，对于 d 维的数据而言，一共分成 s^d 个 *cell*，如图 7(a)所示，假设数据是 2 维的，每维分成 4 片，那么一共有 C_1 到 C_{16} 共 16 个 *cell*，用 *cell. sta* 来记录状态。



(a) 网格过滤中的分片

C_4 1	C_8 1	C_{12} 0	C_{16} 0
C_3 1	C_7 0	C_{11} 1	C_{15} 1
C_2 0	C_6 1	C_{10} 0	C_{14} 1
C_1 0	C_5 0	C_9 1	C_{13} 1

(b) 初始网格

C_4 1	C_8	C_{12} 0	C_{16} 0
C_3 1	C_7 0	C_{11} 0	C_{15} 0
C_2 1	C_6 1	C_{10} 0	C_{14} 0
C_1 1	C_5 1	C_9 1	C_{13} 1

(c) 预处理网格

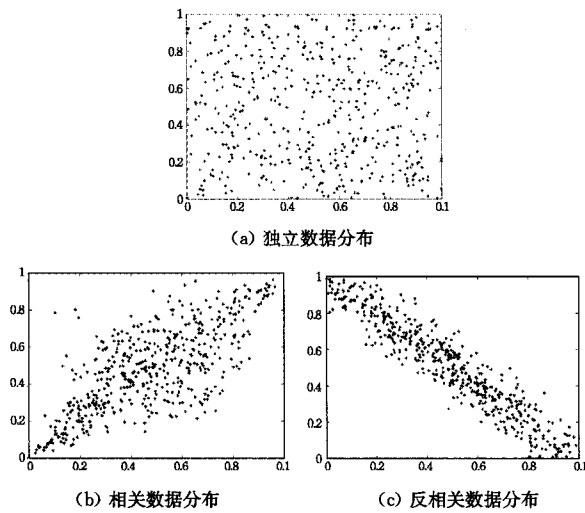
图 7　网格过滤方法

对网格的处理分为两种，一种是初始网格，如果至少有一个点落入到这个 *cell* 中，那么将 *cell. sta* 置为 1，否则为 0。如图 7(b)所示，有点落入 C_8、C_{11} 等 *cell* 中，所以将这些 *cell* 的 *cell. sta* 置 1，其余为 0。另一种为预处理网格，将被支配的 *cell* 的 *cell. sta* 置 0，意味着这里的元组都不会属于 Skyline，其余为 1。如图 7(c)所示，因为 C_6 可以支配 C_{11} 和 C_{15} 中的点，所以将后两个 *cell* 置 0，同时 C_6 不被任何 *cell* 支配，置 1，其它同理。判断一个元组是否被网格支配，初始网格方法需要考虑所有 *cell* 的状态，而预处理的格只需要考虑一个 *cell* 的状态，预处理网格方法要优于初始网格方法。这里采用第二种方法，将预处理网格发送至各个传感器节点来过滤元组，如果元组落入 *cell. sta* 为 0 的 *cell* 中，那么就被丢弃，否则发

送至父节点。

3. 自适应过滤方法

SWAMA 算法通过对传感器节点数据进行随机采样或者用传感器节点的采样直方图来确定网络的数据分布情况，数据分布一般分为独立分布、相关分布和反相关分布。假设数据是两维的，如图 8 所示，每个点代表被检测的数据。图 8(a)表示独立数据分布，不同维上的数据是独立生成的，互相没有联系。图 8(b)表示相关分布，数据如果在一个维度上具有优势，在其它维度上同样具有优势。图 8(c)表示反相关分布，数据如果在一个维度上具有优势，在其它维度上往往较弱。元组过滤和网格过滤方法各有优缺点，元组过滤更适用于独立和相关数据分布，而网格过滤更适用于独立分布。因此，自适应过滤方法根据数据的具体分布情况决定采用哪种过滤策略，如果数据是近似独立或相关的，采用元组过滤方法；如果数据是近似反相关的，采用网格过滤方法。



(a) 独立数据分布

(b) 相关数据分布　　(c) 反相关数据分布

图 8　3 种数据分布

SWSMA 算法能够减少很多非 Skyline 元组的传输，从而减少了通信代价。但元组过滤中需要从所有传感器节点获取本地 Skyline 来确定 T_f，网格过滤中需要获取过滤网格，最后需要将这些结果再广播到所有节点，它的预处理过程比较消耗能量。

4.5 SkySensor 算法

Su 等人[37]提出了从无线传感器网络中有效地获取 Skyline 结果的 SkySensor 算法(Skyline Sensor Algorithm)。SkySensor 算法避免了从传感器网络中的所有节点中获取数据，从而大大节省了能量。SkySensor 算法采用簇架构来获取所有传感器节点的数据。属于簇的传感器节点称为存储节点，用于存储数据。无线传感器网络中簇的个数依赖于元组的属性个数，如果元组的维度是 d，那么将网络分成 d 个簇，即 $C_1, C_2, \cdots, C_d$。如果检测元组 t 第 i 维上的值最小，那么 t 存储在簇 C_i 中，例如 $t=(0.3, 0.22, 0.1)$，因为 t 第三维的属性值最小，所以 t 被分配到 C_3。$S_{i,j}$ 代表 C_i 的第 j 个存储节点，$S_{i,1}$ 决定簇中每个节点的数据范围，假设簇中有 k 个节点，$S_{i,j}$ 的数据范围为 $[L_{i,j}, U_{i,j})=[(j-1)/k, j/k)$。

SkySensor 算法的查询处理机制采用剪枝和数据获取两阶段来实现 Skyline 计算的高效性。剪枝阶段主要修剪掉尽可能多的不包含最终 Skyline 的传感器节点并过滤掉传感器节点中的非 Skyline 点。数据获取阶段从剩下的传感器节点中获取数据生成最终 Skyline。

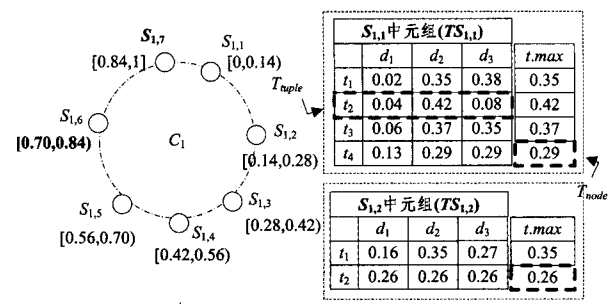
1. 剪枝阶段

查询节点向距其最近的簇(假设为 C_i)发起 Skyline 查询，假设 $S_{i,j}$ 为第一个接收到查询的节点，如果 $S_{i,j}$ 发现当前簇的 ID 在簇 ID 表中有记录，表明其为第一个接收到查询的存储节点，$S_{i,j}$ 从簇 ID 表中删除当前簇 ID。然后 $S_{i,j}$ 在当前簇内选择一个合适的存储节点 $S_{i,k}$ 作为处理 Skyline 查询的第一个节点。最合适的存储节点 $S_{i,k}$ 具有以下两点属性：a) $S_{i,k}$ 中存储至少一个元组；b)在所有满足上一条件的节点中，$L_{i,j}$ 具有最小的值。最后进入剪枝阶段，包括以下 6 个步骤：

1)存储前驱节点的 ID：$S_{i,k}$ 接收到一个查询后，记录查询来自哪个存储节点，即记录其前驱节点的 ID。

2)执行本地支配测试：$S_{i,k}$ 进行本地支配元组测试，使用 4)中的过滤元组阈值过滤掉不满足条件的元组。

3)选择修剪节点阈值：$S_{i,k}$ 选择修剪节点阈值来过滤掉无须访问的存储节点，假设 T_{node} 为修剪节点阈值，元组 t 的最大属性值为 $t.\max$，$TS_{i,k}$ 为节点 $S_{i,k}$ 存储的元组。假设 $\text{minmax}(TS_{i,k})=\min\{t_j.\max \mid t_j \in TS_{i,k}\}$，则 $T_{node}=\text{minmax}(TS_{i,k})$，由此所有 $L_{i,m} > \text{minmax}(TS_{i,k})$ 的存储节点都可以被删除。以图 9 为例，$\text{minmax}(TS_{1,1}) = 0.29$，所以 $T_{node}=0.29$，修剪后簇中只剩 $S_{1,1}$，$S_{1,2}$ 和 $S_{1,3}$ 需要考虑，明显减少了通信代价。



$S_{1,1}$ 中元组($TS_{1,1}$)

	d_1	d_2	d_3	$t.max$
t_1	0.02	0.35	0.38	0.35
t_2	0.04	0.42	0.08	0.42
t_3	0.06	0.37	0.35	0.37
t_4	0.13	0.29	0.29	0.29

$S_{1,2}$ 中元组($TS_{1,2}$)

	d_1	d_2	d_3	$t.max$
t_1	0.16	0.35	0.27	0.35
t_2	0.26	0.26	0.26	0.26

图 9　修剪节点阈值与过滤元组阈值

4)选择过滤元组阈值：尽管 T_{node} 能够过滤掉很多存储节点，但在剩下的节点中还有一些不满足条件的元组，SkySensor 使用一个过滤元组来删除非 Skyline 点。假设 T_{tuple} 为过滤元组阈值，使用过滤元组策略中的支配区域[33]来决定 T_{tuple}。

5)更新阈值：当 $S_{i,k}$ 决定 T_{node} 和 T_{tuple} 后，将查询连同阈值一起发送至当前簇内的下一个存储节点，例如图 9 中的 $S_{1,2}$。$S_{1,2}$ 接收到查询后，比较接收到的阈值与本地阈值，如果本地阈值支配能力更优，则更新阈值。图 9 中节点 $S_{1,2}$ 的 $T_{node}=0.26$ 小于节点 $S_{1,1}$ 的 T_{node}，更新修剪节点阈值，因为节点 $S_{1,2}$ 的 T_{tuple} 不优于节点 $S_{1,1}$ 的 T_{tuple}，不更新过滤元组阈值。

6)选择后继节点：$S_{i,k}$ 结束上述操作后，以顺时针方向查找当前簇中下一个存储节点 $S_{i,k+1}$，并将查询发送至 $S_{i,k+1}$，有效的后继结点 $S_{i,k+1}$ 满足这两个条件：a) $S_{i,k+1}$ 有存储数据；b) $S_{i,k+1}$ 不被 T_{node} 支配。如果 $S_{i,k}$ 检测到其当前簇内顺时针邻居节点被 T_{node} 修剪掉，那么 $S_{i,k}$ 即为当前簇内最后一个节点，此时 $S_{i,k}$ 查找簇 ID 表并选择距其最近的簇发送查询。如果 $S_{i,k}$ 检测到其为无线传感器网络中最后处理的节点，则开始执行数据获取阶段。

2. 数据获取阶段

在剪枝阶段，每个存储节点记录了其前驱节点的 ID，SkySensor 算法根据这个信息建立查询路由，从路径的最后

一个节点开始，沿路由从每个节点采集可能为 Skyline 的数据。假设存储节点 $S_{i,k}$ 为网络中最后一个处理查询的节点，$S_{i,k}$ 将本地 Skyline 数据发送至其前驱节点，每个中间节点合并其后继结点发送的数据和本地 Skyline，然后将结果发送至其前驱节点。最后，路由的第一个存储节点得到 Skyline 结果并将其发送至查询节点。

以图 10 为例，假设数据有 3 个属性，则无线传感器网络分为 3 簇，SkySensor 算法建立路由路径，如图中虚箭头所示，实箭头为数据采集阶段路径，用×标记无须访问的存储节点，每个需访问的存储节点中的数据如表 3 所列。$S_{3,2}$ 为网络中处理查询的最后一个存储节点，$S_{3,2}$ 向其前驱节点 $S_{3,1}$ 发送 t_8，$S_{3,1}$ 比较 t_8 和本地 Skyline 元组 t_7，t_8 被支配，所以 $S_{3,1}$ 仅将 t_7 转发到簇 C_2。$S_{2,3}$ 接收到 t_7 后和本地 Skyline 比较，因为 t_6 和 t_7 没有支配关系，所以 $S_{2,3}$ 将 t_6 和 t_7 发送至 $S_{2,2}$。重复这一过程，直至路由中的第一个存储节点 $S_{1,1}$；$S_{1,1}$ 计算出最终的 Skyline 为 $\{t_1, t_5, t_6, t_7\}$，并将结果发送至查询节点。

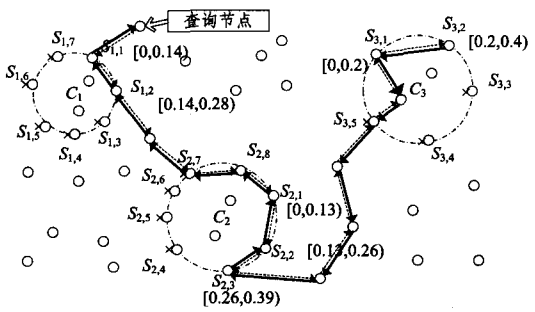


图 10　SkySensor 数据获取过程

表 3　存储节点存储元组

节点 $S_{i,j}$	元组 t_i	d_1	d_2	d_3
$S_{1,1}$	t_1	0.12	0.56	0.3
$S_{1,2}$	t_2	0.27	0.3	0.55
$S_{2,1}$	t_3	0.3	0.12	0.16
$S_{2,2}$	t_4	0.6	0.2	0.4
	t_5	0.25	0.16	0.47
$S_{2,3}$	t_6	0.27	0.27	0.45
$S_{3,1}$	t_7	0.3	0.1	0.01
$S_{3,2}$	t_8	0.33	0.53	0.26

SkySensor 算法保证簇中返回的结果肯定是全局的 Skyline 结果，这样就不需要全局聚集，因此所有传感器节点的计算代价比较平均，其通信代价和查询处理代价都较低，并且随着网络规模的增大和数据维度的增加，SkySensor 算法较稳定。

4.6　数据划分算法

Liang 等人[38]提出基于元组过滤的方法(Filter-Based)，用以最大化网络的生命周期，它把多个而不是一个点作为过滤器。算法思想是：首先找到一个全局的 Skyline 过滤器，它包括潜在的全局 Skyline 点；然后使用全局过滤器通过网内处理的方法，在那些没有被支配的点中找到一个可能的全局 Skyline 的子集；最后使用从全局 Skyline 过滤器中找到的一个可能子集中的点，去除被支配的点。最后的结果由保留的全局过滤器点以及找到的全局 Skyline 点组成。在此基础上，该课题小组继续研究了无线传感器网络 Skyline 查询的优化以及维护问题[39,40]。其中使用了两种策略，一种是将数据分为不相交的几个子集，然后逐步计算每个子集产生的 Skyline 点；另一种是使用全局过滤器，它由已处理过子集中的一些 Skyline 点组成，用来去除掉那些还未检测的子集中不可能成为 Skyline 的点。同时在滑动窗口环境中，提出了对流数据库的增量维护策略，从过去的 Skyline 结果中选取一个全局过滤器并决定何时广播这个过滤器至所有传感器节点。

点 p 具有 d 个维度，$p=(p_1, p_2, \cdots, p_d)$，那么 $R(p)=(p_1^2+p_2^2+\cdots+p_d^2)^{1/2}$ 为点 p 的半径。假设 $R(p)$ 和 $R(q)$ 为点 p 和 q 的半径，如果 $R(p)\leqslant R(q)$，那么点 p 不能被点 q 支配。

1. a-FDP 和 g-FDP 算法

假设 $R(P)_{\max}$ 和 $R(P)_{\min}$ 分别为数据集 P 的最大和最小半径。k 事先给定，将 P 划分成 k 个不相交的子集，$P_1, P_2, \cdots, P_k$，对于所有的 $1\leqslant i<k$，$R(P_i)_{\max}<R(P_{i+1})_{\min}$，子集 P_i 的分区半径为 $R(P_i)_{\max}$。在算法的第 i 次迭代中，检查 P_i 中的点并找到一个新的 Skyline 集 SK_i，SK_i 中的每个点都不能被已找到的 Skyline 点支配。最后 P 的 Skyline 即为 $SK_i(1\leqslant i<k)$ 的并集。

$R(P)_{\max}$ 和 $R(P)_{\min}$ 通过网内聚集的方法产生，算法首先产生 k 个半径 $R(P_i)_{\max}$ 的升序序列。划分半径序列可以是算数或几何序列，分别称为算数固定数据划分(arithmetic-Fixed Dataset Partition，a-FDP)和几何固定数据划分算法(geometric-Fixed Dataset Partition，g-FDP)。其次，算法建立路由树，进行 k 次迭代。$LSK_i(v)$ 为传感器节点 v 本地点及其孩子上传点的 Skyline 集。$LF_i(v)$ 为第 i 次迭代 v 节点的本地过滤器。第 i 次迭代时，传感器节点 v 中的点如果被 $LF_i(v)$ 中的点支配，则被过滤掉。如果 v 是叶子节点，那么发送 $LSK_i(v)$ 中的半径不大于 $R(P_i)_{\max}$ 的点至其父节点；否则，节点 v 计算其本地点及收到点的 $LSK_i(v)$，然后发送其中半径不大于 $R(P_i)_{\max}$ 的点至其父节点，最后基站 r 计算收到所有点的 Skyline 集 $LSK_i(r)$。$SK_i=\{p \mid p\in LSK_i(r), \forall q\in\bigcup_{j=1}^{i-1} SK_j, q \prec p\}$ 为第 i 次迭代新找到的 Skyline 点。最后 SK_i 中的一些点会被广播到传感器网络中的节点，以过滤掉没有被检测的子集中的非 Skyline 点。其中第 i 次迭代广播全局 Skyline 点 GSF_i，每个传感器节点根据 GSF_i 更新本地过滤器，$LF_{i+1}(v)=LF_i(v)\cup GSF_i$。

2. α-DDP 算法

尽管 a-FDP 和 g-FDP 算法性能优于一般的合并算法，但其并未明显降低传感器网络总的能量消耗以及最大能量消耗。首先，需要对路由树花费一次额外的迭代过程来计算 P 中点的最大及最小半径，这期间不传送任何 Skyline 点。其次，对于一个具有偏斜数据分布的数据库而言，固定划分半径不能避免有些子集可能是空的。然而即使子集为空，a-FDP 和 g-FDP 算法仍会执行 i 次迭代，这样就大大消耗了能量。此外，因为 k 需要事先给定，选择一个合适的 k 是非常困难的。为避免这些缺点，提出了动态数据划分(Dynamical Dataset Partition，DDP)的思想。

DDP 算法中的子集数 k 不是事先给定的，它由 P 中数据的数值分布决定，每个子集的划分数据半径动态生成。在每次迭代时，每个传感器节点动态决定它可以上传至父节点的点个数的上层约束半径。最终点被传至基站后，这个子集的划分数据半径就决定了，包含在这个子集中的点的半径要在现在和之前的划分数据半径之间。

$UR_i(v)$ 为传感器节点 v 在第 i 次迭代中划分数据半径的

上层约束半径。如果 v 是叶子节点，$UR_i(v)$ 由 v 上传的所有点的最大半径决定。如果 v 是中间节点，假设 v 有 d_v 个孩子节点，每个孩子 u_j 发送那些半径小于 $UR_i(u_j)$ 的点给节点 v，v 计算 $LSK_i(v)$。$UR_i(v)=\min\{R(LSK_i(v))_{max}, UR_i(u_1), \cdots, UR_i(u_{dv})\}$。基站划分数据半径的 $UR_i(r)$ 比网络内其它所有的节点都小，$UR_i(r)$ 为第 i 个子集的划分数据半径。

使用 α-DDP 算法，$LSK_i(v)$ 中的点根据它们的半径按照升序排序，如果 v 是叶子节点，只发送前一部分点给父节点。参数 α 的使用可以减少叶子节点的划分数据半径的上层约束，同时也减小了每个子集的划分半径，从而增加了迭代次数 k。划分的子集数越多，属于 Skyline 的点被过滤的可能性就越小。尽管 α-DDP 增加了迭代额次数，但是它有效地减少了传感器节点间数据传输所消耗的能量。

假设无线传感器网络中有 8 个节点，结构如图 11 所示，S_i 代表传感器节点，假设 α 为 0.5，用这个例子来介绍 α-DDP 算法。在第一次迭代过程中，每个叶节点发送其自身 Skyline 结果有较小半径的前$\lceil(\alpha * |LSK_i(v)|)\rceil$的点，也就是前一半的点。$S_6$、$S_7$ 和 S_8 分别向上发送点(12,733)、(65,579)和(1976,28)、(939,9)，因为叶子节点的传输数据半径的上层约束为这些上传点的最大半径，所以 $UR_1(S_6)=733.1$，$UR_1(S_7)=1976.2$，$UR_1(S_8)=939.04$。

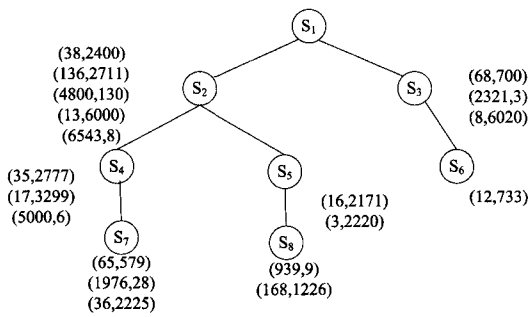


图 11　传感器网络数据

在接收到子节点 S_7 发送的点后，传感器节点 S_4 计算本地 Skyline，并求出 $LSK_1(S_4)$ 的最大半径 $R(LSK_1(S_4))_{max}=5000$，也就是对应点(5000,6)的半径。$UR_1(S_4)=\min\{R(LSK_1(S_4))_{max}, UR_1(S_7)\}=1976.2$，因此，传感器节点 S_4 向 S_2 上传半径小于 $UR_1(S_4)$ 的点(65,579)和(1976,28)。按照同样的方法处理其它节点，最后，根节点 S_1 接收到所有孩子节点上传的点后，计算本地 Skyline，$LSK_1(S_1)=\{(65,579),(939,9),(12,726)\}$，$R(LSK_1(S_1))_{max}=733.1$，因此第一次迭代之后，计算出 Skyline 点 $SK_1=\{(65,579),(12,726)\}$。

在得到 SK_1 后，那些在某个维度上具有最小值的点被添加到 GSF_1，因此点(65,579)和(12,726)作为全局过滤器广播到每个传感器节点。传感器节点在收到 GSF_1 后过滤掉被其支配的点。如果传感器节点中剩余的点都被支配了，则终止迭代，否则继续下一次迭代。最后得到整个数据集的 Skyline $=\{(65,579),(12,726),(939,9),(3,2220),(2321,3)\}$。

α-DDP 算法能够减少大量元组的传递，降低了通信代价，在独立、相关和反相关数据分布以及真实数据库中都具有较低的能量消耗。

结束语　能量有限是无线传感器网络应用的主要约束条件，在应答 Skyline 查询时，避免节点间的通信次数是减少能量消耗的有效途径。本文在论述了传统数据库环境中的 Skyline 计算方法后，介绍了 Skyline 查询在无线传感器网络中的应用场景，进而探讨了无线传感器网络中的 Skyline 查询处理方法。由于无线传感器网络的能量、计算和存储能力有限等特点，其 Skyline 查询算法注重减少能量的消耗。已有的无线传感器网络环境中的 Skyline 查询处理技术包括 Min-Max 阈值方法、SWSMA 算法、SkySensor 算法等，这些算法使用不同的过滤策略，通过过滤掉非 Skyline 结果集元组达到减少传输元组数量的目的，从而降低通信代价，实现无线传感器网络中 Skyline 查询处理的有效性。

通过对这些研究的分析总结，给出对无线传感器网络中 Skyline 查询处理技术研究的几个方向：

(1)已有算法中大多使用基于元组过滤策略，如何根据查询的具体过程和应用特性选择一个或者一组有效的元组进行过滤有待进一步研究。

(2)传感器网络数据是不断产生的，将所有数据收集完后再进行 Skyline 查询处理是不现实的，使用滑动窗口机制可以避免过多的上传数据以及只关注近期感兴趣的数据，因此设置滑动窗口大小以及研制滑动窗口中的元组置换策略值得进一步研究。

(3)可以根据传感器节点的地理位置和数据具体数值的近似程度进行分区，每个分区的传感器数据相近，因此可以先在不同的分区中对整个网络的 Skyline 进行计算，最后通过合并策略再计算整个传感器网络的最终 Skyline 集。

(4)使用近似 Skyline 查询处理策略。对于一些维度较高的数据，可以采用降维方法或者跟维数处理无关的方法，如粒子滤波方法来近似处理，以提高查询处理效率。

参 考 文 献

[1] Akyildiz I F, Su W, Sankarasubramaniam Y, et al. Wireless sensor networks: a survey[J]. Computer networks, 2002, 38(4): 393-422

[2] Yao Y, Gehrke J E. The Cougar approach to in-network query processing in sensor networks[J]. SIGMOD Record, 2002, 31(3): 9-18

[3] Madden S, Franklin M, Hellerstein J. TinyDB: An acqusitional query processing system for sensor networks[J]. ACM Trans. Database Syst., 2005, 30(1): 122-173

[4] Abadi Daniel J, Samuel M, Wolfgang L. REED: Robust, Efficient Filtering and Event Detection in Sensor Networks[C]// Proceedings of the 31st VLDB Conference. Trondheim, Norway, 2005: 769-780

[5] Micro S, Klemens B, Erik B. Processing Continuous Join Queries in Sensor Networks: a Filtering Approach[C]// Proceedings of the ACM Conference on Management of Data. 2010: 267-278

[6] Yao Yu-xia, Tang Xue-yan, Lim Ee-peng. Continuous Monitoring of kNN Queries in Wireless Sensor Networks[C]// Proceedings of International Conference on Mobile Ad-hoc and Sensor Netwoks. 2006: 663-674

[7] Wu Min-ji, Xu Jian-liang, Tang Xue-yan, et al. Top-k Monitoring in Wireless Sensor Networks[J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(7): 962-975

[8] Baljeet M, Nascimento Mario A, Ioanis N. Exact Top-K Queries in Wireless Sensor Networks[J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(10): 1513-1525

[9] Adam S, Rebecca B, Carla E, et al. A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks[C]// Proceedings of the 22nd International Conference on Data Engineering (ICDE'06). 2006:68-78

[10] Alexandru C, Nascimento Mario A, Jorg S. A Framework for Spatio-Temporal Query Processing Over Wireless Sensor Networks[C]// Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN'04). Toronto, Canada, 2004:104-110

[11] Kwon Y, Choi J-H, Chung Y-D, et al. In-Network Processing for Skyline Queries in Sensor Networks[C]// Proceedings of the Institute of Electronics, Information and Communication Engineers (IEICE). 2007:3452-3459

[12] Chen He-kang, Zhou Shui-geng, Guan Ji-hong, et al. Towards Energy-Efficient Skyline Monitoring in Wireless Sensor Networks[C]// 4th European Conference, EWSN 2007. 2007:101-116

[13] Xin J, Wang G, Chen L, et al. Continuously Maintaining Sliding Window Skylines in a Sensor Network[C]// Proceeding of the 12th International Conference on Database Systems for Advanced Applications, DASFAA. 2007:509-521

[14] Borzsony S, Kossmann D, Stocker K, et al. The Skyline operator [C]// Proceedings of the 17th International Conference on Data Engineering. 2001:421-430

[15] Ooi B C, Eng P-K, Tan K-L, et al. Efficient Progressive Skyline Computation[C]// Proceedings of the 27th International Conference on Very Large Data Bases. 2001:301-310

[16] Jan C, Parke G, Jarek G, et al. Skyline with Presorting [C]// Proceeding of the 19th International Conference on Data Engineering. 2003:717-719

[17] Godfrey P, Shipley R, Gryz J. Maximal Vector Computation in Large Data Sets[C]// VLDB. 2005:229-240

[18] Kossmann D, Ramsak F, Rost S. Shooting Stars in the sky: An online algorithm for skyline queries[C]// Proceeding of the 28nd International Conference on Very Large Data Bases. 2002:275-286

[19] Papadias D, Tao Y F, Fu G, et al. Progressive skyline computation in database systems[J]. ACM Transactions on Database Systems, 2005, 30(01):41-82

[20] Papadias D, Tao Y F, Fu G, et al. An Optimal and Progressive Algorithm for Skyline Queries[C]// Proceeding of SIGMOD. 2003:467-478

[21] Lee K C, Zheng B, Lu H, et al. Approaching the Skyline in Z Order[C]// Proceeding of VLDB. 2007:279-290

[22] Lee K C K, Zheng B, Lee W-C, et al. Z-SKY: An Efficient Skyline Query Processing Framework Based on Z-Order[J]. VLDB Journal, 2010, 19(2):333-362

[23] Surajit C, Nilesh D, Raghav K. Robust Cardinality and Cost Estimation for the Skyline Operator[C]// Proceedings of the 22nd International Conference on Data Engineering (ICDE'06). 2006:1-10

[24] Zhang Zhen-jie, Yang Yin, Cai Rui-chu, et al. Kernel-based skyline cardinality estimation[C]// Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD). 2009:509-522

[25] Hose K, Vlachou A. A survey of skyline processing in highly distributed environments[J]. The VLDB Journal, 2012, 21(3): 359-384

[26] Wu Ping, Zhang Cai-jie, Feng Ying, et al. Parallelizing skyline queries for scalable distribution[C]// Proceedings of EDBT. 2006:112-130

[27] Chen Li-jiang, Cui Bin, Lu Hua, et al. iSky: Efficient and Progressive Skyline Computing in a Structured P2P Network[C]// Proceeding of The 28th International Conference on Distributed Computing Systems. 2008:160-167

[28] Wang Shi-yuan, Vu Q H, Ooi B C, et al. Skyframe: a framework for skyline query processing in peer-to-peer systems[J]. The VLDB Journal, 2009, 18:345-362

[29] Tao Yu-fei, Dimitris P. Maintaining Sliding Window Skylines on Data Streams[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(3):377-391

[30] Wolf-Tilo B, Ulrich G, Xin Z J. Efficient distributed skylining for web information systems[J]. EDBT, Lecture Notes in Computer Science, 2004, 2992:256-273

[31] Pei Jian, Jiang Bin, Lin Xue-min, et al. Probabilistic skylines on uncertain data[C]// Proceedings of the 33rd international conference on very large data bases (VLDB'07). 2007:15-26

[32] Lian Xiang, Chen Lei. Reverse skyline search in uncertain databases[J]. ACM Transactions on Database Systems, 2010, 35 (1):49

[33] Khalefa Mohamed E, Mokbel Mohamed F, Levandoski Justin J. Skyline Query Processing for Incomplete Data[C]// Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE'08). IEEE Computer Society. Washington, DC, USA, 2008:556-565

[34] Sharifzadeh M, Shahabi C. The spatial skyline queries[C]// Proceeding of the international conference on Very large data bases (VLDB'06). Seoul, Korea, 2006:751-762

[35] Deng K, Zhou X, Shen H T. Multi-source skyline query processing in road networks[C]// Proceedings of the 23th international conference on data engineering. 2007:796-805

[36] Huang Zhi-yong, Christian S J, Lu Hua, et al. Skyline Queries Against Mobile Lightweight Devices in MANETs[C]// Proceeding of the 22nd International Conference on Data Engineering (ICDE'06). 2006:720-730

[37] Su I-F, Chung Yu-chi, Lee Chiang, et al. Efficient skyline query processing in wireless sensor networks[J]. Journal of Parallel and Distributed Computing, 2010, 70(6):680-698

[38] Liang Wei-fa, Chen Bai-chen, Jeffery X Y. Enery-Efficient Skyline Query Processing and Maintenance in Sensor Networks[C]// Proceedings of the ACM International and Knowledge Management (CIKM'08). 2008:1471-1472

[39] Chen Bai-chen, Liang Wei-fa. Progressive Skyline Query Processing in Wireless Sensor Networks[C]// Proceeding of the Fifth International Conference on Mobile Ad-hoc and Sensor Networks. 2009:17-24

[40] Chen Bai-chen, Liang Wei-fa, Jeffrey X Y. Energy-efficient skyline query optimization in wireless sensor networks[J]. Wireless Networks, 2012, 18(8):985-1004