无线传感器网络中基于聚簇结构的Skyline查询方法

李 青 肖迎元 王晓晔 李玉坤
(天津理工大学计算机与通信工程学院 天津 300384)
(天津市智能计算及软件新技术重点实验室 天津 300384)

摘 要 现有的基于单服务器的Skyline查询算法已经不能很好地应用于无线传感器网络这类分布式多跳自组织网络中。基于聚簇结构的Skyline查询算法就是针对这类特定的网络结构而提出的。该算法采用基于聚簇的路由结构,为了减少Skyline查询处理过程中传感器节点的通信开销,挑选具有最大支配力的数据元组作为全局过滤元组来过滤不满足Skyline条件的数据。同时,在Skyline查询处理过程中引入滑动窗口机制,该机制也能有效地降低通信开销。大量的仿真实验结果显示,所提Skyline查询算法在确保能耗的基础上仍然具有很好的性能。

关键词 Skyline查询,聚簇结构,滑动窗口
中图法分类号 TP311.13 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.10.033

Clustering Architecture-based Skyline Query Processing in Wireless Sensor Networks

LI Qing XIAO Ying-yuan WANG Xiao-ye LI Yu-kun
(School of Computer and Communication Engineering, Tianjin University of Technology, Tianjin 300384, China)
(Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin 300384, China)

Abstract Obviously, the existing Skyline query algorithm based on single server can not be applied to the kind of distributed multi-hop ad hoc networks, such as wireless sensor networks. In this paper, we proposed a clustering based Skyline query method for the specific networks. Clustering architecture-based routing is adopted, which selects the maximum rule power data tuple as global filter to filter the data that do not satisfy the Skyline condition, in order to reduce the communication overhead of sensor nodes in the Skyline query processing. Meanwhile, the sliding window mechanism is introduced into the Skyline query processing, and the mechanism can also effectively reduce the communication overhead. A large number of experimental results show that the proposed Skyline query algorithm has good performance of energy consumption.

Keywords Skyline query processing, Clustering architecture, Sliding window

1 引言

最具代表性的无线传感器网络的数据管理系统是美国加州大学伯克利分校的TinyDB系统和Cornell大学的Cougar系统[1]。但是这些系统只考虑了最基本的查询,并未涉及针对多维数据的复杂查询,比如Skyline查询。Skyline查询是一类非常重要的复杂查询,在多目标优化、数据挖掘等领域有着重要的应用[2]。具体来讲,Skyline查询就是从一个给定的多维数据对象集合S中挑选出不被S中任何数据对象支配的数据对象的集合,数据对象p支配数据对象k,如果p在所有维上都不比k差,并且至少在某一维上比k好[2]。

近年来,随着无线传感器网络的广泛应用,Skyline查询在无线传感器网络的应用越来越多[3],其中典型的应用包括森林火灾监控、环境污染检测、野生动物行为检测、泥石流灾害监控、园艺环境检测等。这些应用都需要综合几项指标来进行多目标优化。

与传统的Client/Server计算环境不同,无线传感器网络属于分布式多跳自组织对等网络。此外,由于在传感器网络中节点的能量都是由电池提供的,因此其能量有限。对于无线传感器网络而言,节点计算消耗的能量远小于节点间通信导致的能量消耗。因此,对于无线传感器网络中的Skyline查询处理而言,所有节点将感知数据通过邻近节点传送到基站(sink节点)后再在sink节点采用现有的Skyline查询算法的策略是不合适的。因为全部的感知数据在节点间的通信将消耗大量的能量,会显著降低整个无线传感器的生命周期。为此,必须针对无线传感器网络中节点能量有限的特点,

到稿日期:2016-08-05 返修日期:2017-01-25 本文受国家自然科学基金重大研究计划(91646117),国家自然科学基金(61170174),天津市自然科学基金(17JCYBJC15200),天津市科技特派员项目(16JCTPJC53600)资助。

李 青(1990-),女,硕士生,主要研究方向为数据库技术;肖迎元(1969-),男,教授,博士生导师,主要研究方向为数据库、个性化推荐系统,E-mail:yyxiao@tjut.edu.cn(通信作者);王晓晔(1973-),女,教授,主要研究方向为数据挖掘;李玉坤(1969-),男,副教授,主要研究方向为数据库技术。

研究高效利用能量的 Skyline 查询方法。

2 相关工作

传统分布式计算环境下的 Skyline 查询得到了广泛的研究。Katja 等人[4]对分布式环境下的现有 Skyline 算法进行了详尽的分析，并对其分类。Akrivi 等人[5]首次结合大规模分布式数据和子空间 Skyline 查询，提出了 SKYPEER 算法，有效地减少了转发的数据量。Wu 等人[6]在有大量服务器的情况下研究了平行 Skyline 查询的问题，提出了 DSL 算法，渐进地找到了 Skyline 点。Wang 等人[7]在 P2P 的环境下提出了 Skyframe 框架，优化了网络通信，降低了通信代价，并加快了查询的响应速度。Balke 等人[8]研究了在分布式 Web 环境下，随着服务器的不断增多，利用基于中间件的方法获得 Skyline 数据点，也提出了一些启发式方法来提高性能。但是现有的基于单服务器的 Skyline 查询算法已经不能很好地应用于无线传感器网络这类分布式多跳自组织网络。

无线传感器网络中的数据就是节点产生的实时数据，以数据流的形式存在，故无线传感器网络中 Skyline 查询又可分为数据流的连续查询和数据快照查询。

Tao 等人[12]在数据流的环境下提出了两个基于滑动窗口的算法框架，即 Lazy 算法和 Eager 算法。Xin 等人[13]在无线传感器网络的数据流环境下提出了能量有效的 SWSMA 算法，该算法采用两种过滤策略，即元组过滤策略和网格过滤策略，可以有效地减少数据转发量。信俊昌等人[14]提出了 FIST 算法，利用超立方作为全局过滤器来减少 Skyline 节点连续查询过程中的通信代价。Li 等人[15]考虑数据流中的倾斜数据，利用多层网格来处理动态的倾斜数据。

以上是基于数据流的连续 Skyline 查询。它需要不断地检测无线传感器网络，故能量消耗也在不断增加。而针对数据快照的 Skyline 查询只是在外部需求时才发出查询请求。文献[9-11,16]都是基于数据快照的 Skyline 查询算法的研究。

目前，传感器网络下的 Skyline 查询方法大都是基于分散树路由结构的。该路由结构一般以基站为根来创建生成树，每个传感器的感知数据都被存储在本地节点上。查询消息由根节点以洪泛的方式下发到各个节点，每个节点根据路由树将数据转发到根节点。Huang 等人[9]首次提出了利用过滤元组的方法来减少移动设备之间的通信，该方法可以应用到无线传感器网络中。Chen 等人[10]提出了基于阈值的分层方法以及改进过滤元组的 MinMax 方法，这些方法都是基于树路由结构的。KWON 等人[11]为了避免非 Skyline 元组在节点之间的传输，提出了基于簇的网内聚集的过滤算法，但是该算法对于减少通信代价并不高效。Su 等人[16]提出了基于聚簇路由结构的 SkySensor 算法，该方法利用过滤节点和过滤元组有效地过滤掉多余的存储节点和元组，通信代价明显减少。相比于其他算法中基于分散树的路由结构，该算法提出的聚簇结构在减少通信代价方面具有明显优势。这是因为该算法采用了以数据为中心的存储结构。但是在元组过滤方面，该算法并未考虑利用统治能力最强的元组进行过滤。

本文针对这一问题，利用聚簇结构来解决滑动窗口内的 Skyline 查询，提高了能量利用率，并优化了过滤策略，挑选具有最大支配力的数据元组作为全局过滤元组来过滤不满足 Skyline 条件的数据。

3 算法描述

3.1 算法的基本思想

本文提出基于聚簇结构的滑动窗口内连续 Skyline 查询算法，简记为 SWSKY 算法。该算法将 Skyline 查询划分为 3 个阶段：聚簇建立、数据过滤和数据回收。在进行 Skyline 查询处理之前需要建立聚簇，每个簇的构建信息都通过 GPSR 路由策略[17]发送给相应的簇中心，然后每个簇中心根据构建信息构建簇环。数组的维数决定了无线传感器网络(WSN)中簇的个数。其中数组的定义如下：

$$t=[D_1,D_2,\cdots,D_n] \tag{1}$$

其中，$D_i(i=1,2,\cdots,n)$表示无线传感器网络(WSN)节点监测的 n 种环境参数(对应 n 个数据维度)。

簇内的每个存储节点都有一个数据存储范围，网内节点产生的数据元组根据该存储范围来确定插入到哪个存储节点。WSN 中的感知数据全部都经过归一化处理，数据范围为[0,1]。如果节点产生的数据元组的第 i 维的数据 $t(D_i)$最小，且落在 $S_{i,j}$ 的数据范围内，则将数据元组插入到存储节点 $S_{i,j}$ 上。$S_{i,j}$ 的数据范围计算公式如下：

$$[L_{S_{i,j}},U_{S_{i,j}})=[\frac{j-1}{k},\frac{j}{k}) \tag{2}$$

查询处理时，利用滑动窗口机制来处理连续查询中的节点和元组过滤问题。网内的簇建好以后，首先通过数据过滤阶段的节点过滤元组 F_{node} 选出簇内的查询路径，然后在这条查询路径上选取支配能力最强的数据过滤元组 F_{tuple} 进行数据元组的过滤，最后进行数据的回收，得到最后的 Skyline 查询结果。SWSKY 算法的查询处理过程如图 1 所示。

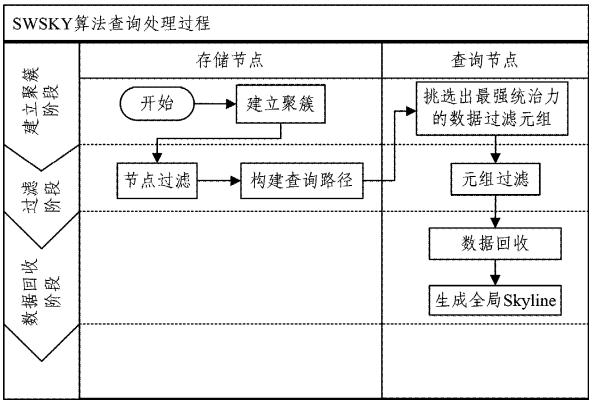


图 1 SWSKY 算法的查询处理过程

3.2 查询处理

在查询开始时，查询节点向距离其最近的簇发出查询请求。簇之间利用 GPSR 路由协议转发查询信息。选定簇之后，需要确定从簇内的哪个节点开始查询。原则上是选择已有存储数据元组且数据边界值最小的作为第一个查询处理节点。这样有助于剪枝，可以过滤掉更多无用的节点。最开始选择存储节点时，应该选择距离查询节点最近的簇内存储节点。若该存储节点是无效节点(无效节点是指该节点的数据边界值不是最小的，也指对 Skyline 结果无贡献的节点)或者无存储数据，则需要从该存储节点的概要信息表中找到数据

边界值最小的存储节点,并将其作为第一个真正处理查询的节点。

在查询处理过程中,需要利用过滤策略来过滤掉无效的节点和数据元组。WSN 中的数据是以数据流的形式存在的,因此必须要有适用于数据流的过滤策略,本文利用滑动窗口来处理数据流。因为 WSN 中的节点都是按照固定周期采集数据的,所以根据节点传输数据的频率确定滑动窗口的大小 W,以保证存储节点的数据不会溢出。这里需要同时考虑聚簇结构和滑动窗口的特点而提出新的过滤策略,过滤分为两个阶段:节点过滤和元组过滤。

3.2.1　节点过滤

SWSKY 算法可以通过节点过滤元组即 F_{node} 过滤掉簇内的无效节点。为此在连续 Skyline 查询中,需要维护 F_{node},减少能量消耗。假设存储节点上 $S_{i,k}$ 的数据元组集为 $SET_{S_{i,k}}$,该节点存储的元组个数为 n,则 F_{node} 的计算方法如下:

$$F_{node}=\mathrm{MIN}(\{\mathrm{MAX}(SET_{S_{i,k}}.t_j\}) \tag{3}$$

其中,t_j 是元组集合 $SET_{S_{i,k}}$ 中的第 j 个元组,$j=1,\cdots,n$。

在节点过滤阶段可以根据滑动窗口来维护 F_{node}。基于数据流的情形,可以将节点过滤再分为两个阶段,即维护阶段的节点过滤和查询阶段的节点过滤。

(1)维护阶段的节点过滤

假设当前时刻为 t_{now},数据元组 t 存入存储节点的时刻为 t_{arr},如果 $t_{arr}+W<t_{now}$,则说明该数据元组过期,需要将其从系统中删除。每到达一个新的元组 t' 或者删除一个过期元组 t,都应重新计算节点的节点过滤元组 F^k_{node}。如果新的节点过滤元组 F^{new}_{node} 比旧的节点过滤元组 F^{old}_{node} 大,则更新 F^k_{node},即 $F^k_{node}=F^{new}_{node}$($F^{old}_{node}<F^{new}_{node}$)。

(2)查询阶段的节点过滤

当查询开始时,假设当前系统的节点过滤元组 F_{node} 为存储节点 $S_{i,k}$ 的节点过滤元组 F^k_{node}。当查询信息从存储节点 $S_{i,k}$ 转发到另一个存储节点 $S_{i,k+1}$ 时,由于两个节点的节点过滤元组不同,因此也需要判断哪个节点作为最新的 F_{node}。判断方法与维护阶段采用的方法类似,即若 $S_{i,k+1}$ 的节点过滤元组 F^{k+1}_{node} 比 $S_{i,k}$ 的节点过滤 F^k_{node} 元组大,则更新系统的 F_{node},$F_{node}=F^{k+1}_{node}$,否则不更新。将最新的 F_{node} 传到下一节点,并利用该节点的过滤元组进行过滤。每一个簇经过节点过滤之后都会确定该簇上的查询路径上的节点,最终构成查询路径。每加入查询路径的一个节点,则将该节点的 ID 加入 SensorIDlist 中,并保存查询路径。

算法具体实现的伪代码描述如算法 1 所示。

算法 1　节点过滤算法

输入:存储节点上的归一化感知数据

输出:返回查询路径

1. 计算每个节点的节点过滤元组 F^k_{node};
2. 初始化 $F^{old}_{node}=F^k_{node}$;
3. 系统的节点过滤元组记为 F_{node};//当前系统的节点过滤元组还未更新
4. While(系统内无查询信息)
5. If(有新的元组存入)
6.　重新计算节点过滤元组,记为 F^{new}_{node};
7.　If($F^{new}_{node}>F^{old}_{node}$)
8.　　$F^k_{node}=F^{old}_{node}$;
9.　　更新概要信息表;
10. If($t_{arr}+W<t_{now}$)
11. 删除该过期元组 t;
12. 重新计算节点过滤元组 F^{new}_{node};
13. If($F^{new}_{node}>F^{old}_{node}$)
14.　$F^k_{node}=F^{old}_{node}$;
15. While(系统内出现查询信息)
16. 更新查询信息表;
17. 计算第一个处理查询的节点 $S_{i,k}$;
18. SensorIDlist+=$S_{i,k}$·ID;
19.　If $S_{i,k+1}.L<F^k_{node}$
20.　　SensorIDlist+=$S_{i,k+1}$·ID;
21.　If($F^{k+1}_{node}>F^k_{node}$)
22.　　$F_{node}=F^{k+1}_{node}$;
23.　Else
24.　$S_{i,k+1}$ 即为最后一个查询节点;
25. 返回 SensorIDlist。

3.2.2　元组过滤

通过节点过滤策略已经过滤掉大量的无效节点,但是有效存储节点内仍存储了大量的对于 Skyline 结果集无用的数据元组。为了保证能量有效,延长 WSN 系统的生命期,仍需过滤掉这些数据元组。因此选择簇内查询路径上的存储节点内支配能力最强的数据元组作为数据过滤元组 F_{tuple}。数据元组的统治能力计算方法如下:

$$t.DR=\prod_{i=1}^{d}(1-t(D_i)) \tag{4}$$

元组过滤也分为两个阶段:维护阶段的元组过滤和查询阶段的元组过滤。

(1)维护阶段的元组过滤

与节点过滤相似,每个存储节点都有自己的数据过滤元组 f^k_{tuple}。根据式(4)计算出支配能力最强的元组,并将其作为存储节点的元数据过滤元组。在未发出查询请求的前提下,每个存储节点只需维护其自身的数据过滤元组,整个网络中系统不必有统一的数据过滤元组。当 WSN 中有新的数据元组到达或者数据元组因过期而被删除,就需要重新计算节点内的数据过滤元组 f^k_{tuple},如果新的数据过滤元组的支配能力比旧的数据过滤元组的强,则更新 f^k_{tuple},即 $f^k_{tuple}=f^{new}_{tuple}$($f^{new}_{tuple}>f^{old}_{tuple}$),否则不更新。

(2)查询阶段的元组过滤

在进行数据集回收之前需要先进行元组过滤,得到每个存储节点的子 Skyline 结果集。在节点过滤阶段每得到一个查询路径上的节点,就计算出整个网络中暂时的数据过滤元组,直到网络内最后一个节点计算出最终的 F_{tuple}。假设当前系统的数据过滤元组 F_{tuple} 为存储节点 $S_{i,k}$ 的数据过滤元组 f^k_{tuple}。当查询信息从存储节点 $S_{i,k}$ 转发到另一个存储节点 $S_{i,k+1}$ 时,由于两个节点的数据过滤元组不同,因此需要判断选取哪个节点作为整个系统的 F_{tuple}。如果 $S_{i,k+1}$ 的数据过滤元组 f^{k+1}_{tuple} 的支配能力比 $S_{i,k}$ 的数据过滤元组 f^k_{tuple} 的强,则需更新系统的 F_{tuple},即 $F_{tuple}=f^{k+1}_{tuple}$($f^{new}_{tuple}.DR>f^{old}_{tuple}.DR$),否则不更新。

在数据收集阶段采用统治能力最强的 F_{tuple} 进行过滤。虽然传输数据过滤元组需要消耗能量,但是查询路径上的存储节点数量很少,因此传输数据元组消耗的能量远远小于传

输已过滤掉的数据元组的能量。

算法具体实现的伪代码描述如算法 2 所示。

算法 2 元组过滤算法

输入:存储节点上的归一化感知数据

输出:返回系统数据过滤元组 F_{tuple}

1. 分别计算每个节点的数据过滤元组,第 k 个节点的数据过滤元组表示为 f_{tuple}^{k};
2. 初始化 $f_{tuple}^{old}=f_{tuple}^{k}$;
3. While(系统内无查询信息)
4. If(新元组存入存储节点)
5. 重新计算该节点的数据过滤元组,记为 f_{tuple}^{new};
6. If($f_{tuple}^{new}.DR>f_{tuple}^{old}.DR$)// 新的数据过滤元组支配能力强
7. $f_{tuple}^{k}=f_{tuple}^{new}$;//更新该节点的数据过滤元组
8. If($t_{arr}+W<t_{now}$)
9. 删除该元组 t;
10. 重新计算该节点的数据过滤元组 f_{tuple}^{new};
11. If($f_{tuple}^{new}.DR>f_{tuple}^{old}.DR$)// 新的数据过滤元组支配能力强
12. $f_{tuple}^{k}=f_{tuple}^{new}$;//更新该节点的数据过滤元组
13. While(系统内发出查询请求)
14. 系统中旧的过滤元组 $F_{tuple}=f_{tuple}^{k}$
15. If($f_{tuple}^{k+1}.DR>f_{tuple}^{k}.DR$)
16. $f_{tuple}^{new}=f_{tuple}^{k+1}$;
17. If($f_{tuple}^{new}.DR>F_{tuple}.DR$))
18. $F_{tuple}=f_{tuple}^{new}$;
19. 返回 F_{tuple}。

3.3 Skyline 结果集的回收

在过滤阶段已经得到一条查询路径和支配能力最强的数据过滤元组 F_{tuple}。该算法从查询路径的最后一个节点用 F_{tuple} 开始进行数据元组的过滤,得到查询路径上每个节点的结果集。每一个查询路径上的存储节点都需要将其本地结果集跟前一个存储节点上的数据进行合并,并将合并之后的结果转发到前一个存储节点。最后由第一个节点将所有可能属于 Skyline 集的数据转发至查询节点,并计算出最后的 Skyline 结果。

4 实验结果与分析

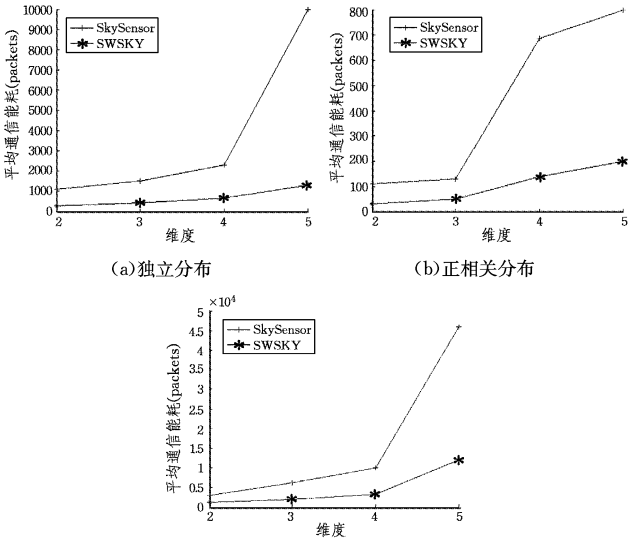
4.1 实验设置

本节比较了本文提出的 SWSKY 算法与 SkySensor 算法在查询阶段的平均传输消耗。实验的硬件环境为 2.30GHz HP PC,2.0GB 内存,操作系统为 32 位的 Win7 系统。该实验利用 MATLAB 进行仿真,实验所用的数据为合成数据,合成的数据集包括 3 类数据(独立分布数据、正相关数据、反相关数据),并分别在这 3 类数据集上进行了算法的性能比较。本实验主要考察了数据维度和网络规模发生变化时查询阶段平均消耗的能量,即传输的数据元组(packets)的数量。

实验中节点的随机分布区域范围为 200m×200m~600m×600m,默认的区域范围为 400m×400m。节点分布密度为 1node/256m^2。感知数据元组维度在 2~5 之间变化,默认值为 3。实验中假定每 1min 产生一次数据,滑动窗口的大小 $W=120$min。

4.2 实验结果

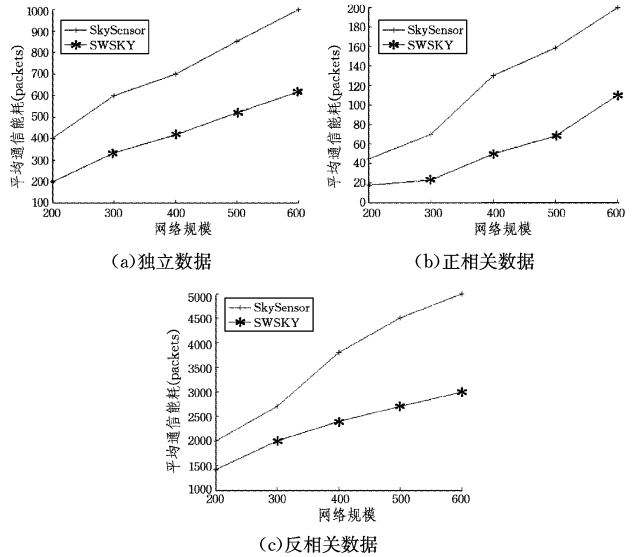
实验结果显示了在不同数据维度和不同网络规模下 SWSKY 算法与 SkySensor 算法的能量消耗对比,即传输的数据元组数量。如图 2 所示,在默认的区域范围内,分别在 3 种不同数据集进行实验,分析不同数据维度的平均能量消耗。从图 3 可知,随着数据维度的不断增加,两种算法在 3 种数据集上的通信消耗都不断增加,而且当数据集是反相关数据时,通信消耗明显最大。



(a)独立分布 (b)正相关分布

(c)反相关分布

图 2 不同维度下两种算法的通信消耗对比



(a)独立数据 (b)正相关数据

(c)反相关数据

图 3 不同网络规模下两种算法的通信消耗对比

从图 2 可以看到,SWSKY 算法明显优于 SkySensor 算法。因为 SWSKY 算法是用支配能力最强的数据过滤元组进行过滤,所以过滤掉的数据元组比 SkySensor 多。因此可以得到:即使是在反相关数据集下,SWSKY 算法消耗的能量依然很少。当数据维度增加时,SWSKY 算法的平均通信消耗平稳增加;而当数据维度大于或等于 4 时,SkySensor 算法的通信消耗增加急剧,尤其是在反相关数据集下。

从图 2 中还可以看出,在正相关的数据集下,平均通信消耗最少。这是因为当数据满足正相关分布时,满足支配关系的数据元组最多,所以被过滤掉的数据元组就会增多。而当数据集满足反相关分布时,满足支配关系的数据元组较少,所以被最强过滤元组过滤掉的数据元组就少很多。因此在反相

关数据集上,Skyline 查询的平均通信消耗最大。

当数据维不变且网络规模不断增加时,节点数不断增加,导致数据元组的传输数量也在不断增加,从而导致最终的 Skyline 结果集增大。图 3 给出了当数据维度为默认值,分别在 3 种不同数据集进行实验,并且网络规模从 200m×200m 到 600m×600m 变化时,查询阶段的平均通信消耗。SWSKY 算法同样利用 F_{tuple} 进行元组过滤,在 3 种数据集下 SWSKY 查询算法都要明显优于 SkySensor 算法。

结束语　Skyline 查询在无线传感器网络中的应用日趋增多,减少能量的消耗依然是无线传感器网络的一个研究重点。本文提出的 SWSKY 算法在聚簇结构的基础上对过滤策略进行改进,在过滤时选取了支配力最强的过滤元组,它能有效地减少无效数据元组的传输,从而节省 WSN 能量,延长 WSN 的生命周期。

参 考 文 献

[1] 颜振亚,郑宝玉. 无线传感器网络[M]. 北京:清华大学出版社,2005.

[2] XIAO Y Y,CHEN Y G. Efficient distributed skyline queries for mobile applications[J]. Journal of Computer Science and Technology,2010,25(3):523-536.

[3] WANG H X,ZHENG J P,SONG B L. Skyline Query Processing in Wireless Sensor Networks[J]. Journal of Computer Science,2013,40(8):14-23. (in Chinese)
王海翔,郑吉平,宋保利. 无线传感器网络中的 Skyline 查询处理技术[J]. 计算机科学,2013,40(8):14-23.

[4] HOSE K,VLACHOU A. A survey of skyline processing in highly distributed environments[J]. Vldb Journal,2012,21(3):359-384.

[5] VLACHOU A,DOULKERIDIS C,KOTIDIS Y,et al. SKYPEER: Efficient Subspace Skyline Computation over Distributed Data[C]// 2014 IEEE 30th International Conference on Data Engineering. IEEE,2007:416-425.

[6] WU P,ZHANG C,FENG Y,et al. Parallelizing Skyline Queries for Scalable Distribution[C]// International Conference on Advances in Database Technology-edbt. 2006:112-130.

[7] WANG S,VU Q H,OOI B C,et al. Skyframe:A framework for skyline query processing in peer-to-peer systems[J]. Vldb Journal,2009,18(1):345-362.

[8] BALKE W T,GÜNTZER U,ZHENG J X. Efficient Distributed Skylining for Web Information Systems[J]. Lecture Notes in Computer Science,2004,2992:256-273.

[9] HUANG Z,JENSEN C S,LU H,et al. Skyline queries against mobile lightweight devices in MANETs[C]// International Conference on Data Engineering. 2006:66.

[10] CHEN H,ZHOU S,GUAN J. Towards Energy-Efficient Skyline Monitoring in Wireless Sensor Networks[C]// Wireless Sensor Networks,European Conference(Ewsn 2007). Delft,the Netherlands,2007:101-116.

[11] KWON Y,CHOI J H,CHUNG Y D,et al. In-Network Processing for Skyline Queries in Sensor Networks [J]. Ieice Transactions on Communications,2007,90(12):3452-3459.

[12] TAO Y,PAPADIAS D. Maintaining sliding window skylines on data streams[J]. IEEE Transactions on Knowledge & Data Engineering,2006,18(3):377-391.

[13] XIN J,WANG G,CHEN L,et al. Continuously Maintaining Sliding Window Skylines in a Sensor Network[C]// International Conference on Database Systems for Advanced Applications (DASFAA 2007). Bangkok,Thailand,2007:509-521.

[14] XIN J C,WANG G R. Continuous Skyline Nodes Query Processing over Wireless Sensor Networks[C]// NDBL 2012. 2012:2415-2430. (in Chinese)
信俊昌,王国仁. 无线传感器网络中 Skyline 节点连续查询算法[C]// 中国数据库学术会议. 2012:2415-2430.

[15] LI H,YOO J. An efficient scheme for continuous skyline query processing over dynamic data set[C]// International Conference on Big Data and Smart Computing. 2014:54-59.

[16] SU I F,CHUNG Y C,LEE C,et al. Efficient skyline query processing in wireless sensor networks[J]. Journal of Parallel & Distributed Computing,2010,70(6):680-698.

[17] KARP B,KUNG H T. GPSR:greedy perimeter stateless routing for wireless networks[C]// International Conference on Mobile Computing and Networking. ACM,2000:243-254.

(上接第 149 页)

[6] SUN B,QU L J,LI C. Impossible Differential Cryptanalysis of SNAKE[C]// Procof NSWCT'09. 2009:63-66.

[7] TOZ D,DUNKELMAN O. Analysis of two Attacks on Reduced-Round Versions of the SMS4[M]// Information and Communications Security. Springer Berlin Heidelberg, 2008: 141-156.

[8] WANG G L. Improved Impossible Differential Cryptanalysis on SMS4[C]// International Conference on Communications and Intelligence Information Security. IEEE,2010:105-108.

[9] LIU Y,GU D,LIN Z,et al. Impossible differential attacks on reduced-round Lblock [C]// ISPEC 2012. 2012:97-108.

[10] LIU X. The design and implementation of the lightweIght block ciper ESF[D]. Shandong: Shandong Normal University, 2014. (in Chinese)
刘宣. 轻量级分组密码 ESF 的设计与实现[D]. 山东:山东师范大学,2014.

[11] LIU X,LIU F,MENG S. Impossible differential cryptanalysis of lightweight block ciper ESF [J]. Computer and Engineering Science,2013,35(9):89-95. (in Chinese)
刘宣,刘枫,孟帅. 轻量级分组密码算法 ESF 的不可能差分析[J]. 计算机工程与科学,2013,35(9):89-95.

[12] CHEN Y L,WEI H R. Impossible Differential Cryptanalysis of ESF[J]. Computer Science,2016,43(8):89-91. (in Chinese)
陈玉磊,卫宏儒. ESF 算法的不可能差分密码分析[J]. 计算机科学,2016,43(8):89-91.

[13] XU P. Differential Fault Analysis on lightweight block ciper ESF[J]. Network Security Technology & Application, 2016(1):99-100. (in Chinese)
徐朋. 轻量级分组密码 ESF 的差分故障攻击[J]. 网络安全技术与应用,2016(1):99-100.