无线传感器网络中Skyline节点连续查询算法

信俊昌 王国仁

(医学影像计算教育部重点实验室(东北大学) 沈阳 110819)
(东北大学信息科学与工程学院 沈阳 110819)

摘 要 作为多目标决策的重要手段之一,Skyline节点查询在传感器网络应用中发挥着非常重要的作用.文中深入地分析了Skyline节点查询的性质,提出了基于过滤的Skyline节点连续查询算法(FIlter based Skyline moniToring algorithm,FIST).FIST算法共包括自底向上、自顶向下和混合3种过滤方式,均通过在传感器节点设置本地或全局过滤器来避免不必要的数据传输,进而节约传感器节点的能量.自底向上过滤方式通过缓存先前Skyline结果作为本地过滤器来避免数据重复传输,而自顶向下过滤则通过设置超立方体作为全局过滤器来避免数据反复更新.由于两者各有利弊,因而提出了混合过滤方式,通过为节点选择合适的过滤器来扬长避短.大量仿真实验的结果表明,FIST算法能有效地减少Skyline节点连续查询过程中传感器节点的通信代价,进而降低传感器网络的能量消耗.

关键词 无线传感器网络;Skyline节点查询;能量有效性;过滤
中图法分类号 TP393 **DOI号**:10.3724/SP.J.1016.2012.02415

Continuous Skyline Nodes Query Processing over Wireless Sensor Networks

XIN Jun-Chang WANG Guo-Ren

(*Key Laboratory of Medical Image Computing (NEU) of Ministry of Education, Shenyang* 110819)
(*College of Information Science and Engineering, Northeastern University, Shenyang* 110819)

Abstract As an important operator for multiple criteria decision making, Skyline node query plays a very important role in many sensing applications. In this paper, the properties of Skyline node query are theoretically analyzed, and then a novel approach, called FIlter based Skyline monitoring algorithm (FIST), is proposed to evaluate the continuous Skyline node query in wireless sensor networks. Specifically, three filtering mechanisms, including bottom-up filtering, top-down filtering and hybrid filtering, are introduced. All of them install local or global filters within each sensor to reduce the amount of data transferred among sensor nodes and save the energy consumption as a consequence. The bottom-up filtering mechanism caches the previous Skyline results as the local filter to avoid the unnecessary data transmissions, while the top-down filtering mechanism assigns a hypercube to each sensor node as the global filter to suppress the unnecessary data updates. Since both bottom-up filtering and top-down filtering have their own pros and cons, a hybrid filtering mechanism is proposed to choose the "right" filter for each sensor node to avoid their disadvantages and fully utilize their advantages. Our extensive simulation studies show that FIST approach can effectively reduce the communication cost among sensor nodes and save the energy consumption during the continuous Skyline node query evaluation in wireless sensor networks.

Keywords wireless sensor network; Skyline node query; energy-efficiency; filtering

收稿日期:2010-03-19;最终修改稿收到日期:2012-07-15.本课题得到国家自然科学基金重点项目(60933001)、国家杰出青年科学基金项目(61025007)、国家自然科学基金青年科学基金项目(661100022)和中央高校基本科研业务费专项资金(N110404009)资助.**信俊昌**,男,1977年生,博士,讲师,主要研究方向为感知数据管理、云计算和不确定数据管理.E-mail:xinjunchang@ise.neu.edu.cn;xinjunchang@gmail.com.**王国仁**,男,1966年生,教授,博士生导师,主要研究领域包括不确定数据管理、数据密集型计算、可视媒体数据管理与分析、非结构化数据管理、分布式查询处理与优化技术和生物信息学.

1 引 言

随着传感技术、通信技术和嵌入式技术的发展，无线传感器网络(Wireless Sensor Network，WSN)在安全防御、交通管理、医疗卫生和灾难预防等领域得到了非常广泛的应用[1-2]. 由于硬件发展水平的限制，传感器节点通常利用电池作为电源. 由于传感器网络大都工作在湖泊、沼泽、沙漠和火山口等人类不易接近的恶劣环境中，更换电池或者为电池充电都是十分困难的. 因此，能量高效性是无线传感器网络应用所面临的主要挑战之一.

作为多目标决策的重要手段之一，Skyline 查询得到了广泛的研究[3-4]. 在 d 维数据库 D 中，如果对象 p 在任何一维都不比对象 q 差，并且在至少一维比对象 q 好，那么对象 p 支配对象 q. D 中所有不被其它任何对象所支配的对象构成了 D 的 Skyline. 如图 1 所示，虚线上的 n_1、n_2 和 n_6 构成了图中数据的 Skyline.

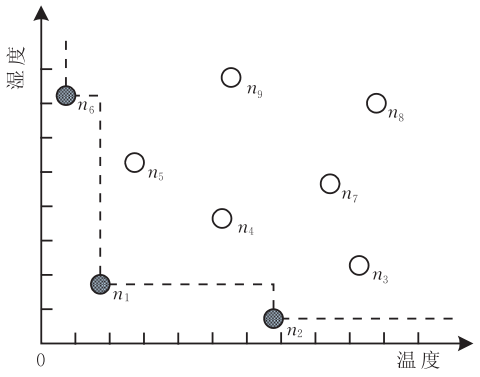


图 1 Skyline 举例

在一些传感器网络应用中，用户往往只关心哪些节点是 Skyline 节点，而不关心 Skyline 节点感知数据的具体数值，称之为 Skyline 节点查询. 例如，在森林监控应用中，若某个地点温度高、湿度低，则该林区更容易发生火灾；反之，则不容易发生火灾. 因此，护林员只需要知道哪些林区是火灾的高危区，而不需要知道高危林区温度与湿度的具体数值，即可实现对高危林区的重点防控. 此时，护林员可以只需向部署在林区的传感器网络发送 Skyline 节点连续查询，就可以时刻知道哪些林区具有高温度、低湿度，从而第一时间发现火灾隐患并采取措施防止火灾的发生.

已有算法大都假定数据保存在一个集中式服务器并且在该集中式环境下计算 Skyline. 由于无线通信对节点能量消耗的巨大影响，将所有的感知数据都收集到基站是不可取的，因而需要研究能量高效的分布式算法处理传感器网络中的 Skyline 节点查询. 通过分析 Skyline 节点查询的性质，提出了一种新颖的基于过滤的 Skyline 节点连续查询算法(FIlter-based Skyline node moniToring algorithm，FIST)连续查询传感器网络中的 Skyline 节点. FIST 算法通过在传感器节点设置本地或全局过滤器来抑制不必要的数据传输，进而节约节点的能量. 根据过滤器的设置规则，FIST 算法可以分为自底向上过滤、自顶向下过滤和混合过滤 3 种方式. 本文的主要贡献点如下：

(1) 提出了本地过滤器和全局过滤器，深入地分析了两种过滤器的有效性.

(2) 以上述过滤器为基础，提出了自底向上、自顶向下和混合共 3 种连续 Skyline 节点查询过滤机制.

(3) 设计了详尽的性能评价实验，实验结果表明 FIST 算法极大地降低了连续 Skyline 节点查询过程中传感器节点的数据传输量.

本文第 2 节介绍相关的研究工作；第 3 节对研究的问题进行描述和分析；第 4 节介绍基于过滤的 Skyline 节点连续查询算法 FIST；第 5 节通过实验对 FIST 算法的性能进行验证；最后，第 6 节总结全文.

2 相关工作

Chen等人[5]研究了传感器网络中的连续 Skyline 查询问题，提出了基于层次型路由结构的过滤算法 MINMAX 来减少网络中的数据传输量. 文献[6-7]研究了传感器网络中的滑动窗口轮廓查询. 前者提出了一种基于元组过滤器和格过滤器的算法来连续地查询传感器网络中的滑动窗口轮廓；后者提出了基于映射的过滤方式来提高计算的性能. 通过映射函数将感知数据映射到整数空间，再以整数空间的 Skyline 作为过滤器来减少网络的数据传输量. 文献[8]研究了传感器网络中的多 Skyline 查询优化问题，提出了基站与节点两阶段优化的多 Skyline 查询算法. Shen 等人[9]研究了传感器网络中基于位置的二维 Skyline 查询并提出了按照距离从近到远的顺序计算每个环中的 Skyline 的 Ring-Skyline 算法. Su 等人[10]提出了基于聚簇型路由结构的以数据为中心的 Skyline 查询算法 SkySensor. SkySensor

算法通过不同节点发出的多个 Skyline 查询共享同一感知数据搜集过程来降低每个查询的能量消耗进而提高查询的执行效率.

3　预备知识

在本节中，首先详细阐述传感器网络中的 Skyline 节点查询；接着，分别介绍本地过滤器和全局过滤器并且深入分析两种过滤器的过滤原理和有效性.

3.1　问题描述

基于 Skyline 查询的基本概念，Skyline 节点查询的定义如下.

定义 1.　传感器网络中的 Skyline 节点查询返回所有不被其它节点所支配的节点. 节点 n_i 支配节点 n_j 当且仅当节点 n_i 的感知数据 r_i 支配节点 n_j 的感知数据 r_j，即在任何一维 r_i 都不比 r_j 差（$\forall k, r_i.v_k \leq r_j.v_k$），并且在至少一维比 r_j 好（$\exists l, r_i.v_l < r_j.v_l$）.

利用集中式算法在基站计算 Skyline 节点是实现 Skyline 节点连续查询的最基本也是最“昂贵”的方式. 由于 Skyline 节点查询可分解[6]，可以利用网内计算技术[11]的方式来提高算法的能量有效性. 如图 2 所示，传感器节点 n_1、n_2、n_3、n_4 和 n_5 构成了一棵以 n_1 为根的路由树. 节点旁边表格里显示的是连续 3 个时刻对应节点采集的感知数据. 其中，第 1 列是数据的采集时刻，其余两列则是具体的感知数值. 集中式算法需要将所有感知数据传输到基站，那么 3 个时刻共传输 33 个元组. 而利用网内计算技术[11]，则每个节点只需传输以自己为根的子树的局部 Skyline 结果，避免了只属于局部 Skyline 而不属于全局 Skyline 的元组的传输，那么 3 个时刻只需要传输 21 个元组. 然而，在网内计算方法中，仍然有大量的无用数据在网络中传输. 例如，节点 n_4 在第 2 个时刻的感知数据为(65,33)，由于该数据只被节点 n_3 的感知数据支配，会被依次地传输到 n_2 和 n_1，而实际上，(65,33)不属于最后的 Skyline 结果，因此节点 n_4 并不是 Skyline 节点. 基于过滤的 FIST 算法的目标就是减少网络中类似的不必要的数据传输.

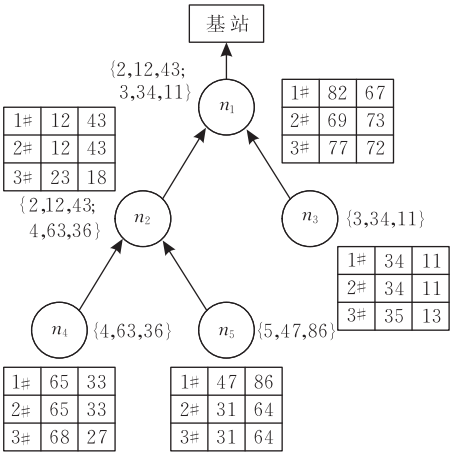


图 2　本地过滤器设置举例

传感器节点采集的传感数据是对周围环境的温度、湿度和光照等连续物理量的离散采样，因此，不同时刻的传感数据之间具有非常密切的时间相关性[12]. 因而可以利用统计信息在节点设置过滤器，利用过滤器阻止不必要的数据传输. 根据过滤器的产生方式，可以将其分为本地过滤器和全局过滤器.

3.2　本地过滤器

本地过滤器的基本思想是每个传感器节点保存前一个时刻的 Skyline 作为过滤器用来避免相同数据的重复传输. 同时，父节点也保存本地过滤器的一个副本以便保持两者之间的同步. 如果节点采集的感知数据属于局部 Skyline 并且与前一时刻的感知数据相同，由于该感知数据已经存储于父节点的副本中，因而不需要将其传输到父节点；只有感知数据虽属于局部 Skyline 却与前一时刻的感知数据不同，才需要将其传输到父节点. 由于该类过滤器是传感器节点根据自身的局部 Skyline 设置的，使用的是本地信息，所以称之为本地过滤器.

如图 2 所示，每个传感器节点都设置了一个本地过滤器，即时刻 0 的局部 Skyline. 本地过滤器中每个元组的第一个元素表示感知数据的节点编号，其余元素为感知数据的数值. 例如，传感器节点 n_2 的本地过滤器包含元组(2,12,43)和元组(4,63,36). 在时刻 1，节点 n_4 的感知数据从(63,36)变成了(65,33)，其余节点的感知数据没有发生变化. 此时，节点 n_4 的局部 Skyline 变为(4,65,33)，由于没有包含在本地过滤器中，因此需要传输给它的父节点 n_2. 节点 n_5 的局部 Skyline 未发生变化，因此不需向父节点 n_2 传输数据. 接着，节点 n_2 根据从 n_4 接收到的更新信息和本地存储的信息计算得到局部 Skyline{(2,12,43),(4,65,33)}. 元组(2,12,43)与过滤器中的元组相同，不需要进行传输；而元组(4,65,33)与过滤器中的元组不同，需要传输到父节点 n_1. 由于节点 n_3 的局部 Skyline 没有发生变化，因而不向节点 n_1 发送消息. 接着，节点 n_1 计算新的局部 Skyline，发现局部 Skyline 结果与本地过滤器完

全相同,因此不需要向父节点 n_1 传输任何数据.最后,基站没有收到任何更新信息,可以判定时刻 1 的 Skyline 与上时刻的 Skyline 相同.时刻 2 和时刻 3 的处理过程与时刻 1 的处理过程类似.这样在 3 个连续时刻共有 9 个元组在传感器网络中传输,与网内计算[11]相比节约了 12 个元组的数据传输代价.

3.3 全局过滤器

当传感器节点采集的感知数据不稳定时,本地过滤器的过滤效果将会变得很差.利用全局知识为每个传感器节点设置一个超立方体作为过滤器来阻止无用数据传输是一个很自然的想法.基站中也同步保存着这些超立方体过滤器的副本以便基站可以随时了解感知数据分布的大致情况.传感器节点只向基站报告超出了超立方体的范围的感知数据,而基站则根据超立方体过滤器的副本和接收的感知数据更新信息来求解 Skyline 节点集合.根据超立方体之间的关系,传感器节点可以划分为 Skyline 节点、非 Skyline 节点和未确定节点 3 类.基站需要获取未确定节点的感知数据来得到最终的 Skyline 节点集合.

首先,分析超立方体之间的关系;接着,介绍如何判断一个节点是 Skyline 节点还是非 Skyline 节点.

定义 2. 给定两个区间 $a(a.l,a.u)$ 和 $b(b.l,b.u)$.如果 $a.u \leq b.l$,则称之为区间 a 支配区间 b,记为 $a \geq b$;如果 $a.l<b.l$ 且 $a.u>b.l$,则称之为区间 a 和 b 关系不确定,记为 $a?b$.

图 3 中给出了区间 a 和区间 b 之间相互关系的例子.

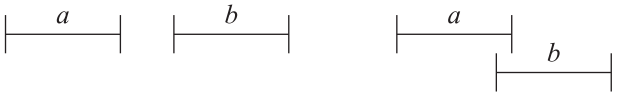


图 3 区间关系举例

超立方体可以表示为 $I_1 \times I_2 \times \cdots \times I_d$.在区间关系的基础上,超立方体之间的关系如定义 3 所示.

定义 3. 给定两个超立方 c 和 d.如果在所有维超立方体 c 的区间均支配 d 的区间,则称之为超立方体 c 支配超立方体 d,记为 $c \geq d$;如果在某一维超立方体 c 的区间支配 d 的区间,而在另外一维超立方体 d 的区间支配 c 的区间,则称之为超立方体 c 和 d 互相不支配,记为 $c<>d$;在其余情况下,超立方体 c 和 d 的关系并不明确,称之为超立方体 c 和 d 之间的支配关系不确定,记为 $c?d$.

图 4 中详细地描述了二维环境下超立方体 c 和超立方体 d 之间的上述 3 种相互关系.

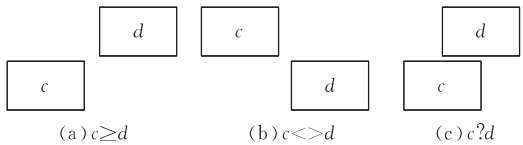


图 4 超立方体间关系举例

设 t 和 t' 是数据空间中的两个元组,超立方体之间的关系具有下列性质.

定理 1. 给定两个超立方 h_i 和 h_j,如果 $h_i \geq h_j$,则 $\forall t \in h_i, t' \in h_j, t \geq t'$.

证明. 根据定义 3 中超立方体之间的关系,可得 $h_i \geq h_j \Rightarrow \forall k, h_i.I_k.u \leq h_j.I_k.l$.

又因为 $t \in h_i \Rightarrow \forall k, t.v_k < h_i.I_k.u$,并且 $t' \in h_j \Rightarrow \forall k, t'.v_k > h_j.I_k.l$,所以 $\forall k, t.v_k < t'.v_k$.

因此,根据支配关系定义可得 $t \geq t'$. 证毕.

定理 1 说明了什么样的传感器节点可以作为非 Skyline 节点被提前过滤.下面的引理和定理将说明什么样的传感器节点为 Skyline 节点.

引理 1. 给定两个超立方 h_i 和 h_j,如果 $h_i<>h_j$,则 $\forall t \in h_i, \forall t' \in h_j, t<>t'$.

证明. 根据定义 3 中超立方体之间的关系,$h_i<>h_{jj} \Rightarrow \exists k, h_i.I_k \geq h_j.I_k$ 并且 $\exists l, h_j.I_l \geq h_i.I_l$.

$t \in h_i \wedge t' \in h_j \wedge h_i.I_k \geq h_j.I_k \Rightarrow t.v_k \leq t'.v_k$,

$t \in h_i \wedge t' \in h_j \wedge h_j.I_l \geq h_i.I_l \Rightarrow t'.v_l \leq t.v_l$.

因此,根据支配关系定义可得 t 不能支配 t' 并且 t' 不能支配 t,即 $t<>t'$. 证毕.

引理 2. 给定 3 个超立方 h_i、h_j 和 h_k,相应的元组 $t_i \in h_i, t_j \in h_j, t_k \in h_k$,如果 $h_i<>h_j$ 且 $h_j \geq h_k$,则元组 t_k 肯定不支配 t_i.

证明. 用反证法证明.假设 $t_k \geq t_i$,根据引理 1,$h_j \geq h_k \Rightarrow t_j \geq t_k$.因为 $t_k \geq t_i$,所以 $t_j \geq t_i$.

又根据引理 1,$h_i<>h_j \Rightarrow t_i<>t_j$,与前面推出的 $t_j \geq t_i$ 相矛盾.因此,元组 t_k 肯定不支配 t_i. 证毕.

定理 2. 设 $H=\{h_1,h_2,\cdots,h_n\}$ 是超立方体的集合,$T=\{t_1,t_2,\cdots,t_n\}$ 是满足 $\forall i, t_i \in h_i$ 的元组集合.同时,设 $H_S=\text{Skyline}(H)$ 是集合 H 的 Skyline 集合,$H_N=H-H_S$ 是非 Skyline 的集合.如果 H_S 中的超立方体 h_i 与 H_S 中其它任意超立方体 h_j 之间都满足 $h_i<>h_j$,那么 $t_i \in \text{Skyline}(T)$.

证明. 根据定义 3,可得 $\forall h_k \in H_N, \exists h_l \in H_S, h_l \geq h_k$.根据已知 $\forall h_j \in H_S (j \neq i), h_i<>h_j$,可得 $h_i<>h_l$.

根据引理 2,$h_l \geq h_k$ 且 $h_i<>h_l \Rightarrow \forall t_k \in h_k, t_k$ 不

支配 t_i. 又根据引理 1，$\forall h_j \in H_S(j \neq i)$，$h_i <> h_j \Rightarrow t_i <> t_j$.

因此，不存在元组 $t_j \in h_j(j \neq i)$ 满足 $t_j \geq t_i$，所以 $t_i \in \text{Skyline}(T)$.　　证毕.

定理 2 说明了如何判断一个传感器节点为 Skyline 节点. 如果根据定理 1 和定理 2 都无法判断传感器节点到底是 Skyline 节点还是非 Skyline 节点，那么该节点为未确定节点. 此时，基站需要获取未确定节点的感知数据以消除上述情况带来的影响. 显然，超立方体过滤器的效果依赖于需要获取感知数据的未确定节点的数量. 由于超立方体过滤器需要基站根据整个传感器网络的统计信息进行设置，使用的是全局信息，所以称之为全局过滤器.

首先，基站根据历史信息建立各个传感器节点的感知数据模型，即感知数据的概率密度函数. 然后，根据感知数据模型计算一个初始过滤器设置方案，为每个节点计算并设置一个超立方体作为全局过滤器. 接着，如果新的感知数据落在超立方体的范围之内，说明没有通过过滤，因而不需要向基站传输任何信息；反之，如果新的感知数据落在超立方体的范围之外，说明通过了过滤，意味着 Skyline 节点集合可能发生改变，因而需要将感知数据的变化通知基站. 基站收到该消息后，意识到 Skyline 节点集合可能发生变化，重新计算传感器网络中的 Skyline 节点集合，同时根据需要进行过滤器的调整.

给每个传感器节点都设置一个合适的全局过滤器是一个非常复杂的工作，需要解决下面 3 个问题.

(1) 感知数据建模. 精确的感知数据模型是给每个传感器节点都设置一个合适过滤器的必备条件之一. 维护精确的感知数据模型需要频繁的数据更新操作，将消耗节点的大量能量. 如何利用较少的能量消耗来维护较为精确的感知数据是需要面临的一个重要挑战.

(2) 过滤器的设置. 通常情况下，需要尽可能地选择更好的过滤器设置方案来减少网络的数据传输量，然而，从大量的过滤器设置方案中选择出最好的方案要花费很大的计算代价. 如何利用较少的计算代价得到较好的过滤器设置方案是问题的难点.

(3) 查询结果维护. 如果传感器节点没有向基站提交任何数据更新信息，那么 Skyline 节点查询的结果保持不变；一旦传感器节点向基站提交了数据更新信息，基站需要根据过滤器设置方案和收到的数据更新信息重新计算 Skyline 节点查询的结果.

如图 5 所示，根据近期的历史统计信息，基站为传感器节点 n_1、n_2、n_3、n_4 和 n_5 分别设置了超立方体(二维时为矩形)[60,100)×[30,100)、[10,20)×[20,50)、[20,40)×[0,20)、[40,100)×[20,100)和[20,100)×[50,100)作为过滤器. 在时刻 0 和时刻 1，全部 5 个传感器节点的感知数据都在各自的过滤器的内部变化，因此不需要向基站提交任何的数据更新信息.

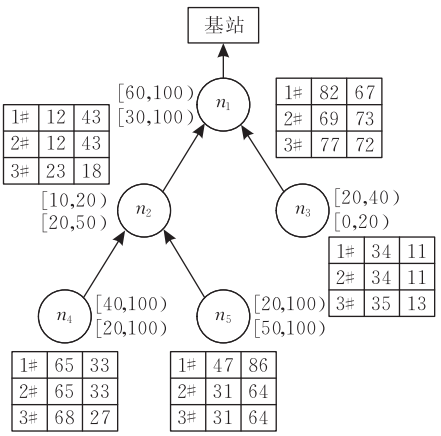


图 5　全局过滤器设置举例

值得注意的是，超立方体[10,20)×[20,50)中的每个元组始终都支配超立方体[20,100)×[50,100)中的所有元组，因此节点 n_2 的感知数据支配节点 n_5 的感知数据，即节点 n_2 支配节点 n_5. 同样地，节点 n_3 支配节点 n_1 和节点 n_4. 同时，由于超立方体[10,20)×[20,50)中元组的第一维都小于超立方体[20,40)×[0,20)中元组的第一维，并且超立方体[10,20)×[20,50)中元组的第二维又大于超立方体[20,40)×[0,20)中元组的第二维，因此，超立方体[10,20)×[20,50)和[20,40)×[0,20)中的元组互相不支配，也即是节点 n_2 和节点 n_3 互相不支配. 根据上述分析可知，无论是时刻 0 还是时刻 1，网络中的 Skyline 节点集合始终为$\{n_2, n_3\}$.

在时刻 3，节点 n_2 的感知数据变为(2,23,18)，超出了超立方体过滤器[10,20)×[20,50)的范围，因此需要向基站提交感知数据更新信息. 新的元组(2,23,18)与超立方体[20,40)×[0,20)和[20,100)×[50,100)在某些维度空间存在着交集，导致了节点 n_2 与节点 n_3 和 n_5 之间的关系无法确定，意味着无法直接判断哪些节点为 Skyline 节点. 因此，基站需要获取节点 n_3 和 n_5 当前的感知数据消除 Skyline 节点集合的不明确性. 在这个过程中，一共需要在传感器网络中传输 7 个元组的数据包和 1 个广播包.

3.4 过滤器开销分析

传感器节点 n 的本地过滤器由以 n 为根的路由子树的局部 Skyline 节点的感知数据构成，如果以 n 为根的路由子树包含的节点数目为 s，则节点存储本地过滤器的空间复杂度为 $O(\log^{d-1}s)$[13]，其中 d 为感知数据维数. 由于本地过滤器完全由节点维护，因而未给基站带来任何的额外开销.

传感器节点 n 的全局过滤器由基站分配的超立方体构成，因此全局过滤器的空间复杂度为 $O(d)$，其中 d 为感知数据维数. 需要注意的是，当传感器节点数目增加时，Skyline 节点的数量也相应增加，全局过滤器的粒度将越来越小，其过滤效果当然也逐渐削弱. 因此，全局过滤器更适合于中小规模的传感器网络，如何在大规模网络环境下进行过滤将是未来的研究工作之一.

全局过滤器的初始化和维护均需要基站发送大量的信息，因而假设基站具有持续的能量供应并且无线通信范围可以覆盖所有的传感器节点[14]，换言之，基站发送的广播信息可以不经过任何节点的转发而直接发送到所有的传感器节点. 此时，基站的能量消耗将不会影响传感器网络的使用寿命，全局过滤器的优化效果将非常显著. 顺便提一下，如果基站使用多跳的方式向传感器节点发送消息，FIST 算法同样有效，只是优化效果略微有所下降.

4 基于过滤的 Skyline 节点连续查询

4.1 自底向上方式

在自底向上方式的初始阶段，可以利用网内计算[11]的方法得到各节点的局部 Skyline 作为本地过滤器. 在算法后续执行过程中，为了动态地维护本地过滤器，传感器节点除了需要保存局部 Skyline 之外，还需要缓存所有子节点的局部 Skyline，以便在收到子节点的更新信息后，可以准确地计算出新的局部 Skyline. 这样，传感器节点需要存储的信息包括以下两个部分：

(1) 缓存. 所有子节点的本地过滤器，即所有子节点的局部 Skyline 构成的集合.

(2) 过滤器. 本地过滤器，即节点前一时刻的局部 Skyline.

既然本地过滤器中的信息在父节点中存在完整的备份，那么传感器节点只需要向父节点传输新的局部 Skyline 与本地过滤器中的 Skyline 不同的部分即可，包括以下 3 个部分：

(1) 当非局部 Skyline 节点的感知数据发生变化使之成为新的局部 Skyline 节点时，由于父节点中没有存储任何关于该局部 Skyline 元组的相关信息，因此它的节点编号和感知数据都需要传输给父节点，以便父节点将其加入缓存.

(2) 当局部 Skyline 节点的感知数据发生变化并且仍然为局部 Skyline 节点时，由于父节点存储的数据不再正确，因此它的节点编号和感知数据都需要传输给父节点，以便父节点可以替换缓存中的数据.

(3) 当局部 Skyline 节点的感知数据发生变化并且变为非局部 Skyline 节点时，只需要将它的节点编号告知父节点，使之将其移出缓存即可.

总体上说，传感器节点需要传输的信息共包括两个部分：一个是节点编号的集合，记为 S_{id}；另一个是(节点编号，感知数据)构成的集合，记为 S_t. 其中，集合 S_t 中的节点既可能是新的局部 Skyline 节点，也可能是感知数据发生变化的旧的局部 Skyline 节点.

传感器节点修改缓存、重新计算局部 Skyline 节点集合并且维护本地过滤器的具体过程如算法 1 所示. 首先，传感器节点合并所有子节点发送的更新信息，将其与新采集的感知数据合并得到新数据集(第 1～5 行)；其次，根据 S_{id} 集合中的信息，将其中包含的肯定不属于新的局部 Skyline 的传感器节点移出缓存(第 6 行)；接着，根据 S_t 集合中的信息，将新的局部 Skyline 节点的编号和感知数据插入缓存，同时更新旧的局部 Skyline 节点的感知数据(第 7 行)；然后，利用缓存中的信息计算新的本地 Skyline 节点集合(第 8 行)；再计算出新的局部 Skyline 与本地过滤器间的不同部分，生成发送到父节点的数据包(第 9～15 行)；最后，将新的局部 Skyline 设置为新的本地过滤器(第 16 行).

算法 1. 自底向上过滤算法.

输入：子节点消息集合 M，缓存 $Cache$，本地过滤器 $Filter$，本地的感知数据 r

输出：提交到父节点的数据包 M'

```
1. for each element m in M do //合并子节点数据
2.     tempID=tempID+m.S_id;
3.     tempSet=tempSet+m.S_t;
4. endfor
5. tempSet=tempSet+r;    //加入本地的感知数据
6. Cache.Delete(tempID);
        //删除不再属于 Skyline 的元组
7. Cache.Update(tempSet);
```

```
        //修改数据变化的元组,加入新增加的元组
8.  S=Skyline(Cache);
9.  for each element p in S do
10.     if p is not contained in Filter then
11.         M'.S_id=M'.S_id+{p.id};
12.     else
13.         M'.S_id=M'.S_id+{p};
14.     endif
15. endfor
16. Filter=S
17. return M'.
```

另外,当传感器网络规模扩大或者感知数据维数增加时,Skyline 结果的数量急剧增加,从而造成本地过滤器的缓存规模过大,无法完全保存在传感器节点的有限内存中.此时,每个传感器节点可以随机或者有选择地选取一个局部 Skyline 结果的子集作为本地过滤器,选择的元组数量由传感器节点存储能力决定.当传感器节点向父节点传输局部 Skyline 结果时,将被选中加入本地过滤器的元组打上标签,以便父节点可以将其缓存以便保证父节点与子节点之间的缓存同步.

4.2　自顶向下方式

如前所述,自顶向下的过滤方式主要包括感知数据建模、过滤器的设置和查询结果维护 3 个主要问题,下面逐一加以阐述.

4.2.1　感知数据建模

感知数据的建模过程就是求解每个传感器节点的感知数据分布满足的概率密度函数.概率密度函数的具体形式由应用环境决定.理论上说,任何形式的概率密度函数都是可行的,只是效率略微有所区别.

首先,由网络管理员指定描述感知数据分布的概率密度函数的具体形式;其次,传感器节点利用已有的机器学习算法计算模型中的参数,将所得的参数提交基站;最后,基站根据接收到的参数确定各传感器节点的感知数据模型,根据这些模型计算初始过滤器设置方案.在计算过程中,基站保存所有传感器节点的感知数据模型作为全局信息,以便保持基站与传感器节点之间的同步.

感知数据时刻发生着变化,感知数据模型也需随之动态变化.一般情况下,传感器节点需要及时地将感知数据模型的变化通知基站,以便基站能快速地调整过滤器设置方案.然而,当感知数据模型更新频率过高时,模型的维护过程将花费传感器节点巨大的通信代价.利用假设-检验方法可以很好地解决上述问题.如果新采集的感知数据落在超立方体的范围内,说明根据当前感知数据模型计算的过滤器有效,因而认为当前的感知数据模型有效;如果新采集的感知数据落在超立方体的范围外,则利用假设-检验方法来对模型的参数逐一进行检验.在检验过程中,一旦某个参数未通过检验,说明当前的感知数据模型失效,重新计算模型中的参数并且将新参数提交基站.通过上述方式,基站可以及时地根据新的感知数据模型对过滤器的设置方案进行动态调整,从而提高传感器网络的总体运行效率.

由于传感器节点通常都携带着多种不同类型的传感设备,因此,采用多元高斯分布对传感器节点采集的多维感知数据进行建模.多元高斯分布是一元高斯分布的扩展,具体函数如式(1)所示:

$$p(\boldsymbol{x})=\frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}\right|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu)\right) \tag{1}$$

其中,μ 是感知数据分布的均值;$\boldsymbol{\Sigma}$ 为感知数据的协方差矩阵,其对角线元素是对应维度的数据方差,而其非对角线元素是对应两个维度的数据协方差;模型中每维数据对应着传感器节点的一种传感设备.

为了便于对感知数据模型进行动态维护,每个传感器节点在保存提交给基站的数据模型 $f(\mu,\boldsymbol{\Sigma})$ 的同时,时刻计算并更新着感知数据的重要统计信息,如均值 $\overline{X}$ 和协方差矩阵 $\boldsymbol{S}$ 等.当新采集的感知数据没有通过过滤器时,认为感知数据模型有效,不需要进行任何处理;一旦新采集的感知数据通过了过滤器,则认为感知数据模型可能会失效,对感知数据模型进行假设-检验.在假设-检验过程中,首先进行均值的检验,检验方程如式(2)所示:

$$T^2=n(\overline{X}-\mu)'\hat{\boldsymbol{\Sigma}}^{-1}(\overline{X}-\mu),\ \hat{\boldsymbol{\Sigma}}=\frac{\boldsymbol{S}}{n-1} \tag{2}$$

一旦 T^2 的数值大于指定的阈值,说明均值 μ 检验失败,需要重新计算感知数据模型并将新模型的参数提交基站.如果通过了均值检验,则继续进行方差检验,检验方程如式(3)所示:

$$L=(n-1)[\ln|\boldsymbol{\Sigma}|-p-\ln|\hat{\boldsymbol{\Sigma}}|+tr(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1})],\hat{\boldsymbol{\Sigma}}=\frac{\boldsymbol{S}}{n-1} \tag{3}$$

同样,当方差 $\boldsymbol{\Sigma}$ 检验失败时,需要重新计算感知数据模型并将新模型的参数提交基站.基站可以根据新模型对过滤器的设置方案进行动态调整.

通过假设-检验可以有效地避免感知数据模型的频繁更新,节约了传感器节点巨大的通信代价.同

时,假设-检验的时间复杂度是 $o(d^2)$,与巨大的通信代价相比,计算过程只消耗较少的能量,可以忽略不计.

4.2.2 过滤器的设置

当所有传感器节点的感知数据均未通过本地过滤器的过滤时,基站不需要从网络中获取任何辅助信息就可以根据过滤器设置方案 $FS=\{h_i \mid i=1,2,\cdots,|N|\}$ 准确地计算出网络中的 Skyline 节点集合,那么该方案就是一个合理的过滤器设置方案.不失一般性,将传感器节点分为候选 Skyline 节点和候选非 Skyline 节点两类,分别用 N_S 和 N_N 表示.显然,可行的过滤器设置方案需要满足 $\{n_i \mid h_i \in \text{Skyline}(FS)\}=N_S$.

定理 3. 设 $H=\{h_1,h_2,\cdots,h_n\}$ 是超立方体的集合,H_S 是 H 的一个子集.如果 H_S 中的任意两个超立方体 h_i 和 h_j 满足 $h_i<>h_j$,同时对 $H-H_S$ 中任何超立方 h_k,H_S 中都存在一个 h_l 满足 $h_l \geq h_k$,那么,$H_S=\text{Skyline}(H)$.

证明.

(1) $\text{Skyline}(H) \subseteq H_S$:

根据已知条件可知,$H-H_S$ 中任何超立方 h_k 都不属于 $\text{Skyline}(H)$,所以可得 $\text{Skyline}(H) \subseteq H_S$.

(2) $H_S \subseteq \text{Skyline}(H)$:

根据已知条件,H_S 中的任意两个超立方体 h_i 和 h_j 都满足 $h_i<>h_j$,可知 H_S 中的任何一个超立方体 h_i 都属于 $\text{Skyline}(H)$,所以 $H_S \subseteq \text{Skyline}(H)$.

综上所述,$H_S=\text{Skyline}(H)$. 证毕.

根据定理 3,一个可行的过滤器设置方案至少需要满足以下两个条件:

(1) N_S 集合内的传感器节点的超立方体过滤器之间互不支配,即 $\forall n_i,n_j \in N_S, h_i<>h_j$.

(2) N_N 集合内的传感器节点的超立方体过滤器需至少被 N_S 集合内的一个传感器节点的超立方体过滤器所支配,即 $\forall n_j \in N_N, \exists n_i \in N_S, h_i \geq h_j$.

如果过滤器设置方案满足上述两个条件,那么当传感器节点采集的感知数据都在各自的超立方体过滤器内变化时,N_S 集合中所有的传感器节点共同构成了网络中的 Skyline 节点集合.

图 3 中的候选 Skyline 节点集合 $N_S=\{n_2,n_3\}$;候选非轮廓节点集合 $N_N=\{n_1,n_4,n_5\}$,传感器节点设置的过滤器满足上述两个条件.在时刻 1 和时刻 2,由于没有任何感知数据的更新消息提交基站,因而 Skyline 节点集合始终为 $\{n_2,n_3\}$.

可行的过滤器设置方案可以保证当所有传感器节点的感知数据都在各自的超立方体过滤器内变化时,不需要在网络中传输任何数据信息就能准确地计算出 Skyline 节点集合.因此,可以认为这种情况出现的概率越高,过滤器设置方案的过滤效果越好,如式(4)所示:

$$\text{Maximize} \prod_{i=1}^{n} P(r_i \in h_i) \tag{4}$$

满足:

(1) $\forall n_i,n_j \in N_S, h_i<>h_j$.

(2) $\forall n_j \in N_N, \exists n_i \in N_S, h_i \geq h_j$.

将条件中的超立方体之间的关系转化为感知数据的关系,可以得到式(5):

$$\text{Maximize} \prod_{i=1}^{n} P(r_i \in h_i) \tag{5}$$

满足:

(1) $\forall n_i,n_j \in N_S, \exists k,l \in D, h_j.I_k.ub<h_i.I_k.lb$ 且 $h_j.I_l.lb>h_i.I_l.ub$.

(2) $\forall n_j \in N_N, \exists n_i \in N_S, \forall k \in D, h_i.I_k.ub<h_j.I_k.lb$.

过滤器设置方案在计算一个非线性约束的非线性优化问题时,计算的空间代价和时间代价都很高.为了适应传感器网络应用的需要,通过放宽上述两个约束条件来减少计算的复杂度.

首先,对于 N_S 中的任何两个传感器节点 n_i 和 n_j 及其对应的超立方体过滤器 h_i 和 h_j,如果指定两个维,使其保证互不支配关系,那么条件(1)就变成了线性约束.为了使过滤器的过滤效果最大化,选择满足式(6)的两维来保证 h_i 和 h_j 的互不支配关系:

$$\text{Maximize}(\bar{r}_i.v_k-\bar{r}_j.v_k)\times(\bar{r}_j.v_l-\bar{r}_i.v_l) \tag{6}$$

式中,$\bar{r}_i$ 是 r_i 的均值期望.

接着,对 N_N 中的传感器节点 n_i,如果指定 N_S 中一个节点 n_j,使得 h_j 确定支配 h_i,保证 n_i 确定不是 Skyline 节点,那么条件(2)也变成了线性约束.为了使过滤器的过滤效果最大化,选择满足式(7)的节点 n_j 使之能支配 n_i,同时称节点 n_j 为 n_i 的支配节点,节点 n_i 为 n_j 的被支配节点.

$$\text{Maximize} \prod_{k=1}^{d} (\bar{r}_i.v_k-\bar{r}_j.v_k) \tag{7}$$

经过上述两步简化,非线性约束的非线性优化问题转化为线性约束的非线性优化问题,可以利用粒子群优化算法[15]来解决.化简后的线性约束的非线性优化问题如式(8)所示:

$$\text{Maximize} \prod_{i=1}^{n} P(r_i \in h_i) \tag{8}$$

满足：

(1) $\forall n_i, n_j \in N_S$, $k, l \in D$, $h_j.I_k.ub < h_i.I_k.lb$ 且 $h_j.I_l.lb > h_i.I_l.ub$.

(2) $\forall n_j \in N_N$, $n_i \in N_S$, $\forall k \in D$, $h_i.I_k.ub < h_j.I_k.lb$.

当传感器节点的感知数据模型发生变化时，需要将新模型的参数提交基站，基站根据新模型相应地调整过滤器设置方案. 每次调整设置方案都需要重新计算所有传感器节点的超立方体过滤器并将它们下发到相应的传感器节点，过滤器的维护过程将导致巨大的通信代价，显然是不合算的. 因此，FIST 算法中考虑只调整受影响较大的一部分传感器节点的过滤器，其余节点的过滤器保持不变. 设感知数据模型发生变化的节点为 n_i，那么传感器节点过滤器的调整方案可以分为 4 种情况：

(1) n_i 是 Skyline 节点，模型变化后还是 Skyline 节点. 除了 n_i 本身的过滤器 h_i 之外，其原来的被支配节点和新的被支配节点的过滤器也需要调整.

(2) n_i 是 Skyline 节点，模型变化后为非 Skyline 节点. 除了 n_i 本身的过滤器 h_i 之外，其原来的被支配节点、新的支配节点和新支配节点的被支配节点的过滤器都需要调整.

(3) n_i 是非 Skyline 节点，模型变化后还是非 Skyline 节点. 除了 n_i 本身的过滤器 h_i 之外，其原来的支配节点和新的支配节点的过滤器都需要调整.

(4) n_i 是非 Skyline 节点，模型变化后为 Skyline 节点. 除了 n_i 本身的过滤器 h_i 之外，所有 Skyline 节点的过滤器都需要调整.

经过这样的处理，过滤器维护过程的计算代价显著降低；同时，由于只有少数传感器节点的过滤器发生了变化，过滤器维护过程中的通信代价也随之降低.

4.2.3　查询结果维护

为传感器节点设置了过滤器后，传感器节点一旦发现某一时刻的感知数据超出了过滤器的范围，就将数据的变化即刻提交基站. 在同一个时刻，可能有多个节点同时向基站提交感知数据的更新信息，在树型路由环境下，可以采用网内计算[11]的办法来减少网络中的数据传输量. 基站收到所有数据更新信息后，用完整的感知数据元组替换过滤器设置方案中对应的超立方体，得到由感知数据元组和超立方体共同组成的混合集；接着，将元组看作是超立方体的特例，求出该混合集的 Skyline 结果集 H_S. 根据定理 2，一旦 H_S 中的超立方体 h_i 与 H_S 中的其它任何超立方体 $h_j (j \neq i)$ 之间都满足互不支配关系，那么相应的传感器节点 n_i 将肯定为 Skyline 节点. 如果超立方体 h_i 不满足上述条件，基站将不能确定节点 n_i 是否为 Skyline 节点，此时，基站需要获取相关节点的感知数据重新计算 Skyline 节点集合. 涉及的关键问题包括中间节点的网内计算算法和基站的查询结果维护算法.

通过在中间节点判断感知数据是否被其它节点的感知数据所支配，可以避免一些不必要的数据传输. 中间节点的具体计算过程如算法 2 所示. 首先，传感器节点合并所有子节点发送的更新信息，将其与新采集的感知数据合并得到新数据集(第 1～5 行)；接着，计算该数据集的局部 Skyline 结果(第 6 行)；如果本地的感知数据未通过过滤器，则将其从 Skyline 结果中移除(第 7～10 行)；最后，将不属于 Skyline 结果且通过了过滤器的传感器节点的编号加入 S_{id}(第 11～15 行).

算法 2.　自顶向下过滤算法(节点).

输入：子节点消息集合 M，本地的感知数据 r

输出：提交到父节点的数据包 M'

```
1. for each element m in M do    //合并子节点数据
2.       M'.S_id = tempID + m.S_id ;
3.       tempSet = tempSet + m.S_t ;
4. endfor
5. tempSet = tempSet + r;    //加入自己新采集的数据
6. M'.S_id = Skyline(tempSet);
7. if passFilter(r) == false then //过滤感知数据
8.       tempSet = tempest - r;
9.       M'.S_id = M'.S'_id - r;
10. endif
11. for each element t in tempSet do //生成数据包
12.      if t.id is not contained in M'.S_t
13.         M'.S_id = M'.S'_id + {t.id};
14.      endif
15. endfor
16. return M'.
```

基站每个时刻对 Skyline 节点查询结果进行维护的具体过程如算法 3 所示. 首先，将编号出现在 S_{id} 中的节点从保存的过滤器集合中移除(第 1～3 行)；其次，用 S_t 中完整的感知数据元组替换过滤器集合中对应的超立方体得到混合集(第 4～6 行)；然后，计算该混合集的 Skyline(第 7 行)；接着，找到所有不确定是否为 Skyline 节点的传感器节点，从这些节点获取必要的感知数据(第 8～16 行)；最后，根据获取的感知数据计算该时刻的 Skyline 节点查询结果(第 17 行).

算法 3. 自顶向下过滤算法(基站).

输入:节点的感知数据更新消息 M;基站保存的过滤器集合副本 H

输出:Skyline 节点集合 S

1. for each element i in $M.S_{id}$ do //移除非 Skyline 节点
2. $H=H-h_i$;
3. endfor
4. for each element t in $M.S_t$ do
//替换有数据更新的节点
5. replace h_i in H with t;
6. endfor
7. $S=$Skyline(H); //计算过滤器中的 Skyline
8. for each element p in S do
//计算不确定是否属于 Skyline 的节点
9. for each element q in S do
10. if $p \neq q$ 且 q ? q then
11. $S_{\text{probe}}=S_{\text{probe}}+\{p.id,q.id\}$;
12. endif
13. endfor
14. endfor
15. probe the nodes appear in S_{probe};
16. replace the hypercube in S with the new data;
17. $S=$Skyline(S);
18. return S.

4.3 混合过滤方式

自底向上和自顶向下的过滤器方式各有利弊.在自底向上的过滤方式中,一旦感知数据发生了变化,节点马上将其提交基站,基站可以直接得到 Skyline 节点查询结果.因此,该种方式的查询响应时间较短,但是消耗较多的能量.在自顶向下的过滤方式中,基站只有在获取了相关传感器节点的感知数据后,才能确定最终的 Skyline 节点查询结果,所以查询响应时间较长;而此时节点设置的过滤器由元组变成了超立方体,因而过滤范围变大,过滤效果也更好,消耗的能量一般也较少;然而,有时虽然传感器节点的感知数据没有超出超立方体的范围,也可能因为其它节点感知数据的变化使得该节点是否为 Skyline 节点变得不确定,从而需要获取该节点的感知数据,造成过滤效果的显著下降,将消耗节点较多能量.为了更好地利用两种过滤方式的优点,回避它们的缺点,提出了混合过滤的方式,通过为传感器节点选择"最合适"的过滤器达到减少节点能量消耗的目的.

如图 6 所示,传感器节点 n_2、n_3 和 n_5 的数据相对稳定,选择自底向上的方式过滤;而节点 n_1 和 n_4 的数据变化相对频繁,选择自顶向下的方式过滤.在每个时刻,节点 n_2、n_3 和 n_5 都时刻将数据变化报告基站,一共需要在网络中传输 4 个元组.在时刻 1 和时刻 2,节点 n_1 和 n_4 的数据均没有超出各自立方体过滤器的范围,因而不需要向基站提交任何数据更新消息;在时刻 3,节点 n_4 的数据超出了超立方体的范围,需要向基站提交数据更新消息,同时由于节点 n_3 和 n_5 的数据已知,因此不需要从网络中获取任何数据就可以得到网络中的 Skyline 节点集合.混合过滤方式在 3 个时刻一共需要在网络中传输 5 个元组,与上述两种过滤方式相比,效率有了进一步的提高.

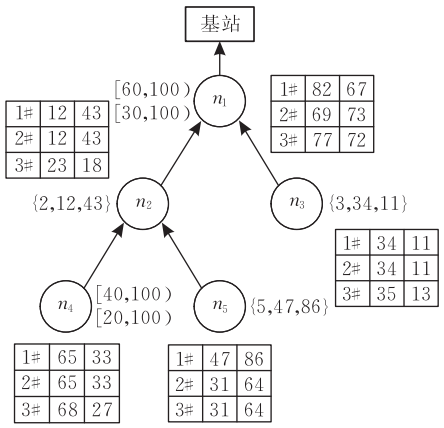


图 6 混合过滤器设置举例

基于上述原理,得出混合过滤方式的基本思想:将传感器节点分成两类,一类是数据变化频率较低的节点,使用本地过滤器(自底向上地过滤),这样既避免了数据的大量重复传输,也避免了节点被频繁访问;另一类是数据变化率较高的节点,使用全局过滤(自顶向下地过滤),避免了大量的不必要的数据更新.在介绍具体的传感器节点分类方法之前,先讨论两种过滤方式的过滤效果.设传感器节点 n_i 的数据变化率为 α_i,当节点 n_i 使用自底向上的过滤方式,也即是使用本地过滤器时,只有数据发生变化时才需要向基站传输消息.因此,节点 n_i 不需要传输消息的概率如式(9)所示,将此概率称为该方法的获益.

$$BenifitB(n_i)=1-\alpha_i \tag{9}$$

当传感器节点 n_i 使用自顶向下的过滤方式,即使用全局过滤器时,可以分为两种情况:

(1) 如果节点 n_i 是候选非 Skyline 节点,n_j 是支配它的 Skyline 节点,那么无论节点 n_j 采集的感知数据 r_j 是否在超立方体 h_j 内变化,只要节点 n_i 采集的数据 r_i 在超立方体 h_i 内变化且元组 r_j 支配 h_i,不需要知道 r_i 的具体数值就可以确定 n_i 不属于 Skyline 节点集合,所以 n_i 不需要向基站传输任何数据;否

则，r_i或因为通过了过滤，或因为与感知数据元组 r_j 的关系无法确定而需要被 n_i传输到基站. 此时，自顶向下的过滤方式的获益如式(10)所示.

$$BenifitT(n_i)=P(r_i\in h_i)\times P(r_j\geq h_i) \quad (10)$$

(2) 如果节点 n_i 是候选 Skyline 节点，候选 Skyline 节点集合为 N_S，那么只要 r_i在超立方体 h_i 内变化且 N_S中所有节点采集的感知数据都与 h_i互不支配，不需要知道 r_i的具体数值就可以确定 n_i属于 Skyline 节点集合，所以不需要向基站传输任何数据. 此时，自顶向下的过滤方式的获益如式(11)所示.

$$BenifitT(n_i)=P(r_i\in h_i)\times \prod_{n_j\in N_S(i\neq j)} P(r_j<>h_i) \quad (11)$$

显然，当 $BenifitB(n_i)\geq BenifitT(n_i)$时，传感器节点 n_i应该使用自底向上的过滤，此时称 n_i为 B 节点；而当 $BenifitB(n_i)<BenifitT(n_i)$时，传感器节点 n_i应该使用自顶向下的过滤，此时称 n_i为 T 节点.

混合过滤方式中过滤器的设置方案如下：首先，基站根据所有节点的感知数据模型计算初始的全局过滤器设置方案；然后，计算各节点分别使用自底向上和自顶向下的方式过滤时的获益，并根据上述的原则将传感器节点分组；最后，基站广播所有 T 节点的全局过滤器，没有收到全局过滤器的节点使用自底向上的方式过滤.

由于使用的是多跳的路由树结构，两组传感器节点往往需要互相转发消息，如果节点利用这些消息来过滤自己的数据将进一步减少网络中的通信代价. 由于 B 节点发送的消息里包含的是连续两个时刻 Skyline 的不同部分，只有被上一级 B 节点完全接收，才能保证两者数据之间的一致性. 一旦被某个 T 节点修改，那么这种一致性就将被破坏. 例如，B 节点 n_i发送的消息中的元组 t 被 T 节点 n_j所过滤，它的上一级 B 节点 n_k就不能收到元组 t. 在下一个时刻，如果 t 是全局 Skyline，那么它也一定属于节点 n_i的局部 Skyline，根据自底向上的过滤原理，节点 n_i不需要传输元组 t，而在上一级 B 节点 n_k的缓存中不包含 t，所以计算的局部 Skyline 中也就不包含 t，导致基站不能得到正确的 Skyline 结果. 因此，在混合过滤方式中，B 节点的消息只能在 B 节点中修改，而不能在 T 节点中修改. 在自顶向下的过滤方式中，上下级节点之间不存在同步的问题，节点只需要通知基站采集的感知数据是否通过过滤，若通过过滤则通知是否属于 Skyline. 当通过过滤的节点确定不属于 Skyline 时，只需要将其编号告知基站即可. 因此，T 节点发送的消息可以在 B 节点中修改，只需要保留被支配节点的编号. 混合过滤方式中，中间节点的计算过程与算法 1 和算法 2 类似，只需做少量修改即可. 基站接收到所有的数据更新后，将两组节点的感知数据合并成一个数据集，其余处理过程与算法 3 相同.

5　性能评价与分析

在本节中，分别利用人工合成数据集和真实数据集对各算法的性能进行了详细的比较和分析. 实验的硬件环境为 2.0 GHz DELL PC，512MB 内存，80GB 硬盘.

5.1　实验设置

在实验中，参与性能分析及比较的算法及其变型主要包括：

(1) TAG. 利用网内计算的轮廓监控算法(基准算法).

(2) MinMax. 文献[5]中提出的 Skyline 查询算法.

(3) SkySensor. 文献[10]中提出的 Skyline 查询算法.

(4) BF. 采用自底向上过滤方式的 FIST 算法.

(5) TF. 采用自顶向下过滤方式的 FIST 算法.

(6) HF. 采用混合过滤方式的 FIST 算法.

实验环境由 JAVA 语言实现，实验中随机地在面积为 n 个平方单位的区域内随机部署 n 个传感器节点，使得每个节点所占的平均面积为 1 个平方单位. 同时将传感器节点之间的通信半径设置为 $2\sqrt{2}$ 个单位长度，并且规定传感器节点可以发送的最大数据包长度为 48 字节. 测试数据被均匀地分布在 n 个传感器节点上，且每个时刻，每个节点采集一个新的感知数据，因此，整个传感器网络中每个时刻将产生 n 个新的元组. 表 1 列出了实验中所要考察的主要参数及其变化范围和默认值. 每次实验变化其中一种参数，而将其余参数设置为默认值. 考察的主要性能指标为传感器网络中的平均通信代价和查询的平均响应时间.

(1) 平均通信代价. 平均每个时刻的通信代价包括过滤器维护代价和 Skyline 数据的传输代价.

(2) 平均响应时间. 将 TAG 算法执行一次网内计算过程的时间设为 1 个时间单位，其余算法计

算与它的相对数值，SkySensor 算法的响应时间为结果返回跳数与 TAG 算法路由树深度的商.

表 1 实验参数

参 数	默认值	变化范围
数据维数	3	2,3,4,5
节点数目	300	100,200,300,400,500
数据变化率	85%	75%,80%,85%,90%,95%
分布协方差	σ	σ,2σ,3σ,4σ,5σ

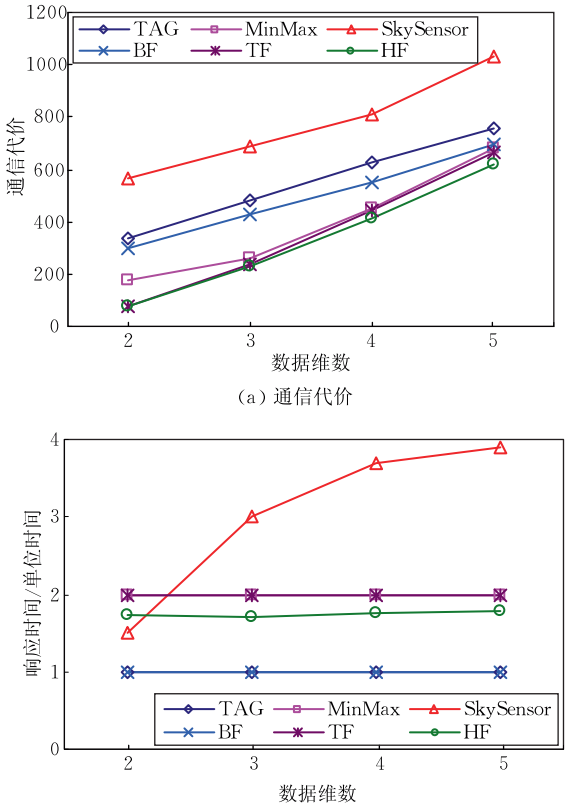
5.2 合成数据测试

实验中的合成数据生成过程如下：首先，利用标准的数据集生成工具[3]分别生成一则独立分布和反相关分布的数据；接着，以这些数据作为均值利用多元高斯分布生成器产生$[0,1]^d$的合成数据.

(1) 独立分布的性能

图 7 表明，随着数据维数的增加，各算法的通信代价也随之增加. 原因在于数据维数的增加使得节点被支配的概率降低，导致了 Skyline 节点数量的增加. 由于数据变化率在数据维数变化时保持恒定，因此 BF 算法的通信代价相对于 TAG 算法减少的比例相对稳定. MinMax 中的元组过滤器在独立分布下有很好的过滤效果，因此优于 BF 算法. SkySensor 将数据传输到聚簇的代价不再被多个查询所分担，因而传输代价最高. 维数的增加使得数据超出超立方体范围的概率增加，不仅使得网内计算过程中的数据传输量增加，也使得全局过滤器的计算和维护代价增加，因此 TF 和 HF 的通信代价都随着维数的增加而急剧增加. BF 与 TAG 都利用网内计算，响应时间为 1 个时间单位；MinMax 是两阶段算法，响应时间为 2 个时间单位. SkySensor 需要按顺序访问各聚簇，因而响应时间随着维数的增加而增加(聚簇数等于维数). TF 经常需要到网络中获取数据来确认 Skyline，响应时间始终为 2 个时间单位；而 HF 中，大部分对结果影响较大的数据已经利用自底向上的方法获取，与 TF 相比响应时间显著缩短.

图 8 表明，随着传感器节点数目的增加，各算法的通信代价相应增加. TAG、BF、MinMax 和 SkySensor 的通信代价增长较快，BF 与 TAG 的比例比较稳定，TF 和 HF 增长相对缓慢. 原因在于节点数目的增加造成参与计算元组数量的增加，也导致了 Skyline 结果数量的增加；同时节点数目的增加也造成了平均路由跳数的增加，所以各算法的通信代价增加. BF 的性能只依赖于数据变化率，它与 TAG 通信代价的比例相对稳定. 因为节点数目的增加对超立方体的过滤能力没有产生直接影响，所以



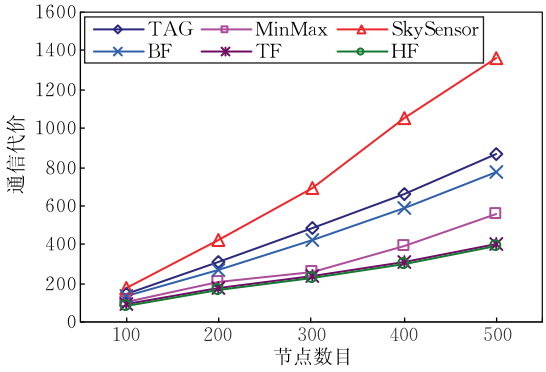
(a) 通信代价

(b) 响应时间

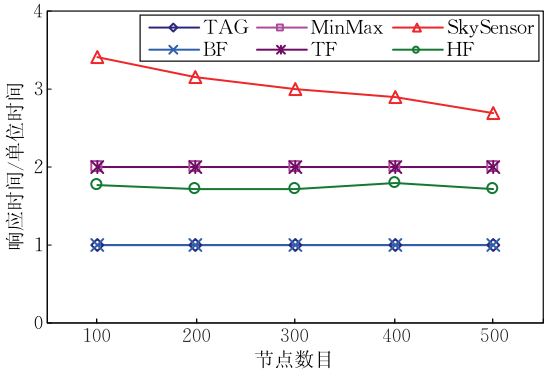
图 7 数据维数的影响

TF 和 HF 的通信代价增长相对缓慢. 由于作为参照的路由树深度增加，SkySensor 的响应时间随着节点数目的增加稍微有所减少，除 SkySensor 外各算法在响应时间方面的表现与数据维数变化时一致.

图 9 表明，随着数据变化率的增加，TAG、MinMax、SkySensor、TF 和 HF 算法的通信代价比较稳定，BF 算法的通信代价成比例增长. 因为 BF 算法的过滤原理依赖于数据变化率，一旦连续时刻数据相同的概率降低，那么过滤效果也随之降低，所以通信代价随着数据变化率的增加成比例增加. TAG 未使用任何过滤措施，通信代价只和参与计算元组数量及维数等相关，与连续时刻的数据是否变化无关；MinMax 和 SkySensor 与连续时刻的数据是否变化无关；TF 和 HF 中使用的超立方体的过滤效果只和数据分布有关，与前后时刻的数据是否相同也无关. 因此，TAG、MinMax、SkySensor、TF 和 HF 算法的通信代价比较稳定. 在响应时间方面的表现依旧相同，TAG 和 BF 的响应时间始终为 1 个时间单位，SkySensor 的响应时间接近 3 个时间单位，MinMax 和 TF 的响应时间始终为 2 个时间单位，HF 的响应时间介于 BF 和 TF 之间.

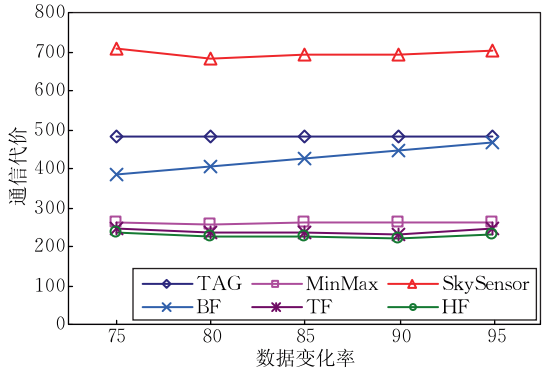


(a) 通信代价

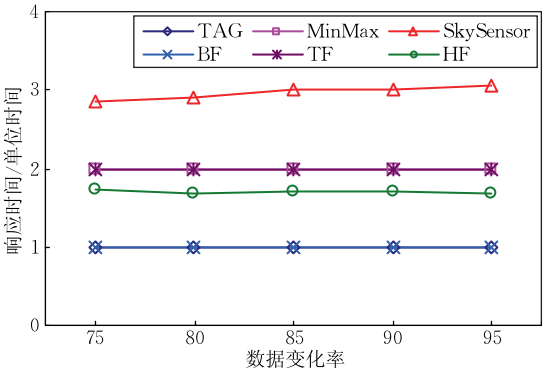


(b) 响应时间

图 8　节点数目的影响



(a) 通信代价



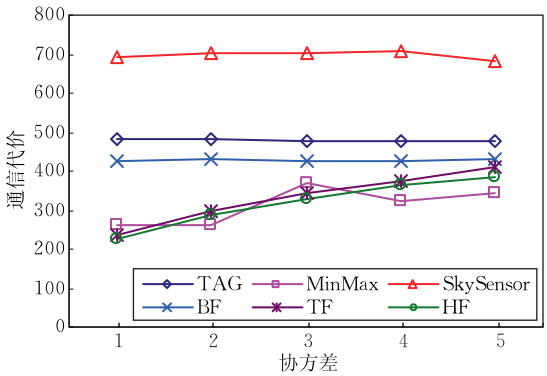
(b) 响应时间

图 9　数据变化率的影响

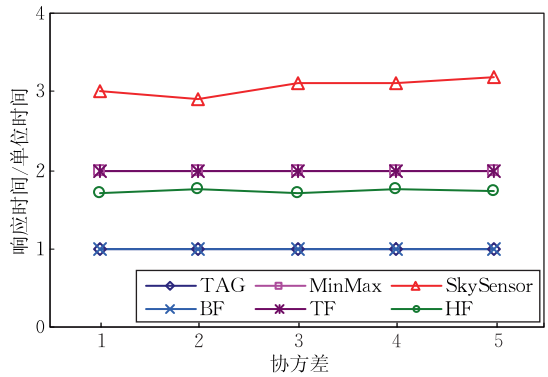
在节点感知数据分布对算法影响的测试中，不失一般性地，假设感知数据模型协方差矩阵中的各元素均相同，将整个数据空间划分为 n 份，平均每份空间的半径设为 σ. 图 10 说明了各算法在数据分布协方差变化时的性能变化规律. TAG、MinMax、SkySensor 和 BF 算法不依赖于数据的分布，因而通信代价相对比较稳定. 随着协方差的增加，传感器节点感知数据的分布区域也随之扩大，也就导致了感知数据超过超立方体范围的概率增加，随之而来的是大量的查询重计算和过滤器的频繁更新，导致 TF 算法的通信代价显著地增加. HF 算法的通信代价始终低于 TF 算法，且随着 TF 算法的变化趋势而变化. 在响应时间方面，各个算法的表现与前面测试的结果保持一致，进一步说明了响应时间与仿真参数的变化无关.

(2) 反相关分布的性能

为了更好地说明算法的性能，在反相关分布下也进行了算法性能的测试. 如图 11 所示，TAG、SkySensor、BF、TF 和 HF 算法在反相关分布下的表现与独立分布数据的性能表现基本一致，只有



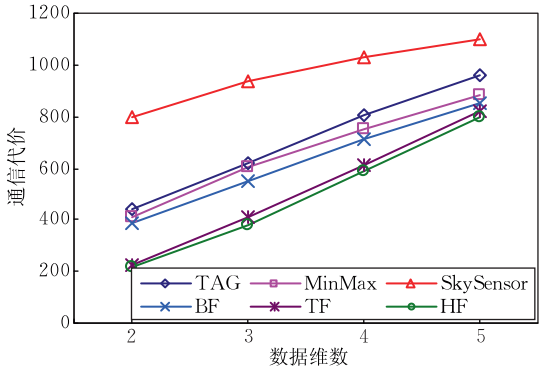
(a) 通信代价



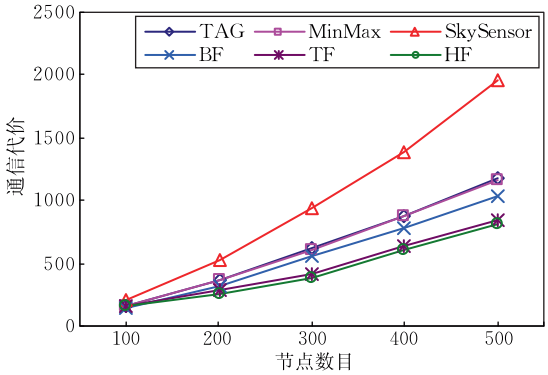
(b) 响应时间

图 10　数据协方差的影响

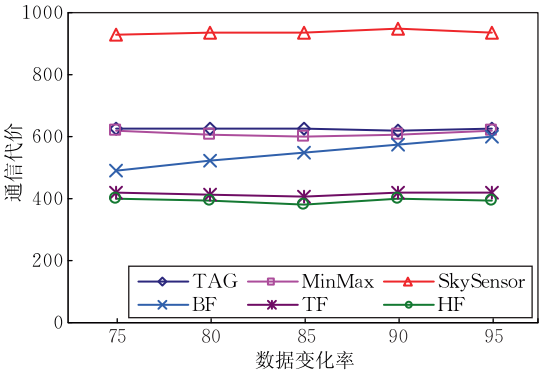
MinMax 算法的性能变得较差，原因在于反相关分布时元组过滤器的过滤效果下降. 同时，HF 算法的性能始终优于其它的算法，进一步证明了算法的有效性.



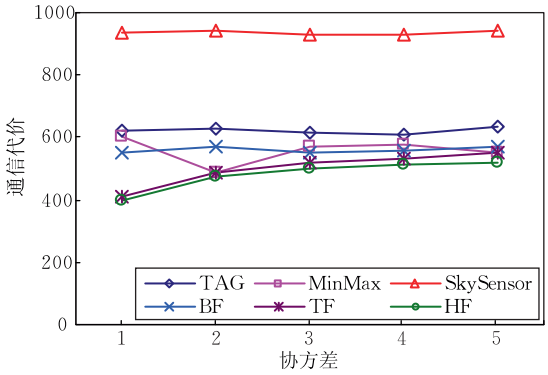
(a) 数据维数的影响



(b) 节点数目的影响



(c) 数据变化率的影响



(d) 数据分布的影响

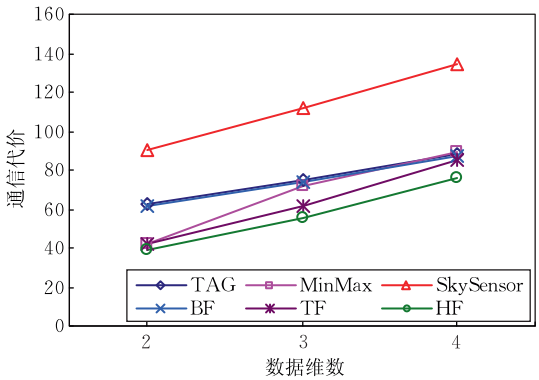
图 11 反相关分布的性能分析

5.3 真实数据测试

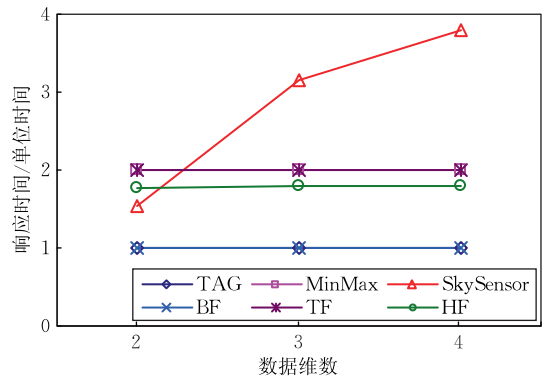
实验中的真实数据集选择了热带海洋与大气项目(Tropical Atmosphere Ocean，TAO)产生数据中的动力高度、等温面深、相对湿度和海面气象等 4 维数据来进行测试. 在测试中，分别对数据维数为 2、3 和 4 时各算法的有效性进行了验证. 图 12 表明，TAG 算法的通信代价始终最高，BF 算法的通信代价始终低于 TAG 的通信代价，证明了真实感知数据依然具有很强的时间相关性. TF 和 HF 比 BF 算法节约了更多的通信代价. 同时，在响应时间方面的结论与合成数据时的结论保持一致.

5.4 实验总结

综上所述，TF 和 HF 算法在通信代价上的表现远远优于其它算法；而 HF 算法在响应时间方面优于 TF 算法. 由于传感器节点携带的能量有限，通信代价是衡量算法性能的最重要指标之一，因此可以得出如下结论：HF 算法是计算及维护 Skyline 节点查询综合性能最优的算法.



(a) 通信代价



(b) 响应时间

图 12 真实数据的性能分析

6 结　论

深入地研究了传感器网络中的 Skyline 节点连续查询处理技术，提出了基于过滤的 Skyline 节点连续查询算法 FIST，利用设置在传感器节点的本地或者全局过滤器来避免不必要的数据传输，进而节约传感器节点的能量. 根据过滤器设置方式的不同，FIST 算法分为自底向上(BF)、自顶向下(TF)和混合(HF) 3 种不同的过滤方法. TF 利用前一时刻的局部 Skyline 作为本地过滤器来避免数据的重复传输；TF 利用感知数据的统计规律为节点分配超立方体作为过滤器来避免不必要的数据更新. HF 综合了 BF 与 TF 的优势，通过将节点划分为两大类来充分发挥两种过滤器算法的优势，一类节点使用本地过滤器，另一类使用全局过滤器. 最后，通过合成数据和真实数据对算法的有效性进行了验证. 结果表明，FIST 算法极大地减少了 Skyline 节点连续查询过程中传感器网络内的数据传输量.

参 考 文 献

[1] Ren Feng-Yuan, Huang Hai-Ning, Lin Chuang. Wireless sensor networks. Journal of Software, 2003, 14(2): 1282-1291(in Chinese)
(任丰原，黄海宁，林闯. 无线传感器网络. 软件学报，2003，14(2)：1282-1291)

[2] Li Jian-Zhong, Li Jin-Bao, Shi Sheng-Fei. Concepts, issues and advance of sensor networks and data management of sensor networks. Journal of Software, 2003, 14(10): 1717-1727(in Chinese)
(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. 软件学报，2003，14(10)：1717-1727)

[3] Borzsonyi S, Stocker K, Kossmann D. The Skyline operator//Proceedings of the 17th International Conference on Data Engineering. Heidelberg, Germany, 2001: 421-430

[4] Wei Xiao-Juan, Yang Jing, Li Cui-Ping, Chen Hong. Skyline query processing. Journal of Software, 2008, 19(6): 1386-1400(in Chinese)
(魏小娟，杨婧，李翠平，陈红. Skyline 查询处理. 软件学报，2008，19(6)：1386-1400)

[5] Chen H, Zhou S, Guan J. Towards energy-efficient Skyline monitoring in wireless sensor networks//Proceedings of the 4th European Conference on Wireless Sensor Networks. Delft, Netherlands, 2007: 101-116

[6] Xin J, Wang G, Chen L, Zhang X, Wang Z. Continuously maintaining sliding window Skylines in a sensor network//Proceedings of the 12th International Conference on Database Systems for Advanced Applications. Bangkok, Thailand, 2007: 509-521

[7] Xin J, Wang G, Zhang X. Energy-efficient Skyline queries over sensor network using mapped Skyline filters//Proceedings of the Joint 9th Asia-Pacific Web Conference and 8th International Conference on Web-Age Information Management. Huangshan, China, 2007: 144-156

[8] Xin J, Wang G, Chen L, Oria V. Energy-efficient evaluation of multiple Skyline queries over a wireless sensor network//Proceedings of the 14th International Conference on Database Systems for Advanced Applications. Brisbane, Australia, 2009: 247-262

[9] Shen H, Chen Z, Deng X. Location-based Skyline queries in wireless sensor networks//Proceedings of the 1st International Conference on Networks Security, Wireless Communications and Trusted Computing. Wuhan, China, 2009: 391-395

[10] Su I F, Chung Y C, Lee C, Lin Y Y. Efficient Skyline query processing in wireless sensor networks. Journal of Parallel and Distributed Computing, 2010, 70(6): 680-698

[11] Madden S, Franklin M J, Hellerstein J M, Hong W. TAG: A tiny aggregation service for ad-hoc sensor networks//Proceedings of the 5th Symposium on Operating System Design and Implementation. Boston, Massachusetts, USA, 2002: 131-146

[12] Silberstein A, Braynard R, Ellis C S, Munagala K, Yang J. A sampling-based approach to optimizing top-k queries in sensor networks//Proceedings of the 22nd International Conference on Data Engineering. Atlanta, GA, USA, 2006: 68-78

[13] Bentley J L, Kung H T, Schkolnick M, Thompson C D. On the average number of maxima in a set of vectors and applications. Journal of the ACM, 1978, 25(4): 536-543

[14] Wu M, Xu J, Tang X, Lee W-C. Top-k monitoring in wireless sensor networks. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(7): 962-976

[15] Kennedy J, Eberhart R. Particle swarm optimization//Proceedings of the International Conference on Neural Networks. Perth, Australia, 1995: 1942-1948

XIN Jun-Chang, born in 1977, Ph. D., lecturer. His research interests include sensor data management, cloud computing and uncertain data management.

WANG Guo-Ren, born in 1966, professor, Ph. D. supervisor. His research interests include uncertain data management, data-intensive computing, visual media data management and analysis, distributed query processing and optimization techniques, and bioinformatics.

Background

Wirelss sensor networks, which integrate sensor technology, embedded computing, networks, wireless communication and distributed information processing, are widely used in many fields. Since sensor nodes are generally battery powered, and most wirelss sensor networks work in an unattended, hard-to-reach environment, it is impossible or at least very difficult to change their batteries. Moreover, due to the unreliable sensing devices and signal interferences during wireless communications, sensing data are often noisy or missed. Therefore, the applications over wireless sensor networks need a scalable, energy-efficient and fault-tolerant method to gather the data collected by sensors.

As an important operator for multiple criteria decision making, Skyline node query plays an important role in many sensing applications, such as environment monitoring, habitat monitoring, and building structural monitoring. And most of the previous techniques proposed for Skyline queries assume that all the data are collected in a central server and Skyline queries are conducted in the collected data. However, since wireless communication is the major energy consumption factor of a wireless sensor network and the energy of the wireless sensor network is limited, it is impractical to collect all the sensing data to the sink and compute Skylines at the sink. The utility of Skyline queries in many real applications and the unsuitability of the previous proposed approaches motivate us to develop an energy-efficient algorithm to fulfill Skyline node monitoring queries over wireless sensor networks.

Therefore, in this paper, a comprehensive study on Skyline node query is performed, and a novel approach, called FIlter-based Skyline node moniToring (FIST), is proposed to evaluate a continuous Skyline node query in wireless sensor networks. Firstly, two types of filters including local filter and global filter are introduced, and their effectiveness of suppressing the unnecessary sensor updates are analyzed accordingly. The basic idea of the local filer for a sensor node is to cache the Skyline results of the sensor node at the last sampling round and use it to suppress the unnecessary data transmissions. It is obvious that the local filter of a sensor node becomes inefficient when the readings of the sensor node are not stable. A natural idea of global filter is to install a hypercube for such kind of sensor node to suppress unnecessary updates. Then, three Skyline node monitoring mechanisms are proposed based on the above two types of filters, including bottom-up filtering, top-down filtering and hybrid filtering. Last but not the least, our extensive simulation studies show that the proposed FIST approaches significantly reduce the unnecessary message transmissions and save the energy consumption during the evaluation of Skyline monitoring queries over wireless sensor networks.

This research is supported by the State Key Program of National Natural Science of China (Grant No. 60933001), the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61025007), the National Natural Science Foundation for Young Scientists of China (Grant No. 61100022), and the Fundamental Research Funds for the Central Universities (Grant No. N110404009).