ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2009/03(01)-0037-14
DOI:10.3778/j.issn.1673-9418.2009.01.004

E-mail:fcst@public2.bta.net.cn
http://www.ceaj.org
Tel:+86-10-51616056

无线传感器网络中滑动窗口轮廓查询算法*

信俊昌+,王国仁,张小艺
东北大学 信息科学与工程学院,沈阳 110004

A Sliding Window Skyline Query Algorithm in Wireless Sensor Networks*

XIN Junchang+, WANG Guoren, ZHANG Xiaoyi
College of Information Science & Engineering, Northeastern University, Shenyang 110004, China
+ Corresponding author: E-mail: xinjunchang@ise.neu.edu.cn

XIN Junchang, WANG Guoren, ZHANG Xiaoyi. A sliding window skyline query algorithm in wireless sensor networks. Journal of Frontiers of Computer Science and Technology, 2009,3(1):37-50.

Abstract: A filter based algorithm (FBA) which continuously maintains sliding window skylines over a wireless sensor network is proposed. Specifically, two approaches using tuple and grid respectively to reduce the amount of data transferred among sensor nodes are first investigated. Since both of them have their own pros and cons, adaptive filtering which chooses the "right" filter according to data distribution is proposed. In addition to FBA, a series of optimization techniques are also discussed to improve the energy efficiency of FBA. Both the synthetic simulation and real data experimental results show that FBA together with the optimization techniques can effectively reduce the communication cost and save the energy on continuously maintaining the sliding window skylines over wireless sensor networks.

Key words: wireless sensor network; skyline query; energy efficiency; filtering; optimization

摘 要:提出了一种基于过滤的算法(filter based algorithm,FBA)来连续地维护传感器网络中的滑动窗口轮廓查询。首先,研究了利用元组过滤器和格过滤器来减少网络中数据传输量的两种方法。由于它们各有利弊,提出了根据数据分布来选择合适的过滤器的自适应过滤法;另外,提出了一系列的优化方法来进一步提高算法的能量有效性。仿真和真实数据的实验结果表明,FBA 及其优化方法能有效地减少连续维护传感器网络中滑

* The National Natural Science Foundation of China under Grant No.60773221,60773219 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA09Z139 (国家高技术研究发展计划(863)); 高等学校科技创新工程重大项目培育资金项目(No.706016).
Received 2008-05, Accepted 2008-07.

动窗口轮廓时的通信代价,进而节约传感器网络的能量。

关键词:无线传感器网络;轮廓查询;能量有效性;过滤;优化

文献标识码:A **中图分类号**:TP311.13

1 引言

无线传感器网络(wireless sensor network,WSN)是由大量的具有感知、计算和通信能力的微型传感器节点以 Ad Hoc 方式构成的无线网络。这些传感器节点协同工作,实时地监测、感知和采集网络部署区域内的各种有用的信息,如温度、湿度、压力和声音等。它为人们随时随地地获取现场信息提供了方便,因而被广泛地应用到了国防和国民经济各个领域,典型的应用如地震监测、生活环境监测、农业监测、矿井环境监测、生物和化学攻击监测以及战场态势感知等[1-2]。传感器网络经常被部署在非常恶劣的环境,有的甚至是人类无法接近的区域(如火山口)。传感器节点是靠电池供电的,而充电或更换电池都是不现实的,因此,传感器网络中数据管理所面临的最大挑战就是能量的有效利用。

近年来,传感器网络中感知数据的查询处理技术得到了广泛研究,而作为多目标决策重要依据的轮廓查询却没有得到相应地重视。给定一个元组的集合 T,轮廓 S 是由 T 中所有不被其他任何元组所支配的元组组成的 T 的子集。对于 T 中的两个元组 t_i 和 t_j,如果 t_i 在所有维不比 t_j 差,并且在至少一维比 t_j 好,那么 t_i 支配 t_j。例如,负载较高且剩余能量较低的节点是传感器网络中的关键点,它们的寿命直接决定着传感器网络的寿命,因此在应用中网络的管理者应该时刻关注这些节点,以便根据情况进行相应地调整来延长网络寿命。这些节点就构成了传感器网络中的轮廓(图 1 中实线上的点)。

传感器网络中,每个传感器节点通常都产生较大的实时流式数据,这些传感器节点就组成了一个分布式实时多数据流系统。由于每个传感器节点只有少量的计算资源,难以处理如此巨大的实时数据流,同时在整个流上计算轮廓也是没有意义的,因此,在传感器网络中,只计算滑动窗口内数据的轮廓。这种查询

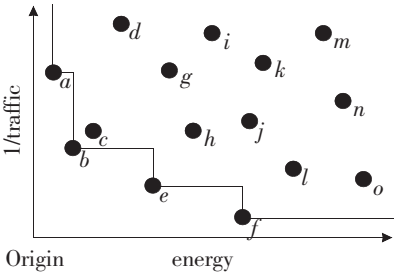


Fig.1 Example of skyline

图 1 轮廓的示例

在传感器网络中有着广泛需求,例如,研究一个森林中的鸟类行为的鸟类学家往往需要知道在什么时间、什么地点某几种鸟类更容易出现,以便能更好地分析各种鸟类之间的关系。

在传统关系数据库领域,轮廓查询得到了广泛地研究,然而由于传感器网络采用无线通信、只有有限的计算资源和能量,因此这些算法都不能直接应用到传感器网络中。同时,由于能量是传感器网络中的稀缺资源而无线通信又是其主要消耗者,因此本文提出了一种基于过滤的滑动窗口轮廓查询算法(filter based algorithm,FBA)旨在减少网络中的数据传输量,进而节约传感器节点的能量。首先,分析了传感器网络中轮廓查询的可分解性,进而论证了采用网内计算的可行性;其次,提出了两种不同的过滤策略来提前终止非轮廓元组的传输,理论证明了过滤的有效性并给出了相应过滤器的维护策略;再次,通过一系列的优化方法来进一步减少节点的数据传输量; 最后,通过大量而详尽的实验验证了 FBA 算法能有效性地减少网络中的数据通信量(数据包的数量),进而节约了传感器节点的能量。

本文第 2 章介绍了轮廓查询的相关工作。接着,在第 3 章给出了问题的描述和相关性质。基于合并的滑动窗口轮廓查询算法(merge based algorithm,MBA)和基于过滤的滑动窗口轮廓查询算法(FBA)将分别

在第 4 章和第 5 章中进行详细介绍。第 6 章提出了一系列的优化方法来提高 FBA 算法的性能。第 7 章通过仿真实验来验证 FBA 的有效性。最后在第 8 章给出结论并对未来的工作加以展望。

2 相关工作

由于传感器网络的巨大应用价值,引起了世界各国的广泛重视。文献[3-5]对传感器网络相关问题的研究进行了详尽说明。作为传感器网络研究的重要分支之一,感知数据查询处理技术近年来得到了越来越多的重视和研究。TinyDB[6-8]和 COUGAR[9]是两种典型的感知数据查询系统,它们实现了一个类 SQL 的接口来完成一般的聚合查询,如 MAX、MIN、AVERAGE、SUM 和 COUNT 等,都采用了网内计算的方法来减少通信的代价。还有一些文献研究了连接运算在传感器网络中的实现,典型的有:文献[10]研究了连接操作符在传感器网络中的动态调整算法;文献[11]研究了静态表与传感器网络内数据的连接;文献[12]和[13]分别研究了一般连接运算和范围连接运算在传感器网络中的实现。

Borzsonyi 等[14]首次提出了轮廓查询的概念并给出了 D&C(divide-and-conquer)方法和嵌套循环方法(blocked nested loop,BNL)来求解轮廓。其中,D&C 方法通过对空间的划分来减少每次处理的数据量,最后通过将各分区上的结果进行合并来得到最终的轮廓。嵌套循环方法通过扫描数据文件并在主存保留轮廓候选的方法来求得轮廓。在 BNL 的基础上,文献[15]将数据按照非单调函数排序来提高算法的性能。文献[16]中提出了位图和索引两种方法来加速轮廓的计算。由于最近邻(nearest neighbor,NN)肯定属于最后的轮廓集合,因此 Kossmann 等[17]提出了一种利用最近邻查询的结果来递归地划分数据空间再逐步求解的方法,这种方法允许用户与程序在求解过程中进行交互,是一种在线算法。在此基础上,文献[18]提出了 BBS 算法(branch-and-bound skyline),利用 R-tree 来进一步提高该算法的性能。

文献[19]中将轮廓查询问题扩展到了万维网,数据被分散地存储在不同的 Web 服务器上,同时提出了基本的分布式轮廓算法(basic distributed skyline,BDS)和改进的分布式轮廓算法(improved distributed skyline algorithm,IDS)来解决这种分布式环境中的轮廓查询问题。其中,BDS 利用一个简单的方法找到一个包含轮廓的子集,将其他的非轮廓元素滤除,而 IDS 则利用了启发式规则来更快地找到包含轮廓的子集。文献[20]利用排序估计的方法进一步提高了 BDS 和 IDS 的性能。与本文研究内容最相似的是文献[21],它主要研究 MANET(mobile Ad-hoc networks)中数据轮廓的计算方法,并利用一种混合的存储模型来提高移动设备上轮廓的计算速度,利用过滤手段来减少 MANET 中的通信代价,不同之处在于它只关注瞬时查询,如快照等,而本文要研究的是连续的轮廓查询。

有些工作主要研究数据流上的轮廓查询,Tao 等[22]提出了在流数据上连续计算并维护轮廓的方法;Lin 等[23]提出了 *n-of-N* 查询的概念并提出了一种剪枝技术来减少需要保存的数据量,同时提出了一种编码技术来减少使用的内存空间,以及一种触发机制来连续地计算最新的 N 个数据中的最新的 n 个数据($n<N$)的轮廓结果。

到目前为止,传感器网络中的轮廓查询还没有研究,已有的轮廓查询算法大多是基于集中式存储环境的,因此不适合计算传感器网络中的轮廓。那些分布式和数据流上的算法虽然和本文研究的问题类似,但是由于没有考虑传感器网络的特殊性,因此也不能直接应用到传感器网络中。

3 问题描述及性质

3.1 问题描述

传感器网络通常由一个基站和大量的传感器节点组成。其中,基站是传感器网络的控制中心,远程用户可以通过卫星或因特网来访问它以获取传感器网络所采集的信息;传感器节点则被大量地部署到监测

环境中来收集用户感兴趣的信息,如湿度、温度和噪音等。如上所述,能量是传感器网络中的稀缺资源且无线通信是其主要消耗者。由于无线通信的能量消耗与距离相关,当发送节点与接收节点的距离小于阈值 d(d 是常数,数值取决于使用环境)时,发送方发送数据的能量损耗与距离的平方成正比,否则与距离的四次方成正比[24]。因此,为了减小通信的传输半径,基站与传感器节点之间以及传感器节点与传感器节点之间都采用多跳(multi-hop)通信。这样在传输数据的时候就需要建立一定的路由结构,以便能节能地完成相应操作。

在传感器网络中,数据是由传感器节点上的传感部件周期性采集而来的,每个采样周期,每个节点采集一个元组。每个采集的元组 t 都有一个属性 $t.arr$ 来表明数据的采集时间。出于节能的考虑,节点采集的数据在非必要的情况下是不会向基站传输的,因此所有的感知数据都分散地存储在各个传感器节点上。严格地说,传感器网络更像是一个分布式的多数据流系统而不是传统的数据库系统。由于数据流是无限的,从理论上讲它的数据量也是无限大的,所以,要在所有的数据都采集后再计算传感器网络中的轮廓是不实际的。因此,本文主要考虑最新采集的数据的轮廓,也即是滑动窗口中数据的轮廓。本文讨论的滑动窗口是基于时间戳的滑动窗口(timestamp-based sliding window),它保留最近 W 时间内到达的数据,W 称为滑动窗口的大小(size of sliding window)。新采集的数据进入窗口后经过 W 的时间间隔后移动到窗口外,而同时新采集的数据则连续不断地进入窗口。设当前的时间为 t_{cur},那么滑动窗口大小为 W 的轮廓查询要考虑所有满足 $t.arr+W>t_{cur}$ 的数据,而那些不满足条件的数据将被丢弃。

3.2 性质

设整个传感器网络中元组的集合为 T,每个节点 i 上的元组的集合为 T_i,维的集合为 D,则元组集合 T 的维度为 $|D|$。t 为 T 中的一个元组,$t.x_d$ 表示 t 在第 d 维上的属性。同时,设传感器网络中节点的个数为 n,$skyline()$ 表示轮廓操作,f 表示支配关系。

根据轮廓查询的定义,可以得到下面的引理。

引理 1 支配关系具有传递性,也就是说,对元组集合 T 中的任意 3 个元组 t_i、t_j 和 t_k,如果 $t_i \succ t_j$,并且 $t_j \succ t_k$,那么 $t_i \succ t_k$。

证明 根据支配关系的定义可以得到,因为 $t_i \succ t_j$ 且 $t_j \succ t_k$,所以对任意的 $d \in D, t_i.x_d \geqslant t_j.x_d, t_j.x_d \geqslant t_k.x_d$,可得 $t_i.x_d \geqslant t_k.x_d$ 并且存在 $d' \in D, t_i.x_{d'} > t_j.x_{d'}$,由传递性可得 $t_i.x_{d'} > t_k.x_{d'}$。因此,可以得出结论 $t_j \succ t_k$。 □

定理 1 传感器网络中的轮廓查询是可分解的,即 $skyline(T)=skyline(\cup skyline(T_i))$。

证明 根据轮廓的定义可知,对于 $skyline(T)$ 中的任何一个元组 t,由于 T 中没有任何元组能支配它,因此将会出现在包含它的 T_i 的轮廓 $skyline(T_i)$ 中,而 $T=\cup T_i$,因此 $skyline(T)\subseteq skyline(\cup skyline(T_i))$。

假设 $skyline(\cup skyline(T_i))$ 中存在一个元组 t 不属于 $skyline(T)$,根据轮廓的定义,必然存在一个元组 $t' \in skyline(T)$,t' 能支配 t。又因为 $skyline(T)\subseteq skyline(\cup skyline(T_i))$,所以 $t' \in skyline(\cup skyline(T_i))$,由于 t 被 t' 支配,所以 $t' \notin skyline(\cup skyline(T_i))$,与假设相矛盾。因此,$skyline(\cup skyline(T_i))\subseteq skyline(T)$。

根据上面的结论可得 $skyline(T)=skyline(\cup skyline(T_i))$。满足文献[25]中对可分解性的定义,因此,传感器网络中的轮廓查询是可分解的。 □

定理 2 对于一个元组 t,不管这个元组是否真实存在,只要有一个真实存在的未过期元组能够支配它,那么所有能被 t 支配的元组都不属于轮廓。

证明 根据引理 1 支配关系的传递性可直接推得。 □

4 基于合并的滑动窗口轮廓查询算法

计算传感器网络中滑动窗口轮廓的最简单方法就是传感器节点时刻将新采集的数据全部传输到基站,然后在基站利用已有的集中式算法来进行求解。由于轮廓是元组集合的子集并且通常只是其中很小

的一部分,因此被传输到基站的数据中,很多都不属于轮廓。这些数据的传输浪费了大量的能量。

根据定理1,传感器网络中的轮廓查询具有可分解性,所以可以采用与TAG(tiny aggregation)[3-4]类似的办法将轮廓计算的过程下移到传感器节点中执行。与TAG相同,算法采用以基站为根节点的树形结构作为系统的路由结构。路由树的建立过程如下:首先,基站广播一个包含它自身编号和层次的消息。网络中任何收到该消息的未设置层次的节点都将自己的层次设置为消息中的层次加1,并且将发送者设为自己的父节点,然后用自己的编号和层次替换消息中的相应信息并将其广播给自己的邻居。这样的过程将一步一步地进行下去,直到所有的节点都被设置了所属层次和相应的父节点为止。这个过程将会周期性地由基站发起,网络的路由结构也会周期性地重建,因此,这种建立方式能很快地适应网络拓扑结构的变化,如节点的加入、删除和移动等。

建立起路由树后,便开始轮廓初始值的计算,过程如下:首先,叶节点计算自身窗口内数据的轮廓并将结果转发到父节点。接着,中间节点先将接收到的轮廓与自身的轮廓结果集合合并,然后利用定理1中的合并函数得出新的局部轮廓,即以该节点为根的子树的窗口内数据的轮廓,再将新的结果转发给自己的父节点。这样逐级计算,最终基站将得到整个网络中的轮廓结果。由于在中间只传输局部的轮廓结果而不是传输整个数据集,因此极大地降低了传输的数据量。

随着时间的变化,新数据不断地被传感设备采集,它们会进入到滑动窗口中,而那些不满足 $t.arr+W>t_{cur}$ 的元组将过期,不再参与轮廓运算,此时的轮廓结果将发生变化。求解新轮廓的最简单的办法就是按照上面的过程重新计算轮廓,这样当进行连续轮廓查询时,上面的过程将重复地执行,显然,这样的策略是不可取的。因为在连续时刻的窗口之间有相当大的一部分数据是相同的,所以两个时刻的轮廓之间也会有很多数据是相同的,所以,在连续计算轮廓结果的过程中,没有必要将已经传输的数据再重新传输一遍,只要每个节点都保存那些在未来可能成为轮廓结果的数据[22],而节点之间只传输那些从来没有被传输过的数据即可。这样,网络中的数据传输量将被进一步降低。

5 基于过滤的滑动窗口轮廓查询算法

利用网内计算可以减少网络中的数据传输量,但仍然有很大一部分不属于全局轮廓的元组将被传输。如果在每个节点设置一个过滤器,提前判断出那些不属于全局轮廓的元组,并将它们过滤掉,那么网络中的数据传输量将被极大地降低。按照过滤器的选择,将过滤方式分为元组过滤、格过滤和自适应过滤。下面分别加以介绍。

5.1 元组过滤

如果一个元组 t_i 属于局部轮廓,而不属于全局轮廓,那么数据集 T 中一定存在一个元组 t_j 能支配 t_i。如果将元组 t_j 发送给 t_i 所在的传感器节点,那么 t_i 将因为确定不属于轮廓而不需要被传输。如果能找到可以支配最多元组的元组并将它通知所有节点,那么传感器网络中的数据传输量将被极大地降低。

设 $x=(x_1,x_2,\cdots,x_{|D|})$ 与 $y=(y_1,y_2,\cdots,y_{|D|})$ 是 $|D|$ 维空间中的元组集合 T 中的两个元组,如果元组的概率密度函数为 $p(x)=p(x_1,x_2,\cdots,x_{|D|})$,那么一个元组能被 y 所支配的概率为

$$r(y)=\iint_R p(x_1,x_2,\cdots,x_{|D|})\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_{|D|} \quad (1)$$

其中,$R=\{x|x_1\geqslant y_1,x_2\geqslant y_2,\cdots,x_{|D|}\geqslant y_{|D|}\}$。

如果元组集合的总量为 $|T|$,那么能被元组 y 支配的元组总量为 $count(y)=r(y)\times|T|$。由于 $|T|$ 是常量,因此在计算时使用公式(1)即可。显然,根据公式(1)得到最大值的元组就是能支配最多元组的元组。在很多情况下,$p(x)$ 是未知的,但是 $p(x)$ 的近似分布却可以通过随机采样或直方图[25-26]的方法获得。由于节点的计算能力有限,因此在这里选择多项式来表达 $p(x)$,在计算的初始阶段只需要将积分后的多项式的参数传递给传感器节点即可。这样并不需要传输很多的数

据,只需要消耗少量的能量,同时也避免了节点进行大规模的积分运算。

利用这种方式来计算轮廓的处理过程如下:

(1)每个传感器节点根据公式(1)计算每个元组的 r 值,并找到最大的元组,作为候选;

(2)利用网内聚合的方法得到具有最大 r 值的元组,将其设置为过滤器;

(3)将过滤器广播到整个网络中,节点将其保存在各自的内存中;

(4)利用过滤器过滤那些能被其支配的局部轮廓元组;

(5)利用与合并算法相同的方法来计算全局的轮廓。

在轮廓查询的连续执行过程中,过滤器同样可以起到减少数据传输量的作用。为了保证结果的正确性,有效而正确地维护过滤器成了滑动窗口轮廓维护过程中的重要问题。

由定理 2 可知,元组过滤器过期的时候并不一定需要马上替换,这是因为,如果有一个元组能支配它的话,那么用它来过滤掉的那些元组肯定不属于轮廓,因此并不影响结果的正确性。但是如果一个有很小的 r 值的元组在某一时刻被选做了过滤器,那么它被新产生的元组所支配的概率将会很大,就很可能长时间地不会被替换。这种低下的过滤能力将导致许多本该被过滤掉的元组在网络中的传输,极大地浪费了能量。然而,虽然替换过滤器会带来很好的过滤效果,但是这种过滤效果的获得是以过滤器替换时的广播为代价的。在此,利用一个代价-获益模型来进行过滤器替换与否的判断。

$$cost(f)=broadcast(f)$$

$$benefit(f)=(r(f)-r(f_{old}))\times n\times \bar{t}_f\times c_0$$

其中,c_0 为每个元组的传输代价,$\bar{t}_f$ 是元组过滤器的平均有效时间,n 为网络中节点的数量,则在时间 $\bar{t}_f$ 内一共产生的数据总量为 $n\times\bar{t}_f$。过滤器 f 的代价就是它的广播代价,它的获益就是在有效期内与原过滤器 f_{old} 相比能多过滤的元组的传输代价。有了评价模型,过滤器的替换策略可以总结如下:

(1)当前元组过滤器过期,并且它不被新的轮廓元组所支配;

(2)有一个新的元组的获益超过了代价。

只要满足一个条件,就进行过滤器的替换,一个新的元组将被选做过滤器并被广播到整个传感器网络中。条件(1)满足时,当前过滤器已经无效,不能保证它所支配的元组一定不属于轮廓,所以不能继续使用;条件(2)则说明了当前的过滤器过滤效果不如新的过滤器,所以需要替换。这样,过滤器就能在执行过程中被有效地维护,从而提高算法的执行效率,节能而又准确地维护了滑动窗口中的轮廓结果。

5.2 格过滤

容易看出,对独立分布或相关分布的数据来讲,元组过滤器可以过滤掉大部分数据,而对反相关分布的数据来讲,由于一个元组所能支配的数据只占整个数据集的一个小部分,因此过滤效果将会变得很差。为了解决这个问题,提出了一种格过滤的方法。

格过滤方法将数据空间分割成规则的单元格,每个维被分成了 s 片,那么一共有 $s^{|D|}$ 个单元格。每个单元格 c 都有一个 $c.sta$ 属性来记录该单元格的状态。当有元组落到单元格里的时候就将 $c.sta$ 设置为 1,反之则设为 0(如图 2(a)),称之为原始格。还有另一种选择就是将格进行预处理,如果单元格的元组确定不属于轮廓,那么将 $c.sta$ 设置为 0,其余的单元格状态 $c.sta$ 都将被设置为 1(如图 2(b)),称之为预处理的格。在判定一个元组是否被格所支配时,原始格需要考察所有单元格的状态,而预处理的格只需要考察一个单元格的状态。同时,这种预处理并没有增加格在聚合计算过程中的代价,只需将原来的“或”操作变为“与”操作即可。由于在判断元组是否被格所支配时原始格的代价较高,而进行聚合运算时两者的代价相同,因此,可以得出结论:预处理的格要优于原始格。所以,算法中倾向于使用预处理的格,在后面的论述中提到的格都是预处理的格。

1	1	1	1	1
1	1	1	1	0
1	0	1	1	1
0	1	0	1	1
0	0	1	1	1

(a)Original grid

1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	1	0	0	0
1	1	1	1	1

(b)Pre-processing grid

(a)初始格 (b)预处理格

Fig.2 Example of grid

图 2 格的例子

利用格过滤来计算轮廓的过程与元组过滤基本一致,唯一的不同只是使用格来过滤数据,而不是使用原来的元组进行过滤。在过滤时只需要检查元组所在单元格的状态 $c.sta$ 就可以了。当所在单元格的状态 $c.sta$=0 时,元组将被过滤,从而不被传输,只有所在单元格的状态 $c.sta$=1 时,元组才被传输。

同样地,在轮廓查询的连续执行过程中,格的维护也非常重要。在讨论具体的维护策略之前,先给出格间支配关系的定义以及一些相应的性质。

定义 1 对两个格 g_1 和 g_2。如果 g_1 中的某个单元格 c 的状态 $c.sta$ 为 0,g_2 中相应的单元格状态也为 0,那么 g_2 支配 g_1。

定理 3 对两个格 g_1 和 g_2,如果 g_2 支配 g_1,那么能被 g_1 过滤的元组一定也能被 g_2 过滤。

证明 根据格的过滤原则和定义 1 可直接推得。□

推论 1 对于一个格 g,不管它是否是根据真实数据计算来的,只要根据未过期数据计算出来的格能支配它,那么所有能被 g 支配的元组都不属于轮廓。

根据推论 1 可知,当格过期的时候也不一定要马上进行替换,而当一个更好的格出现的时候需要进行相应的判断来决定是否替换。相同的代价-获益模型同样适用于格,判断标准与元组类似,只是具体的函数形式发生了变化。设 T 中元组的概率密度函数为 $p(x)$,那么元组能被格 g 支配的概率为

$$r(g)=\iint_{G} p(x_1,x_2,\cdots,x_{|D|})\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_{|D|} \qquad (2)$$

其中,$G=\{x|x\in c_i \wedge c_i.sta=0\}$。

(1)当前格过滤器过期,并且根据新轮廓计算出的新格不支配当前格;

(2)一个新格的获益超过了代价。

只要有一个条件满足,就将进行过滤器的替换,一个新格将被选做过滤器并被广播到整个传感器网络中。与元组过滤器的替换原则类似,条件(1)满足时,说明当前格过滤器支配的元组不能保证被新格支配,也就不能保证被当前轮廓支配,因此过滤能力失效;条件(2)则说明了用新格替换原来的过滤器可以获得更好的过滤效果,因此可行。这样,格过滤器就能在连续执行过程中被有效地维护,从而提高算法的执行效率,节能而准确地维护滑动窗口中的轮廓结果。

5.3 自适应过滤

由于元组过滤与格过滤各有利弊,当数据为独立分布或相关分布时,元组过滤的效果较好;而当数据为反相关分布时,格过滤效果较好。一个可行的办法就是利用一定的策略来选择“合适”的过滤器进行运算,以便发挥它们各自的长处而回避短处。这种方法称为自适应过滤。

首先,利用采样或直方图[24-26]的方法来得到数据的大致分布,然后根据具体的分布来确定使用的过滤策略。如果数据近似于独立分布或者相关分布,将使用元组过滤;如果数据近似于反相关分布则使用格过滤。当然,在运算过程中数据的分布也可能发生变化,为此可以在基站同时计算两种过滤器的获益和代价,通过代价-获益来对比判断下一个时刻应该使用什么样的过滤策略,始终选择总体过滤效果最好的过滤器。

6 优化策略

6.1 偷听技术

在前面的讨论中,有一个始终没有注意的事实:传感器网络中节点是通过共享信道来进行无线通信的。也就是说,当两个节点通信的时候,在通信半径内

的其他节点也能收到它们发送的消息。有效地利用这些“不经意”监听到的数据将会使系统的性能有所提高[7]。

在通信过程中,一个节点发送的数据可以同时被很多上级节点接收,但只有它的父节点才真正地使用这些数据来参与运算,因此可以在其他收到数据的节点上合理地利用这些数据,以达到减少系统通信量的目的。

在偷听方式中,中间节点不仅保存它的子节点发送来的数据,同时偷听信道上其他节点发送的消息,并将偷听到的数据放入缓存。在计算轮廓的时候像利用元组过滤一样利用这些偷听到的数据来过滤数据,偷听到的数据只参与过滤,并不会进入到最后的局部轮廓结果中。一些本该被传送的局部轮廓元组由于被偷听到的元组所支配而不必传输到父节点,这将进一步减少计算过程中传感器网络内的数据传输量。

6.2 修边技术

对格过滤器而言,在格的聚合过程中,由于格的右上边缘不会支配任何的单元格,因此没必要传输;而在格作为过滤器被广播到传感器网络中时,由于其左下边缘的数据恒为1,因此也没有传输的必要。所以,在格的传输过程中可以根据不同的情况而剪掉不同的边缘来减小格的传输大小,达到减少通信量的目的。

6.3 压缩技术

传感器网络中数据的传输量是关键问题,因此在传输时必然会采取一定的压缩技术来减少数据大小。而对于格来讲,又有着其特殊性。它由一系列有规律的0和1组成,因此可以跟文献[27]一样修改文献[28]中介绍的方法来压缩数据。

由于任何一个单元格的状态 $c.sta$=0 的概率是可求的,因此可以将格过滤器中的单元格按概率由大到小的顺序来排列,这样数据连续为1和连续为0的数据出现的概率都很高,然后用文献[28]中的编码方式来进行数据的压缩。这种方式可以使压缩后的数据仅占原数据大小的30%左右,这将进一步地减少传感器网络中的通信代价。

7 实验结果及分析

实验主要比较基于合并的滑动窗口轮廓查询算法MBA与基于过滤的滑动窗口轮廓查询算法FBA的性能。独立分布、相关分布及反相关分布是验证轮廓查询算法的标准[14],而独立分布与相关分布又很接近,因此,在实验中将主要考察在独立和反相关两种分布下两种算法(MBA和FBA)的性能,同时为了说明FBA中自适应过滤的效果,加入了元组过滤(FBA_T)和格过滤(FBA_G)两种算法一起参与比较。加入了优化技术(optimized production technology,OPT)的FBA也参与了评价。主要的性能指标就是传感器网络中的总通信代价。

随机地在面积为 $\sqrt{n}\times\sqrt{n}$ 个单位的区域内产生 n 个传感器节点,节点之间的通信半径设为 $2\sqrt{2}$ 个单位长度。实验的数据(包括独立分布和反相关分布)均匀地分布在 n 个传感器节点上。每个时刻,每个节点采集一个新数据,因此,整个传感器网络中每个时刻将产生 n 个新元组。实验主要考察算法在传感器节点数量、滑动窗口大小和感知数据维度变化时的性能。表1列出了所要考察的主要参数及其范围和默认值。在每次实验中,只变化一种参数,其余参数为默认值。所有的实验都在一台具有2.8 GHz CPU、512 M内存和80 G硬盘的PC机上完成。

Table 1 Parameters in simulation

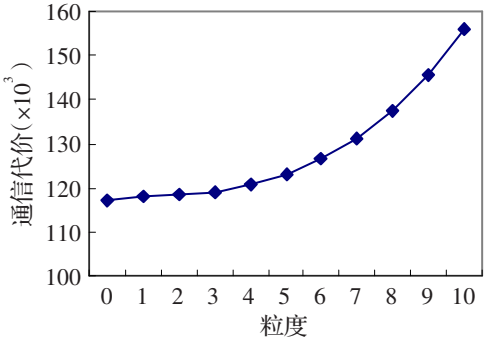
表1 仿真参数

参数	默认值	范围
传感器节点数量	1 000	600,700,800,900,1 000
滑动窗口大小	300	100,200,300,400,500
感知数据维度	3	2,3,4

在比较两种算法在初始轮廓计算过程中的性能之前,先来考察格划分粒度的对格过滤的影响。图3

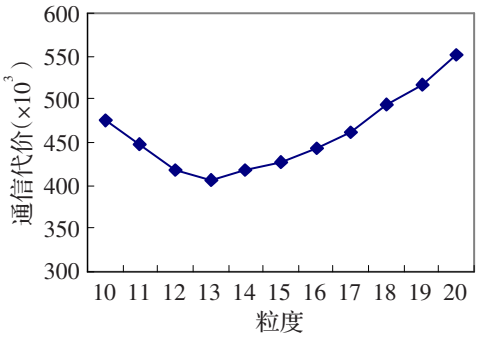
显示了不同格粒度下的通信代价。在独立分布下,代价最低的格粒度为 0;在反相关分布下,代价最低的格粒度为 13。这是因为虽然格会过滤掉一些数据,但同时它的计算和广播也是要花费代价的。在独立分布下,随着格粒度的增长,格的广播代价的增速超过了过滤带来的效果,所以使用格过滤的效果算法不如没有格过滤的算法的效果好;而对反相关分布,代价和获益找到了一个很好的切合点,在格粒度为 13 时达到了获益与代价之差最大,所以效果最好。因此在初始轮廓计算过程实验中,独立分布下格的粒度为 0,即退化为 MBA,而在反相关分布下格粒度为 13。

图 4 和图 5 分别给出了独立和反相关两种分布下,节点数量及窗口大小的变化对算法性能的影响。可以看到,无论是节点数目的增加还是窗口大小的增加,都导致了 MBA 算法通信代价的增加,而 FBA 以及 FBA+OPT 虽然在反相关分布下也遵循着同样的规律,但是在独立分布下却略有波动。这是因为,无论是节点数目增加还是窗口大小增加都使得参与计算的元组数量增加,导致了中间轮廓结果大小的增加,所以 MBA 的通信代价也随之而增加。而对 FBA 以及 FBA+OPT 来讲,元组数目的变化也导致了过滤器的变化,过滤器性能的好坏直接影响了通信的代价。对独立分布而言,使用的是元组过滤,数据的微小差异会影响过滤器的选择,所以曲线会有一些波动;而对反相关分布,由于使用的是格过滤,因此对数据的细微变化不敏感,所以比较稳定。同时可以看出,无论节



(a)Independent

(a)独立分布

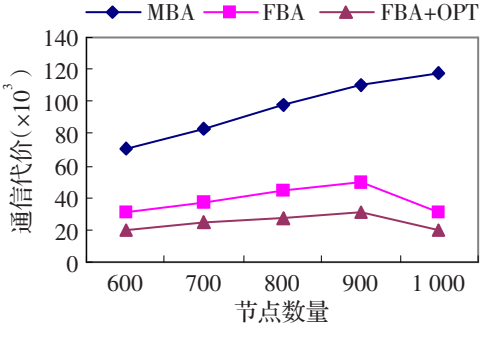


(b)Anti-correlated

(b)反相关分布

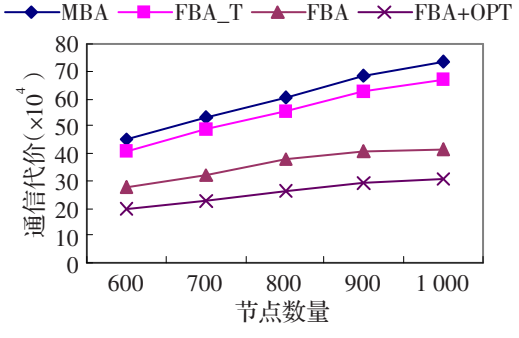
Fig.3 Effect of granularity

图 3 格划分粒度的影响



(a)Independent

(a)独立分布



(b)Anti-correlated

(b)反相关分布

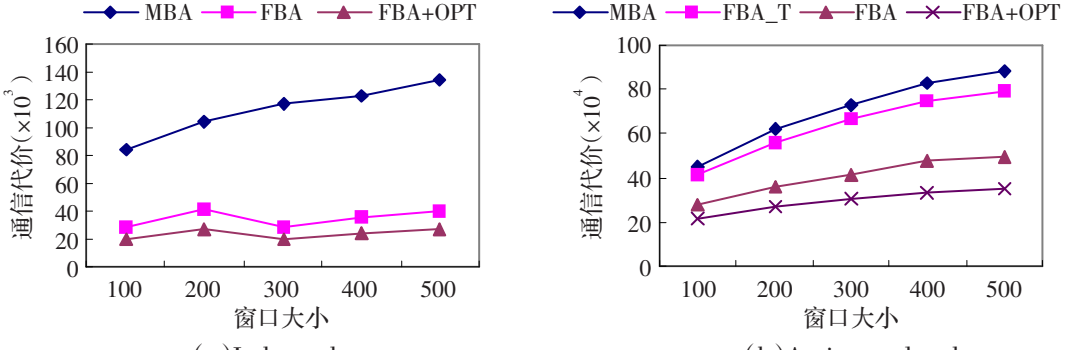
Fig.4 Communication cost vs. number of node

图 4 通信代价 vs.节点数量

点数目和窗口大小如何变化,FBA 都始终优于 MBA,这验证了过滤算法的有效性,同时 FBA 也优于没有使用自适应过滤的方法,这又说明了自适应过滤方法的有效性。而加入了优化技术的 FBA+OPT 算法比 FBA 在性能上又有了进一步的提高,同时在反相关下的提高幅度要高于独立分布下的提高幅度。这是因为,独立分布时 FBA 使用的是元组过滤,因此只能使用一种优化策略,即偷听技术;而反相关分布时 FBA 使用的是格过滤,除了可以使用偷听技术以外,还可以使用另两种优化技术,即修边和压缩技术,因此反相关分布下优化的效果更好。

图 6 给出了独立和反相关两种分布下,维度变化对算法性能的影响。可以看到,随着维度的增加通信代价大幅度增加,这是因为维度的增加使得元组大小增加,导致单位元组的通信代价增加;同时,维度的增加使得两个元组之间互相不支配的概率增加,所以轮廓结果的大小也相应增加。由于单位元组的传输代价和需要传输的元组数目同时增加,因此通信代价大幅度地增加。同时可以看出,无论维度如何变化,FBA 都始终优于 MBA,FBA 也优于没有使用自适应过滤的方法,这又进一步验证了过滤以及自适应过滤方法的有效性。同样地,加入了优化技术的 FBA+OPT 与 FBA 算法相比,在性能上又有所提高,同时还遵循着在反相关分布下提高幅度高于独立分布的规律。

接着,比较轮廓维护阶段算法的性能。图 7 给出了轮廓维护过程中格划分粒度对算法性能的影响。可

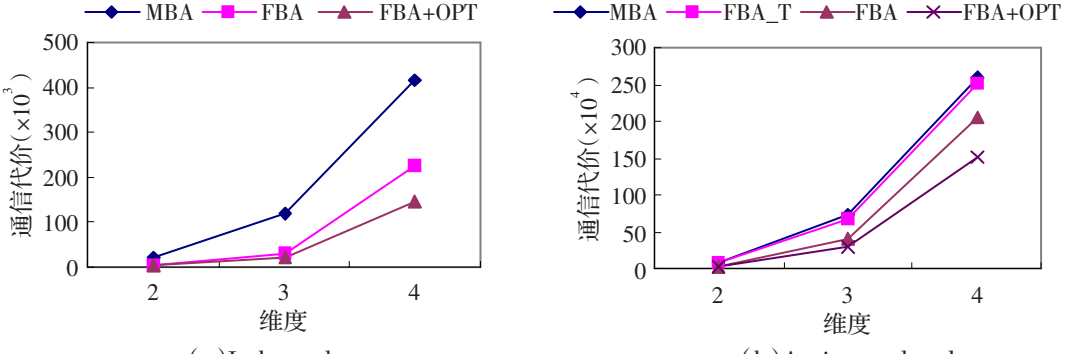


(a)Independent (b)Anti-correlated

(a)独立分布 (b)反相关分布

Fig.5 Communication cost vs. window size

图 5 通信代价 vs.窗口大小



(a)Independent (b)Anti-correlated

(a)独立分布 (b)反相关分布

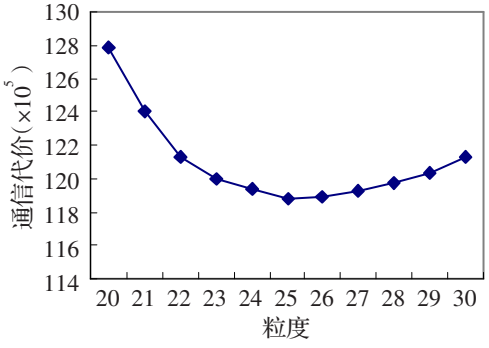
Fig.6 Communication cost vs. dimension

图 6 通信代价 vs.维度

以看出,最优的格粒度发生了变化。在独立分布下最好的格粒度是 25，而在反相关分布下最好的格粒度是 15。这是因为在轮廓维护过程中,格有了较长的时效,能过滤更多的元组,同时计算和广播的代价被各个时间段所分担，所以与初始轮廓计算过程相比,过滤器的单位时间代价变小了,所能承受的过滤器代价就增加了，因此相应的最优粒度都有一定程度的增加。在新的格粒度下,代价和获益又重新找到了切合点。

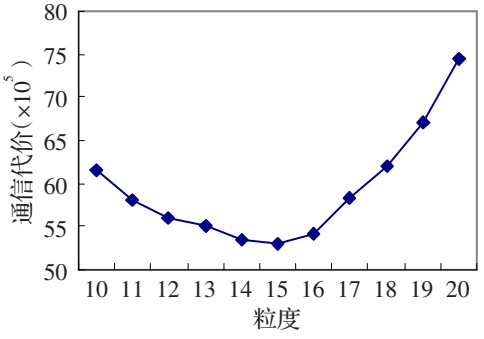
图 8 给出了独立和反相关两种分布下,算法在轮廓维护过程中通信代价随时间变化的规律。可以看出,随着时间的变化,总通信代价平稳地增长。这是因为,在轮廓维护过程,不断有新的数据加入到窗口中,也就会有许多新的轮廓数据产生,因此代价会逐步增加。又因为所有的数据将最多只被传输一次,所以代价增加非常平稳,并不会出现急剧的增加。同时可以看出，在连续运行过程中,FBA 都始终优于 MBA,也优于没有使用自适应过滤的方法。既验证了过滤算法的有效性也验证了自适应过滤策略的有效性,同时也验证了在轮廓维护过程中过滤器维护策略的有效性。

下面研究优化策略在轮廓维护过程中对算法的影响。图 9 表明,使用了优化策略以后,FBA 的性能被提高了。同时在独立分布下,提高的幅度并不大,而在反相关分布下,性能的提高则非常明显。这是因为在独立分布下,FBA 使用的是元组过滤,因此只能使用偷听技术来对算法进行优化;而在反相关分布下使用的是格过滤,FBA 除了可以利用偷听技术来优化



(a)Independent

(a)独立分布

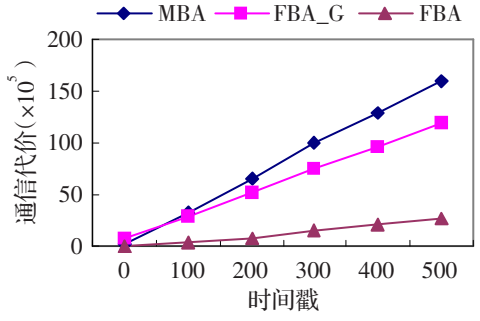


(b)Anti-correlated

(b)反相关分布

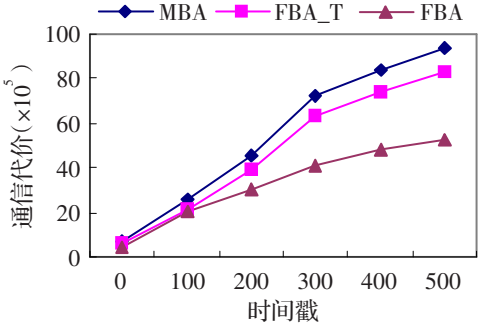
Fig.7 Effect of granularity in maintenance

图 7 维护过程中划分的影响



(a)Independent

(a)独立分布

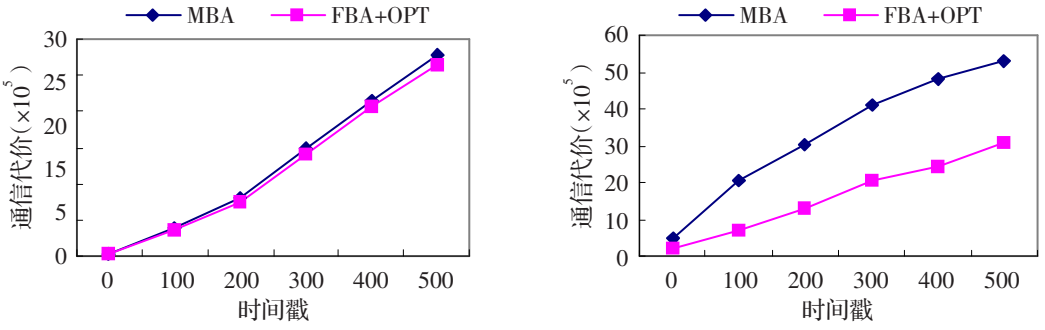


(b)Anti-correlated

(b)反相关分布

Fig.8 Communication cost vs. time

图 8 通信代价 vs.时间



(a)Independent (b)Anti-correlated

(a)独立分布 (b)反相关分布

Fig.9 Optimizations in maintenance

图 9 维护过程中的优化

外,另外两种技术,修边和压缩也可以使用。因此在反相关分布下优化的效果要好于独立分布。

最后,用实际数据来进一步验证算法的有效性。实验中选择了热带海洋与大气项目(tropical atmosphere ocean,TAO)中的动力高度、等温面深、相对湿度共三维数据来进行测试。图 10 给出了在节点数量为 59,窗口大小为 30 时初始轮廓计算及轮廓维护过程(连续滑动次数为 30 次)中,算法的性能表现。可以看出,FBA 及其优化策略始终要优于基本的 MBA 算法。同时由于数据近似地满足独立分布,因此 FBA 使用了元组过滤,而初始轮廓计算时,最优格大小为 0,所以同 MBA。而连续维护阶段,格过滤有效,但是效果不如元组过滤。

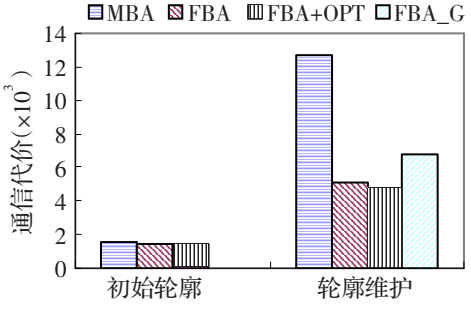


Fig.10 Performance on real dataset

图 10 真实数据集的性能

通过一系列仿真数据和真实数据的实验,FBA 在各种条件下的性能均优于 MBA。因此可以得出结论,FBA 可以有效地节约传感器网络的窗口轮廓查询中初始轮廓计算及其维护过程中的能量消耗,进而延长网络的使用寿命。

8 结论及今后的工作

能量是传感器网络中的稀缺资源,无线通信又是能量的主要消耗者,因此如何减少网络中的数据通信量是传感器网络中感知数据管理技术要解决的主要问题。本文提出了一种基于过滤的滑动窗口轮廓查询算法(FBA),算法首先提出了元组和格两种过滤器,进而提出了元组过滤、格过滤和自适应过滤三种过滤方式并从理论上给出了相应的证明。接着,给出了一系列的优化方法来提高过滤效率或减少数据的传输量,进而提高算法的效率。最后,通过大量的实验证明,FBA 算法极大地减少了传感器网络中的数据传输量,有效地节约了能量,进而有效地延长了传感器网络的使用寿命。

由于有些应用只需要了解轮廓的大致分布即可,并不需要真实的轮廓,因此如何在轮廓的精度和通信代价之间进行折中,使之既能满足用户的需求,又能避免能量的浪费,这将是下一步研究工作的重点。

References:

[1] Cardell-Oliver R, Kranz M, Smettem K, et al. A reactive soil moisture sensor network: Design and field evaluation[J]. International Journal of Distributed Sensor Networks, 2005,1(2):

149-162.

[2] Xue W W, Luo Q, Chen L, et al. Contour map matching for event detection in sensor networks[C]//Proc of the ACM SIGMOD International Conference on Management of Data, June 27-29, 2006:145-156.

[3] Ren Fengyuan, Huang Haining, Lin Chuang. Wireless sensor networks[J]. Journal of Software, 2003,14(2):1148-1157.

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

[5] Li Jianzhong, Li Jinbao, Shi Shengfei. Concepts, issues and advance of sensor networks and data management of sensor networks[J]. Journal of Software, 2003,14(10):1717-1727.

[6] Madden S, Franklin M J, Hellerstein J M, et al. The design of an acquisitional query processor for sensor networks[C]// Proc of the 2003 ACM SIGMOD International Conference on Management of Data, June 9-12, 2003:491-502.

[7] Madden S, Franklin M J, Hellerstein J M, et al. TAG: A tiny aggregation service for Ad-Hoc sensor networks[C]//Proc of 5th Symposium on Operating System Design and Implementation, December 9-11, 2002:131-146.

[8] Madden S, Szewczyk R, Franklin M J, et al. Supporting aggregate queries over Ad-Hoc wireless sensor networks[C]// Proc of 4th IEEE Workshop on Mobile Computing Systems and Applications, June 20-21, 2002:49-58.

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

[10] Bonfils B J, Bonnet P. Adaptive and decentralized operator placement for in-network query processing[C]//Proc of the Second International Symposium on Information Processing in Sensor Networks, April 22-23, 2003:47-62.

[11] Abadi D J, Madden S, Lindner W. REED: Robust, efficient filtering and event detection in sensor networks[C]//Proc of the 31st International Conference on Very Large Data Bases, August 30-September 2, 2005:769-780.

[12] Chowdhary V, Gupta H. Communication-efficient implementation of join in sensor networks[C]//Proc of the 10th International Conference Database Systems for Advanced Applications, April 17-20, 2005:447-460.

[13] Pandit A, Gupta H. Communication-efficient implementation of range-joins in sensor networks[C]//Proc of 11th International Conference Database Systems for Advanced Applications, April 12-15, 2006:859-869.

[14] Borzsonyi S, Stocker D, Kossmann K. The skyline operator[C]// Proc of the 17th International Conference on Data Engineering, April 2-6, 2001:421-430.

[15] Chomicki J, Godfrey P, GRya J, et al. Skyline with presorting[C]//Proc of the 19th International Conference on Data Engineering, March 5-8, 2003:717-719.

[16] Tan K L, Eng P K, Ooi B C. Efficient progressive skyline computation[C]//Proc of 27th International Conference on Very Large Data Bases, September 11-14, 2001:301-310.

[17] Kossmann D, Ramsak F, Rost S. Shooting stars in the sky: An online algorithm for skyline queries[C]//Proc of 28th International Conference on Very Large Data Bases, August 20-23, 2002:275-286.

[18] Papadias D, Tao Y F, Fu G, et al. An optimal and progressive algorithm for skyline queries[C]//Proc of the 2003 ACM SIGMOD International Conference on Management of Data, June 9-12, 2003:467-478.

[19] Balke W T, Güntzer U, Zheng J X. Efficient distributed skylining for Web information systems[C]//Proc of the 9th International Conference on Extending Database Technology, March 14-18, 2004:256-273.

[20] Lo E, Yip K Y, Lin K I, et al. Progressive skylining over Web-accessible database[J]. Journal of Data and Knowledge Engineering, 2006,57(2):122-147.

[21] Huang Z Y, Jensen C S, Lu H, et al. Skyline queries against mobile lightweight devices in MANETs[C]//Proc of the 22nd International Conference on Data Engineering, April 3-8, 2006:66.

[22] Tao Y F, Papadias D. Maintaining sliding window skylines on data streams[J]. IEEE Transactions on Knowledge and Data Engineering, 2006,18(3):377-391.

[23] Lin X M, Yuan Y D, Wang W, et al. Stabbing the sky: Efficient skyline computation over sliding windows[C]//Proc of the 21st International Conference on Data Engineering, April 5-8, 2005:502-513.

[24] Heinzelman W B, Chandrakasan A P, Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks[J]. IEEE Transactions on Wireless Communications, 2002,1(4):660-670.

[25] Chaudhuri S, Dalvi N N, Kaushik R. Robust cardinality and cost estimation for skyline operator[C]//Proc of the 20th International Conference on Data Engineering, April 3-8, 2006:64.

[26] Silverman B W. Density estimation for statistics and data analysis[M]. London: CRC Press, 1986.

[27] Considine J, Li F F, Kollios G, et al. Approximate aggregation techniques for sensor databases[C]//Proc of the 20th International Conference on Data Engineering, March 30-April 2, 2004:449-460.

[28] Palmer C R, Gibbons P B, Faloutsos C. ANF: A fast and scalable tool for data mining in massive graphs[C]//Proc of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002:81-90.

附中文参考文献:

[3] 任丰原,黄海宁,林闯.无线传感器网络[J].软件学报,2003,14(2):1148-1157.

[5] 李建中,李金宝,石胜飞.传感器网络及其数据管理的概念、问题与进展[J].软件学报,2003,14(10):1717-1727.

XIN Junchang was born in 1977. He received the M.S. degrees in Computer Science and Technology from the Northeastern University in 2005. He is currently a Ph.D. candidate and an assistant at Northeastern University. His research interest includes query processing in sensor network.

信俊昌(1977-),男,辽宁辽阳人,博士研究生,2005 年在东北大学获硕士学位,现任东北大学助教,主要研究领域为传感器网络中查询技术。

WANG Guoren was born in 1966. He received the Ph.D. degree from the Northeastern University in 1996. He is a professor and doctoral supervisor at Northeastern University, and the senior member of CCF. He is also the director of Computer Center of Northeastern University, the director of Institute of Computer Systems. His research interests include XML data management, query processing and optimization, bioinformatics, high-dimensional indexing, parallel database systems and P2P data management.

王国仁(1966-),男,湖北崇阳人,1996 年在东北大学获博士学位,现任东北大学计算中心主任、计算机系统研究所所长、教授、博士生导师,计算机学会高级会员,主要研究领域为 XML 数据管理,查询处理与优化,生物信息学,高维数据索引,并行数据库系统以 P2P 数据管理,发表论文 100 余篇,主持或承担过多项国家及省部级项目。

ZHANG Xiaoyi was born in 1984. She is a M.S. candidate at Northeastern University. Her research interest includes query processing in sensor network.

张小艺(1984-),女,河南洛阳人,东北大学硕士研究生,主要研究领域为传感器网络中查询技术。