# 在传感器网络中构造延迟限定的最大化生命周期树

梁俊斌[1,2],王建新[1],陈建二[1]
(1.中南大学信息科学与工程学院,湖南长沙 410083; 2.广西大学计算机与电子信息学院,广西南宁 530004)

**摘 要:** 在一些对延迟敏感的持续性监视应用中,无线传感器网络中的数据收集需要构造延迟限定的最大化生命周期树,这属于 NP 完全问题.提出一个新的算法 MILD,通过限定树的高度来满足延迟限定,然后通过使树上"瓶颈节点"的度最小化来延长树的生命周期.实验表明,与目前已有的协议相比,MILD 能有效地限定延迟并延长树的生命周期.

**关键词:** 无线传感器网络; 数据收集; 最大化生命周期; 延迟限定; 生成树
**中图分类号:** TP301 **文献标识码:** A **文章编号:** 0372-2112 (2010) 02-0345-07

# On the Construction of a Delay-Constrained Maximum Lifetime Tree in Wireless Sensor Networks

LIANG Jun-bin[1,2], WANG Jian-xin[1], CHEN Jian-er[1]
(1. *School of Information Science and Engineering, Central South University, Changsha, Hunan* 410083, *China*;
2. *School of Computer and Electronic Information, Guangxi University, Nanning, Guangxi* 530004, *China*)

**Abstract:** In some delay-sensitive and durative surveillance applications, a tree that can satisfy user's requirements of maximizing the network lifetime and constraining the data gathering delay needs to be constructed in wireless sensor networks. The problem of constructing the tree is NP-complete. A novel algorithm, named MILD, is proposed to solve this problem. MILD satisfies user's requirement for the delay by limiting the tree's height, and it extends the tree lifetime by decreasing the degree of bottleneck nodes to the minimum. Simulation results show that MILD can construct a tree that has longer lifetime than previous protocols under constrained delay.

**Key words:** wireless sensor networks; data gathering; maximum lifetime; delay constrained; spanning tree

## 1 引言

综合了无线通信技术、传感器技术和嵌入式计算技术的无线传感器网络(wireless sensor networks),是目前国际上前沿热点的研究领域.传感器节点能够协作地实时监测、感知网络区域内各种信息,然后以多跳的方式将这些信息传送给远方的基站(Sink).无线传感器节点具有数量庞大、通信和计算能力弱、电源能量有限且无法补充等特点.节点在工作时会在感知、计算和通信 3 个方面消耗能量,其中通信(发送和接收数据)所占的比重最大,而感知和计算的能耗可以忽略不计[1,2].

在一些持续性的监视应用中,如:森林火警监视、矿道瓦斯监测、战场监控等,不仅要求网络能保存能量长期地工作,还要求网络在收集到数据后能尽快地将数据传送给用户进行处理,即网络需要满足最大化生命周期和最小延迟这两个要求.本文首先通过对网络生成树的拓扑结构和节点的能量消耗模型进行分析,证明了以上两个要求是相互矛盾的,很难同时实现.然后,提出一个新的基于树的数据收集算法 MILD(an algorithm for constructing Maximum lifetIme data gathering tree while Limiting Delay)来有效平衡这两个目标.

## 2 相关工作

目前,有大量基于树的数据收集协议被提出.PEDAP-PA[3]从一棵只包含 Sink 节点的树出发,以树外节点和树上节点间的通信代价作为权值,不断挑选权值最小的节点加入树,直至网络中所有节点加入树.MLDGA[4]和 MNL[5]是对 PEDAP-PA 的改进,设计了更合

收稿日期:2008-12-06;修回日期:2009-10-17
基金项目:国家 973 重点基础研究发展规划(No.2008CB317107);国家自然科学基金(No.60673164;No.60873265);国家教育部创新团队资助计划(No.IRT0661)

理的权值来挑选节点加入树.然而,PEDAP-PA、MLDGA 和 MNL 在构造生成树的过程中,树外节点是作为叶子节点被考虑并加入树的,在树没有最终建立前无法确定一个节点最终在树上拥有多少孩子.因此,它们的生成树往往存在节点负载不均衡、树的高度无法控制等情况.

IAA[6]从一棵任意树出发,迭代地选择一条能使"瓶颈节点"包含在一个圈中的任意边加入树,然后删除该"瓶颈节点"在圈中关联的任意一条边以使"瓶颈节点"的度减少 1 并打破圈.IAA 能做到生成树上节点负载均衡,树的生命周期达到近似最优.但是,树的高度也无法控制.

根据我们目前掌握的情况,DB-MDST[7]是目前唯一通过限定树的高度来限定延迟的工作.它利用与 IAA 类似地"加入 - 删除"边的操作来优化树的生命周期.但是,DB-MDST 算法存在以下缺陷:(1)算法不考虑节点的能量,只优化树中度最大的节点,无法做到负载均衡;(2)对度最大的节点进行优化时,无论是选择加入树的边还是删除度最大的节点所关联的边都是任意的.这样会导致树高增长很快,算法很容易受树高限制而过早结束.

在本文,我们对 DB-MDST 进行改进,提出一个能在限定的树高下,根据节点能量水平来优化节点负载的算法 MILD.理论分析和实验表明,MILD 能在同样的延迟限定下有效地延长树生命周期.

## 3 网络模型和问题描述

### 3.1 网络模型

$n$ 个传感器节点随机地分布在一个面积为 $M \times M$ 平方米的正方形区域 $A$ 内,存在 1 个 Sink 节点.整个传感器网络组成一个连通的无向图 $G(V,E)$,其中 $V$ 是节点集合,$V=\{v_0,v_1,v_2,\cdots,v_n\}$,$|V|=n+1$,其中 $v_0$ 是 Sink 节点,$v_1,v_2,\cdots,v_n$ 是传感器节点;$E$ 是 $G$ 中边的集合,如果两个传感器节点 $v_i$ 和 $v_j$ 相互处于对方的通信半径内,则$(v_i,v_j)\in E$.$|E|=m$ 为边的数量.为了使算法具有可扩展性,不需要节点知道自己的位置信息,只要求节点知道自己的邻居信息,这可以很容易地通过相互交换 1 个"Hello"消息来实现.此外,网络具有如下性质:

(1)网络是连通的静态网络,节点部署后不再移动;

(2)传感器节点的种类可以有多种,它们的初始能量是异构的,且不能补充.

与 IAA 和 DB-MDST 一样,为了便于问题的分析,假定节点采用固定发射和接收功率,每个节点发射 1bit 数据的能耗为 $E_{tx}$,接收 1bit 数据的能耗为 $E_{rx}$.该假定可以很容易扩展到发射功率可变的情况,问题的性质不会改变.

### 3.2 相关定义

在本文研究的数据收集网络中,节点能对数据进行完全聚合(Fully aggregate)[19],即:在一轮数据收集中,树上的每个节点会接收其孩子发来的多个大小为 $k$ bits 的数据包,与自己产生的 $k$ bits 数据进行聚合,然后发送一个大小也为 $k$ bits 的数据包给自己的父节点.为了描述方便,给出以下定义:

**定义 1** 轮(Round):是从所有传感器节点收集一次数据,并传送到 Sink 节点的过程[3].

**定义 2** 在一轮中节点 $v_i$ 在一棵树上的能量耗费 $S(v_i)$为:

$S(v_i)=C(v_i)kE_{rx}+kE_{tx}=D(v_i)kE_{rx}+k(E_{tx}-E_{rx})$,其中 $C(v_i)$是节点 $v_i$ 在树上的孩子数量,$D(v_i)=C(v_i)+1$ 是节点 $v_i$ 在树上的度数.

**定义 3** 节点的生命周期(Node lifetime):是节点 $v_i$ 在一棵树中能存活的轮数.存活指节点 $v_i$ 的能量 $E(v_i)>0$.节点 $v_i$ 在一棵树中的生命周期可定义为:

$$L_{\text{node}}(v_i)=\left\lfloor\frac{E(v_i)}{D(v_i)kE_{rx}+k(E_{tx}-E_{rx})}\right\rfloor \tag{1}$$

**定义 4** 树的生命周期(Tree lifetime):是树 $T$ 中第一个节点死亡时,该节点存活的轮数.定义为:

$$L_{\text{tree}}(T)=\min_{i=0\cdots n}\{L_{\text{node}}(v_i)\} \tag{2}$$

**定义 5** 最优树:网络中生命周期最大的生成树,定义为:$T^*=\{T\mid L_{\text{tree}}(T)=\max_{T'\in Ts(G)}L_{\text{tree}}(T')\}$,$T'$ 是网络 $G$ 中任意一棵生成树,$Ts(G)$是图 $G$ 中所有生成树的集合.

### 3.3 问题描述

网络中的数据收集延迟主要受树的高度和树上节点的孩子数量等因素的影响,其中树高对延迟的影响最大[7].这是因为树高越大,数据传送到 Sink 需要的时间越长;而节点的孩子数量越多,节点就需要花费越多的时间等待所有孩子传送来数据并进行聚合.网络中节点的数量 $n$ 等于树上各层节点孩子的总和,即:

$$n=\sum_{i=1}^{h'-1}\sum_{v_j\in h_i}C(v_j)=\sum_{i=1}^{h'-1}\sum_{v_j\in h_i}(D(v_j)-1) \tag{3}$$

其中 $h'$代表树高,$h_i$ 代表树上第 $i$ 层所有节点的集合.根据式(3),当树中每个节点的度最小,树的高度最大;反之亦然.因此,要得到小的延迟,需要减少树的高度,但是这样可能会使树上某些节点的度变大.

另一方面,根据式(1)和式(2),生命周期最大的树可以表示为:

$$\max L_{\text{tree}}(T)=\max\min_{i=0\cdots n}\left\{\left\lfloor\frac{E(v_i)}{D(v_i)kE_{rx}+k(E_{tx}-E_{rx})}\right\rfloor\right\} \tag{4}$$

由于 $E_{rx}$和$E_{tx}$是固定的,只有 $D(v_i)$可以调节,是主要的优化目标.为了简化表达,对式(4)进行等价变换:

$$\max L_{\text{tree}}(T) \Leftrightarrow \max \min_{i=0\cdots n} \frac{E(v_i)}{D(v_i)+c} \tag{5}$$

其中 $c = E_{tx}/E_{rx} - 1$.根据式(5),要使树的生命周期最大,每个节点的度需要最小化.但是根据式(3)这样会增加树的高度,使延迟增加.因此,树的生命周期最大和最小延迟这两个目标是相互矛盾的,很难同时实现.针对这个问题,本文研究:在限定延迟的情况下,如何使得树的生命周期最大? 文献[8]证明构造这类最优的数据收集树是 NP 完全的.

## 4　算法 MILD 的设计

根据式(5),$E(v_i)$的值越大,$D(v_i)$的值也应越大,反之亦然.这说明树上节点的度应该与其能量成比例(即负载均衡),而"瓶颈节点"就是度超过合理比例最多的节点.要提高树的生命周期,必须使所有"瓶颈节点"的负载最小化.同时,为了保证延迟要求,树的高度不能超过 1 个限定值 $h$.

### 4.1　算法 MILD 描述

首先,利用网络 $G$ 的拓扑构造一棵最少跳生成树 $T$,这棵树中每个节点都可以以最短路径到达 Sink.方法为:先将图 $G$ 中所有的边赋予相同的权值 1,然后采用 Dijkstra 最短路径算法从 Sink 节点出发求解即可.注意:$T$ 具有最小的树高.

接下来,以树 $T$ 为基础进行优化,不断地将树上"瓶颈节点"的孩子转移到"非瓶颈节点"上去,以减小"瓶颈节点"的度($D(v_i)$).但是,$D(v_i)$在式(5)中是式子的分母,并不容易进行调整.因此,将式(5)转换为如下等价的形式:

$$\max L_{\text{tree}}(T) \Leftrightarrow \min \max_{i=0\cdots n} \frac{D(v_i)+c}{E(v_i)} \tag{6}$$

定义 $r(T) = \max_{i=0\cdots n} \frac{D(v_i)+c}{E(v_i)} = \max_{i=0\cdots n} r(v_i)$,称为树 $T$ 的最大反生命周期,$r(v_i)$是树 $T$ 上节点 $v_i$ 的反生命周期.这样,问题 $\max L_{\text{tree}}(T)$就转化为求树的最大反生命周期最小化问题,即:

$$\max L_{\text{tree}}(T) \Leftrightarrow \min r(T) \tag{7}$$

此外,要减少"瓶颈节点"的度,首先要明确哪些节点属于"瓶颈节点".首先,定义"瓶颈节点"为与树 $T$ 的生命周期(或反生命周期)一致的节点.再观察式(6),除去 Sink 节点,树 $T$ 上节点反生命周期的最小变化幅度是 $\delta = 1/E_{\max}$,其中 $E_{\max} = \max_{i=1\cdots n} E(v_i)$.因此,采用与文献[6]类似但参数不同的分类方法,根据反生命周期大小将树上节点划分到 3 个不同的集合 $V_1$、$V_2$、和 $V_3$ 中:

(1) $V_1 = \{v_i \mid r(T) - \delta < r(v_i) \leqslant r(T), v_i \in V\}$,在这个集合中节点的反生命周期与树 $T$ 的反生命周期在同一区间内,属于"瓶颈节点".

(2) $V_2 = \{v_i \mid r(T) - \delta - 1/E(v_i) < r(v_i) \leqslant r(T) - \delta, v_i \in V\}$,在这个集合中节点的反生命周期非常接近于"瓶颈节点"的反生命周期.如果它们的孩子增加一个,它们就会变为"瓶颈节点".因此,称这个集合中的节点为"次瓶颈节点".

(3) $V_3 = V - V_1 - V_2$,这个集合中节点的负载较轻,即使增加一个孩子也不会成为"瓶颈节点",称为"富裕节点".

接着,再定义算法 MILD 中"优化"操作的含义为:在树 $T$ 中针对某个节点 $x$ 的优化,就是从图 $G$ 中选择一条合适的边$(u, v)$加入树 $T$ 并产生一个包含节点 $x$ 的圈,接着再选择性地删除另一条在圈上且与 $x$ 相连的边,使 $x$ 的度减 1 同时产生的新树的高度不增长或者增长最小以不超过树高限定 $h$.我们的目标就是不断地优化"瓶颈节点",使它们的负载最小化.接下来,将描述如何在树 $T$ 上进行优化操作.

```
Funciton FindEdge(T, L)
1.    计算树 T 及其上所有节点的反生命周期,判断节点属于
      V1、V2 或 V3 中的哪个集合;
2.    T0←T;
3.    将树 T0 中所有属于 V1 和 V2 的节点删除,得到一个组件
      的集合 F;
4.    for(图 G 中每一条连接不同组件的边(u, v))
5.    {   wuv = level(u) + level(v);
6.        根据权值 wuv的大小,按递增的次序将记录(u, v, wuv)
          保存到表 L 中;
7.    }
```

图 1　函数 FindEdge($T$, $L$)

首先,定义一个函数 FindEdge($T$, $L$)来寻找所有加入树后能将一个或多个"瓶颈节点"包含在一个圈中的边,如图 1 所示.其中,$L$ 是一个表(List).在图 1 中,函数 FindEdge($T$, $L$)首先根据树 $T$ 上节点的度和能量计算各节点和树的反生命周期,判断节点属于 $V_1$、$V_2$ 或 $V_3$ 中的哪个集合.接着,复制树 $T$ 到 $T_0$,并将树 $T_0$ 中所有属于 $V_1$ 和 $V_2$ 的节点(即:所有"瓶颈节点"和"次瓶颈节点")删除.此时会产生一系列的子树,子树中每个节点均为"富裕节点".将每棵子树上的所有节点作为一个集合保存起来,称为一个组件(Component),并将这些子树所对应的组件的集合定义为 $F$.注意到有这样一个性质:如果图 $G$ 中有一条边连接两个不同的组件,则把它加入树 $T$ 后会产生一个唯一的圈,而且圈中肯定包含属于 $V_1$ 或 $V_2$ 的节点.最后,对于图 $G$ 中所有连接两个不同组件的边,首先对它们赋予新的权值 $w_{uv}$ = level($u$) + level($v$),其中 level($v_i$)表示节点 $v_i$ 在树 $T$ 中的层次.然后根据权值 $w_{uv}$的大小,将边的记录($u$, $v$,

$w_{uv}$)按递增的次序保存在表 $L$ 中.

```
Funciton FindEdge(T, L)函数运行时,一条边(u,v)已经被加入树 T
并使"瓶颈节点"x 被包在一圈 C 中
1.    找到节点 x 在圈 C 中所关联的 2 条边 g1 和 g2;
2.    t1←将边 g1 从树 T 中删除后新树的高度;(得到 t1 后将边
      g1 加回树 T 中;)
3.    t2←将边 g2 从树 T 中删除后新树的高度;(得到 t2 后将边
      g2 加回树 T 中;)
4.    if(t1 > h && t2 > h) return false; //h 是树高限定
5.    if(t1 < = t2)
6.      把边 g1 从树 T 中删除;
7.    else 把边 g2 从树 T 中删除;
8.    return true;
```

图 2 函数 Optimal $(x,T)$

对于表 $L$ 中的每条边,将按权值从小到大的次序被逐个加入树 $T$ 中尝试对"瓶颈节点"进行优化.因此,再定义一个函数 Optimal$(x,T)$来实现优化操作,如图 2 所示.在图 2 中,当函数 Optimal$(x,T)$被执行时,一条边$(u,v)$已经被加入树 $T$ 并使"瓶颈节点"$x$ 被包含在一个圈 $C$ 中.首先,分别找到 $x$ 在圈 $C$ 中所关联的 2 条边 $g1$ 和 $g2$.然后,再分别尝试删除边 $g1$ 和 $g2$,统计相应产生的新树的高度 $t1$ 和 $t2$ (得到 $t1$ 或 $t2$ 后,边 $g1$ 或 $g2$ 分别被加回树 $T$ 中).如果 $t1$ 和 $t2$ 均大于树高限定 $h$,说明利用边$(u,v)$无法对树 $T$ 进行优化,则函数返回 false;否则,哪条边被删除后产生的新树的高度最小,则被正式地删除,然后函数返回 true.

定义了函数 FindEdge$(T,L)$和函数 Optimal$(x,T)$,算法 MILD 就可以对整棵树进行优化操作了.算法 MILD 的详细描述如图 3 所示.MILD 首先构造一棵最少跳生成树 $T$,以确保最小的树高.然后,调用函数 FindEdge$(T,L)$找到所有能优化"瓶颈节点"的边,并将这些边保存到表 $L$ 中.表 $L$ 建立完成后,从第一条记录开始访问它,取出当前记录所保存的边$(u,v)$.将边$(u,v)$加入树 $T$ 中,会产生一个圈 $C$.如果圈 $C$ 中没有出现"瓶颈节点",则将边$(u,v)$从树 $T$ 中删除.接着访问表 $L$ 中的下一条记录,找出下一条边继续尝试进行优化操作.如果圈 $C$ 中出现了 1 个或多个"瓶颈节点",则对任意一个"瓶颈节点"$v_j$ 执行优化操作.

如果优化操作成功,即 Optimal$(v_j,T)$ = true,则 $v_j$ 的度会减 1,树的结构随之发生改变.此时,算法跳出当前循环并进入下一轮迭代以继续对新树进行优化.如果优化操作不成功,即 Optimal$(v_j,T)$ = false,说明优化 $v_j$ 会使树高超过限定 $h$,则放弃优化 $v_j$ 并继续尝试优化圈 $C$ 中下一个"瓶颈节点".如果圈 $C$ 中所有的"瓶颈节点"均无法利用边$(u,v)$进行优化,则把边$(u,v)$从树 $T$ 中删除.接着,访问表 $L$ 中的下一条记录,找出下一条边继续尝试进行优化操作.当表 $L$ 中最后一条记录被访问完毕,没有可以进行优化操作的边时,算法结束并得到一棵高度最多为 $h$ 的负载均衡树.

```
算法 MILD
输入:图 G 和树高限定 h
输出:高度最多为 h 的最大化生命周期树
1.    将图 G 中的边赋权值 1,采用 dijkstra 算法从 v0 出发构造一
      棵最小跳生成树 T;
2.    if(树 T 的高度 > h) return false;
3.    Ischanged = TRUE;
4.    while(Ischanged)
5.    { Ischanged = FALSE; L = ϕ; i = 0;
6.      FindEdge(T, L);
7.      while(i < Length(L))
8.      { 访问记录 L[i],将其包含的边(u,v)加入树 T 中产生
        一个圈 C;
9.          for(圈 C 中的每一个"瓶颈节点"vj)
10.             if(Optimal(vj, T)) {Ischanged = TRUE; break;}
11.       if(Ischanged) break;     //成功优化,进入下一轮迭代
12.       把边(u,v)从树 T 中删除;
13.       i = i + 1;
14.     }//while
15.   }//while
```

图 3 算法 MILD 描述

图 3 中,算法 MILD 第 1 步采用 Dijkstra 算法构造一棵最少跳生成树 $T$,在 $O(n^2)$时间内算法能够结束.第 4 ~ 15 步迭代地对 $T$ 进行优化.假设最优树的反生命周期是 $r^*$,而 MILD 可以通过执行一定数量 $U$ 的迭代使 $T$ 的反生命周期 $r(T)$降低 $\delta$(即:使所有的"瓶颈节点"的反生命周期小于或等于 $r(T)-\delta$).因此,当算法结束时,至多执行了$\lceil (r(T)-r^*)/\delta \rceil U$ 次迭代.每次迭代或者使一个"瓶颈节点"小于或等于 $r(T)-\delta$ 而进入下一轮迭代,或者无法继续优化任何一个"瓶颈节点"而中止算法.因此,MILD 一定能结束.执行完算法 MILD 后,得到的生成树就是一棵在高度限制内负载均衡的树.以下将对算法的时间复杂度和优化程度进行分析.

## 4.2 算法 MILD 的时间复杂度

由图 3,算法 MILD 第 1 步采用 Dijkstra 算法构造一棵最少跳树 $T$,其时间复杂度为 $O(n^2)$.算法 4 ~ 15 步是一个迭代的优化过程.对于某个"瓶颈节点"$v_i$,每优化成功一次,能使其反生命周期减少 $1/E(v_i)$,接着进入下一轮迭代.而最多经过$\lceil \delta E(v_i) \rceil = 1$ 次优化操作,该"瓶颈节点"的反生命周期将小于 $r(T)-\delta$.因此,将树 $T$ 的反生命周期 $r(T)$减少 $\delta$,需要进行的迭代次数 $U$ 为:

$$U=\sum_{v_i\in V_1}\lceil \delta E(v_i)\rceil = |V_1| \tag{8}$$

对于任意一个"瓶颈节点"$v_i$,均有 $r(T)-\delta < r(v_i)\leqslant r(T)$,根据式(6)有 $r(T)-\delta<(D(v_i)+c)/E(v_i)\leqslant r(T)$,因此$(r(T)-\delta)E(v_i)<D(v_i)+c$.对于

一棵树,树上所有节点的度的总和不会超过 $2(n-1)$.因此,对于所有"瓶颈节点":

$$(r(T)-\delta)\sum_{v_i\in V_1}E(v_i)<\sum_{v_i\in V_1}(D(v_i)+c)$$
$$\leqslant 2(n-1)+cn<(2+c)n \tag{9}$$

对于式(9),有

$$(r(T)-\delta)E_{\min}|V_1|\leqslant(r(T)-\delta)\sum_{v_i\in V_1}E(v_i)<(2+c)n$$

其中 $E_{\min}=\min_{i=1\cdots n}E(v_i)$,因此

$$|V_1|<(2+c)n/(r(T)-\delta)E_{\min} \tag{10}$$

将式(10)代入式(8),有:

$$U=|V_1|<(2+c)n/(r(T)-\delta)E_{\min} \tag{11}$$

注意到 $r(T)\leqslant\lceil(n+c)/E_{\min}\rceil$并联合式(11),可知 $U=O((n/\delta E_{\min})\log(n/\delta E_{\min}))$.每次优化操作中,寻找连接不同组件的边可采用文献[9]中的 Disjoint Set Union-find 算法实现,时间复杂度为 $O(m\alpha(m\cdot n))$,其中 $\alpha(\cdot)$是逆 Ackerman 函数(Inverse Ackerman Function).而每删除一条边都需要对树进行遍历以确定新的树高,这需要 $O(n)<O(m\alpha(m\cdot n))$的时间.又因为$\lceil(r(T)-r*)/\delta\rceil<(n+c)/\delta E_{\min}$,因此算法 4~15 步总共需要 $\lceil(r(T)-r*)/\delta\rceil UO(m\alpha(m\cdot n))=O(n^2\log(n/\delta E_{\min})m\alpha(m\cdot n)/(\delta E_{\min})^2)$的时间.

**定理 1**　算法 MILD 的时间复杂度为 $O(n^2\log(n/\delta E_{\min})m\alpha(m\cdot n)/(\delta E_{\min})^2)$.

证明:由以上分析,算法第 1 步的时间复杂度为 $O(n^2)$;第 4~15 步的时间复杂度为 $O(n^2\log(n/\delta E_{\min})m\alpha(m\cdot n)/(\delta E_{\min})^2)>O(n^2)$.因此,整个算法的时间复杂度为 $O(n^2\log(n/\delta E_{\min})m\alpha(m\cdot n)/(\delta E_{\min})^2)$.得证.

根据定理 1 的结果,算法的执行需要节点具有较强的计算能力.而 Sink 节点拥有无限的能量和强大的计算能力,因此算法 MILD 最适合在 Sink 节点上运行.另一方面,对于 DB-MDST,其时间复杂度为 $O(n^2m\log^2n)$.这虽然略小于 MILD 的时间复杂度,但是 MILD 与 DB-MDST 均运行在 Sink 节点上,而且 Sink 节点的计算能力强大,因此我们更关注算法构造的树在生命周期和延迟方面的性能.

另一方面,为了使 Sink 节点知道网络的拓扑和节点的能量水平,可以通过 Sink 节点广播一个消息,然后所有节点根据这个消息组成一棵任意树来上传自己的邻居和能量信息[6].Sink 节点计算出延迟受限的最大化生命周期树后,再将树结构通过广播发送给所有节点.

### 4.3　算法 MILD 的近似比

设 $L^*$为最优树的生命周期,$L$ 为算法 MILD 构造的树的生命周期,则:

$$L^*=\frac{E(v^*)}{D(v^*)kE_{rx}+k(E_{tx}-E_{rx})},$$

$$L=\frac{E(v_m)}{D(v_m)kE_{rx}+k(E_{tx}-E_{rx})}.$$

其中,$v^*$是最优树上的"瓶颈节点",$v_m$ 是 MILD 树上的"瓶颈节点".因此,算法 MILD 的近似比可表示为:

$$\frac{L^*}{L}=\frac{(D(v_m)E_{rx}+(E_{tx}-E_{rx}))E(v^*)}{(D(v^*)E_{rx}+(E_{tx}-E_{rx}))E(v_m)} \tag{12}$$

对于节点 $v^*$,在度最小时达到生命周期的上界.而 $v^*$是一个叶子节点时度最小,即 $D(v^*)=1$.而对于节点 $v_m$,在度最大时达到生命周期的下界;MILD 算法结束时,最坏情况下 $v_m$ 通信范围内的所有邻居节点均以 $v_m$ 为父节点,此时 $D(v_m)\approx\pi r^2\rho=n\pi r^2/M^2$,其中 $r$ 为节点的通信半径,$\rho$ 为网络中的节点密度.为了保证网络连通,$r=\frac{\sqrt{2}}{2}M\sqrt{\frac{1}{n}\log(\frac{n}{\varphi})}$,$1-\varphi$ 是网络连通的概率[10].此外,如果节点采用固定发射功率,则其发送能耗大约是接收能耗的 2 倍[10],即 $E_{tx}=2E_{rx}$.因此,在最坏情况下,MILD 的近似比可表示为:

$$\frac{L^*}{L}=\frac{(\pi\log(\frac{n}{\varphi})+2)E(v^*)}{4E(v_m)}<\frac{(\pi\log(\frac{n}{\varphi})+2)E_{\max}}{4E_{\min}}$$
$$=\frac{\pi\log(\frac{n}{\varphi})+2}{4\delta E_{\min}}=O(\log(n/\varphi)/\delta E_{\min}) \tag{13}$$

## 5　模拟试验

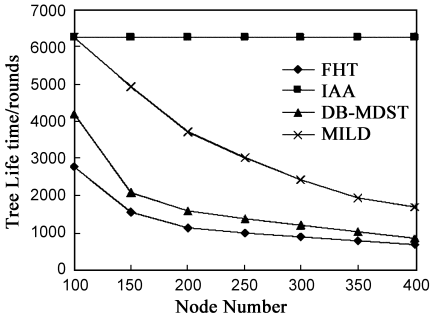
假设网络中所有节点随机地分布在一个 100 * 100 平方米的正方形区域.每个节点的初始能量在[1J,1.5J]之间随机分布,节点的数据产生率为 1000 bits/round,$E_{rx}$ = 50nJ/bit.所有节点的最大通信半径为 20 米.选择算法 FHT、IAA、DB-MDST 来进行对比.为了测试算法在最小延迟限制下的性能,限定 DB-MDST 和 MILD 生成树的最大树高为最少跳树 FHT 的高度.IAA 的树高不受限制,观察最大化生命周期树的上界.网络的 MAC 层采用 IEEE 802.15.4 非时隙 CSMA-CD 标准[11].

为了考察 Sink 节点的位置以及不同网络节点密度对算法性能的影响,选择 2 个场景:场景 1 中 Sink 节点位于区域的中心,坐标(50,50);场景 2 中 Sink 节点位于区域的边缘,坐标(100,50).在两个场景中以 50 为增量,分批放置 100~400 个节点.分别观察各个算法的树生命周期和最大延迟,实验的结果均是执行 20 次后的平均结果.

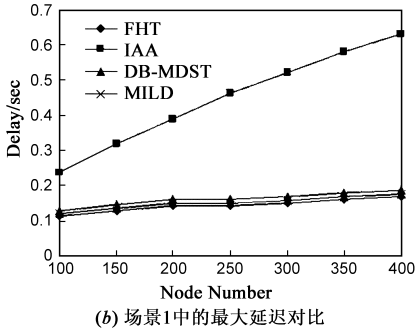
### 5.1　场景 1 中的对比

由图 4(a)可以看到,在场景 1 中 FHT 的树生命周期最小,IAA 的树生命周期最大.MILD 能在节点密度低的网络中取得与 IAA 近似的树生命周期,但是随着节点数量的增加,其树生命周期逐渐下降.另一方面,MILD在所有网络密度下,均取得比DB-MDST更高的树

生命周期.由图 4(*b*)可以看到,虽然 IAA 的树生命周期最大,但是造成的网络延迟也最大.FHT、DB-MDST 和 MILD 的延迟相差不大,这是因为 DB-MDST 和 MILD 的树高被限制为与 FHT 的树高一样.




(*a*) 场景1中的树生命周期对比

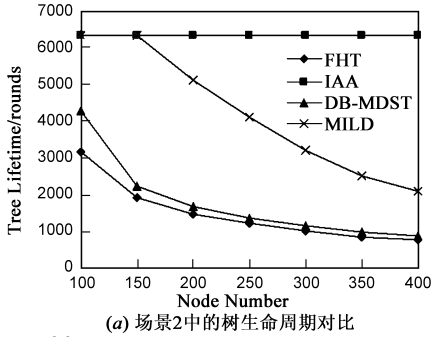



(*b*) 场景1中的最大延迟对比

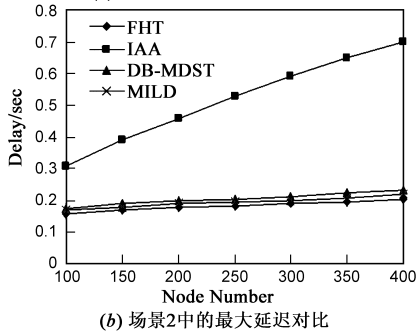
**图4 场景1中的对比**

## 5.2 场景 2 中的对比

由图 5(*a*)可以看到,与场景 1 相比,在场景 2 中 IAA 的树生命周期基本保持不变,但是 FHT、DB-MDST 和 MILD 的树生命周期均有增加.由图 5(*b*),与场景 1 的图 4(*b*)相比,所有算法的延迟都增加了.这是因为 Sink 节点位于区域的边缘,某些节点到 Sink 节点的距离增加了,这就使所有的树的高度增加了.另一方面,FHT 的树的高度增加后,又有利于 DB-MDST 和 MILD 在更宽松的树高限制内优化节点的度.此外,无论节点密度如何,MILD 的树生命周期均优于 FHT 和 DB-MDST.




(*a*) 场景2中的树生命周期对比




(*b*) 场景2中的最大延迟对比

**图5 场景2中的对比**

## 6 总结

在本文中,为了满足网络中延迟限定和最大化生命周期的要求,研究如何构造一棵有高度限制的、负载均衡的生成树.求解这样一棵最优树是一个 NP 完全问题.通过对延迟模型和生命周期模型进行分析,提出一个新的算法 MILD 来解决.仿真实验表明,与目前已有的研究对比,MILD 能够在同样的延迟限定下构造生命周期更长的生成树.

本算法主要研究如何集中式地构造一棵树,适用于拓扑变化并不频繁的网络.但是,在拓扑频繁变化的动态网络中,Sink 节点需要周期性地采集网络拓扑信息并重新计算树结构.如果树结构发生变化,则 Sink 节点再将新的树结构广播给网络中的所有节点.以上这些操作需要节点花费额外的能量,因此在下一步工作中,我们将综合考虑以上因素来研究适合动态网络的算法.此外,本算法研究的是如何最大化网络中第一个死亡的节点的生命周期.而在一些密集部署的网络中,节点大量冗余,只有死亡一定比例的节点后网络才会失效.因此,下一步我们将研究合适的网络生命周期界定问题以及密集部署网络中的生命周期最大化问题.

**参考文献:**

[1] 刘贞,丁明理,王祁.WSN 多节点决策信息融合在机器人自主导航中的应用[J].电子学报,2008,36(12):2299 – 2305.
Liu Zhen, Ding Ming-li, Wang Qi. Implementation of WSN multi – node decision information fusion in autonomous navigation of robot[J]. Acta Electronica Sinica, 2008, 36(12): 2299 – 2305. (in Chinese)

[2] 蔚赵春,周水庚,关佶红.无线传感器网络中数据存储与访问研究进展[J],电子学报,2008,36(10):2001 – 2010.
Yu Zhao-chun, Zhou Shui-geng, Guan Ji-hong. Data storage and access in wireless sensor networks: A survey[J]. Acta Electronica Sinica, 2008, 36(10): 2001 – 2010. (in Chinese)

[3] TAN H, KORPEOGLU I. Power efficient data gathering and aggregation in wireless sensor networks[A]. Proc. ACM SIGMOD Record[C]. New York USA: ACM NY, 2003. 66 – 71.

[4] 张卿,谢志鹏,凌波,等.一种传感器网络最大化生命周期数据收集算法[J].软件学报,2005,16(11):1946 – 1957.
Zhang Qing, Xie Zhi-peng, Ling Bo, et al. A maximum lifetime

data gathering algorithm for wireless sensor networks[J]. Journal of Software, 2005, 16(11): 1946 – 1957. (in Chinese)

[5] Liang Wei-fa, Liu Yu-zhen. Online data gathering for maximizing network lifetime in sensor networks[J]. IEEE Transaction on Mobile Computing, 2007, 6(1): 2 – 11.

[6] Wu Yan, Sonia F, Ness S. On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm[A]. Proc The IEEE 27th Conference on Computer Communications (INFOCOM2008)[C]. Washington, DC, USA: IEEE Computer Society, 2008. 356 – 360.

[7] Kwon S, Kim J, Kim C. An efficient tree structure for delay sensitive data gathering in wireless sensor networks[A]. Proc The IEEE 22nd International Conference on Advanced Information Networking and Applications[C]. Washington, DC, USA: IEEE Computer Society, 2008. 738 – 743.

[8] Buragohain C, Agrawal D, Suri S. Power aware routing for sensor databases[A]. Proc The IEEE 24th Conference on Computer Communications (INFOCOM2005)[C]. Washington, DC, USA: IEEE Computer Society, 2005. 1747 – 1757.

[9] Thomas C, Chomas L, Ronald R, et al. Introduction to Algorithms[M]. Cambridge: MIT Press, 2001. 25 – 28.

[10] Vivek M, Catherine R. Design guidelines for wireless sensor networks: communication, clustering and aggregation[J]. Ad Hoc Network Journal, 2004, 2(1): 45 – 63.

[11] Bougard B, Catthoor F, Daly C, et al. Energy efficiency of the IEEE 802.15.4 standard in dense wireless micro-sensor networks: modeling and improvement perspectives[A]. Proc IEEE Design, Automation and Test in Europe Conference and Exhibition[C]. Washington, DC, USA: IEEE Computer Society, 2005. 196 – 201.

**作者简介:**

**梁俊斌**　男, 1979 年 3 月出生于广西壮族自治区南宁市. 博士研究生. 主要研究方向为无线传感器网络.
E-mail: liangjb2002@163.com

**王建新**　男, 1969 年 12 月出生于湖南省邵阳市. 博士, 教授, 博士生导师. 主要研究方向为无线传感器网络, 网络优化理论.
E-mail: jxwang@mail.csu.edu.cn

**陈建二**　男, 1954 年 3 月出生于广西壮族自治区桂林市. 博士, 教授, 博士生导师. 主要研究方向为计算复杂性及优化.
E-mail: chen@cs.tamu.edu