文章编号: 1001-0920(2017)07-1259-07 DOI: 10.13195/j.kzyjc.2016.0929

有向感知网络中分簇目标覆盖算法

王槐元, 丁 旭, 黄 成, 吴晓蓓, 王力立†

(南京理工大学 自动化学院，南京 210094)

摘 要: 利用最少数量的感知节点来覆盖最大数目的目标位置(MCMS)一直是有向感知网络中的重要问题. 为了保证覆盖和对相关事件的及时汇报,针对该问题提出一种基于分簇的目标位置覆盖算法(TCCA). 通过所有节点自组织进行分簇并且在各个簇内为成员节点分配相应的感知扇区,TCCA算法能够在保证网络生命周期的前提下有效解决该问题. 与其他已有算法相比,仿真结果很好地验证了所提出TCCA算法的有效性.

关键词: 有向感知网络；目标覆盖；分簇

中图分类号: TP273 **文献标志码:** A

Clustering based target coverage in directional sensor networks

WANG Huai-yuan, DING Xu, HUANG Cheng, WU Xiao-bei, WANG Li-li†

(Institute of Automation，Nanjing University of Science and Technology，Nanjing 210094，China)

Abstract: The maximum target coverage with the minimum number of directional sensor nodes, known as maximum coverage with minimum sensors(MCMS) problem, is an important problem in directional sensor networks. In order to ensure the network coverage and event reporting, the target coverage algorithm(TCCA) based on clustering is proposed. By self-organizing clustering and sensing sector assignments in each cluster, the TCCA algorithm can solve this problem and enhance the network lifetime in an energy-efficient way. Comparing to the existing algorithms, the simulation results show the effectiveness of TCCA.

Keywords: directional sensor networks；target coverage；cluster

0 引 言

有向感知网络是由大量感知有向的传感器节点组成的,它们通过自组织的形式组网进而将感知到的数据传输到sink节点. 与一般的全向感知节点不同,有向感知节点的感知方向被分为若干个扇区,节点可以根据需要决定某个或某几个感知扇区处于激活状态. 因此,有向感知节点可以用于许多实际应用中,如农业监控、军事侦察、基础设施检测等[1]. 这类应用中,一般有多个目标位置需要被覆盖,所以需要用动态算法来计算得到最优数量的感知节点,进而覆盖并监测这些目标位置.

全向感知网络中的覆盖问题已经得到了充分的研究[2-5],而有向感知网络中该问题的研究才刚刚开始[6]. 覆盖问题一般可以分为3类:区域覆盖[7-9]、目标覆盖[10-12]和栅栏覆盖[13]. 随机分布式算法ACDA[9]通过在有向感知网络中分簇来实现网络的连通和对区域的有效覆盖,其主要分为确定感知扇区、选择通信扇区和簇头节点、选择网关节点以及更新簇头和网关节点4个阶段. ACDA 算法利用随机等待时间来选择簇头和网关节点,这期间交换的信息数量较大,其在选择簇头和网关节点时也没有考虑节点的剩余能量,而且最后的簇头和网关节点更新需要消耗大量的能量从而减少算法的有效性.

目标覆盖在相关领域中也有比较重要的应用. 文献[10]提出了一个多重有向覆盖集(MDCS)的概念,通过选取多个覆盖集的方式来轮流覆盖监测区域. 其所提出的算法为集中式算法,计算和通信开销较大,从而使其扩展性较差. 文献[12]提出了集中式贪婪算法CGA和分布式贪婪算法DGA,主要考虑利用最少数量的节点来最大化所能覆盖的目标位置数目(MCMS)的问题. 同时DGA提出了一个感知邻居协作休眠机制(SNCS)来节省能量开销,此时节点的剩余能量被用作参考,所有的节点需要再次运行DGA算法. 因此,这个休眠机制有大量能量开销,从

收稿日期: 2016-07-21；**修回日期:** 2016-10-26.

基金项目: 江苏省高校优势学科建设工程二期项目.

作者简介: 王槐元(1989−), 男, 博士生, 从事无线传感器网络的研究；吴晓蓓(1958−), 女, 教授, 博士生导师, 从事网络控制、无线传感器网络等研究.

†通讯作者. E-mail: mail_wll@126.com

而影响网络性能和生命周期. 由于各个节点之间没有相互的协调管理者,节点间的计算和通信开销都较大. 同时,以上算法均未考虑利用分簇来决定各个节点的感知扇区以及减小处于激活状态的节点数量.

还有一些利用分簇实现目标覆盖的算法,如算法TCDC[11]在ACDA的基础上提出了相应的系数CSP、GSP和TCP来分别选择簇头、网关节点和感知扇区. 其把节点剩余能量、到sink节点的距离和邻近节点数等作为权衡的指标. 然而,TCDC算法选择的簇头和网关节点可能由于能量消耗而需要频繁地更新,进而影响网络整体性能和生命周期. 而且根据其TCP指标给各个节点分配的感知扇区可能使得簇内的某些目标位置没有得到覆盖.

针对上述问题,本文提出一种分簇目标覆盖算法TCCA,利用分簇方法把有向覆盖集的选择问题划分到各个簇内进行,尽量选择没有覆盖目标位置的节点作为簇头,并在给簇内各个节点分配相应感知扇区时,不仅仅使用相应系数来决定,而是充分考虑簇内各个目标位置的覆盖情况,提高算法性能和网络生命周期. 最后,通过仿真结果表明了本文算法的有效性,并给出了在部署网络时的一些参数量的取值建议.

1 网络模型和参数

假设在二维平面中随机部署了n个感知有向的节点来覆盖m个目标位置,节点通过自组织成网把感知的数据传送到sink节点. 各个目标位置的坐标(x_i^m, y_i^m)已知,每个节点S_i有唯一的ID并且有固定的已知坐标(x_i, y_i). 所有节点都有相同个数的感知扇区、感知半径R_s、通信半径R_c以及初始的能量E_0,节点的通信是全向的.

节点的各个感知扇区可以处于激活或者睡眠模式,所有位于节点感知半径内的目标位置都能够被覆盖,但节点各个感知扇区能够覆盖的目标位置各不相同. 在网络部署之后,有向感知网络中的所有节点需要根据要求各自选择合适的一个或多个感知扇区作为工作扇区以完成相应的覆盖任务. 假设节点只有一个感知扇区处于工作状态,感知模块需要消耗的能量为正常值的$1/|\Phi_s|$,其中$|\Phi_s|$表示节点感知扇区数.

由于每个感知节点都有相同数量的感知扇区,每个感知扇区可以用如下的参数表示:

1) $\theta^s(0 < \theta < 2\pi)$: 第$s$个感知扇区的最大感知角度;

2) $V_{i,j}^s$: 第s个感知扇区的中心线,用来表示感知方向的矢量.

部署之后,每个节点都知道自己一跳邻居的信息以及各个感知扇区覆盖目标位置信息. 为了方便描述,定义以下符号:

N: 所有部署节点的集合;

S_i: 第i个节点,$1 \leqslant i \leqslant n, S_i \in N$;

Φ_s: 各个节点所有感知扇区的集合;

S_s^i: 感知节点i的第s个感知扇区,$S_s^i \in \Phi_s$;

E_i: 节点S_i的剩余能量;

E_0: 所有节点的初始能量,$\forall i, S_i \in N$;

n_i: 节点S_i的邻居节点集合;

Tn_i: 节点S_i所有感知扇区覆盖的目标位置集合;

T_s^i: 节点S_i第s个感知扇区覆盖的目标位置集合;

$d(S_i, S_j)$: 节点S_i与S_j之间的距离;

CHS_i: 节点S_i的簇头选择优先级;

TCS_s^i: 选择节点S_i的第s个感知扇区来覆盖目标位置的优先级;

n_k^c: 簇头S_k所在簇的所有节点集合,包括S_k.

定义1 目标位置T_j被节点S_i覆盖,当且仅当S_iT_j与节点S_i第s个感知扇区中心线$V_{i,j}^s$的夹角位于区间$[-\theta/2, \theta/2]$内并且$d(S_i, T_j) \leqslant R_s$.

定义2 假设簇头S_k所在簇组成一个元组$G = (D_k, T_k)$,D_k表示簇内节点中所有覆盖了目标位置的感知扇区的集合,T_k表示簇内节点覆盖的所有目标位置集合. 如果D_k中包括同一个节点的多个感知扇区,则认为这些扇区是相互冲突的.

定义3 簇头S_k所在簇的覆盖集表示为簇内各个节点处于激活状态并且用来覆盖簇内所有目标位置的感知扇区集合. 而最优覆盖集则是能够覆盖簇内所有目标位置且感知扇区数目最小的集合,由相互不冲突的扇区组成.

2 分簇目标覆盖TCCA

TCCA算法通过分布式的分簇机制解决有向感知网络中的MCMS[12]问题,其只需要一跳邻居信息,且分簇之后各个簇头只计算各自簇内的最优覆盖集,从而减少计算和通信开销. 虽然TCCA不一定能得到全局最优的覆盖集合,但具有更大的计算可行性,其主要步骤如下.

1) 成簇: 每个节点$S_i \in N$通过计算各自的CHS_i来决定自己是否能够成为簇头节点,然后通知邻近的节点加入.

2) 簇内最优覆盖集选取: 每个簇头节点S_k根据收集的成员信息,计算n_k^c中每个节点各个感知扇区的TCS_s^i,然后选择相应的感知扇区组成簇内的最优覆盖集,其余感知扇区则处于休眠状态.

3) 簇头更新: 当簇头节点的能量低于某个阈值E_th时, 簇头节点需要被更换. 此时新簇头的选择不再需要各个邻居节点之间交换信息, 而是由簇头节点根据簇内各个节点的信息直接选择替补节点, 自己则变为簇成员且只传递数据.

2.1 成　簇

节点部署之后, 各个节点相互交换邻居信息(NI), 包括: 1) 节点ID; 2) 节点坐标; 3) 节点覆盖的目标位置集合Tn_i; 4) 节点各个感知扇区覆盖的目标位置集合T_s^i; 5) 节点的邻居节点集合n_i; 6) 节点的剩余能量E_i. 每个节点只保留自己一跳邻居的信息, 然后各个节点根据收集到的NI计算各自的CHS_i, 即

$$\begin{aligned}\mathrm{CHS}_i = {} & w_1 \times \frac{E_i}{E_0} + w_2 \times \frac{|n_i|}{n_{\max}^i} + \\ & w_3 \times \Big(1 - \frac{|\mathrm{Tn}_i|}{\mathrm{Tn}_{\max}^i}\Big) + \\ & w_4 \times \Big(1 - \frac{d(S_i, \mathrm{sink})}{d_{\max}}\Big). \end{aligned} \tag{1}$$

其中: $w_1 \sim w_4$是权重值, 且$w_1 + w_2 + w_3 + w_4 = 1$; $|n_i|$和$|\mathrm{Tn}_i|$分别表示集合n_i和Tn_i中元素的个数; $n_{\max}^i$是节点S_i自身和所有一跳邻居节点中邻居节点数的最大值, 即

$$n_{\max}^i = \max\{\max_{\forall j, S_j \in n_i}(|n_j|), |n_i|\}; \tag{2}$$

而$\mathrm{Tn}_{\max}^i$表示节点S_i自身和所有一跳邻居节点中覆盖目标数的最大值, 即

$$\mathrm{Tn}_{\max}^i = \max\{\max_{\forall j, S_j \in n_i}(|\mathrm{Tn}_j|), |\mathrm{Tn}_i|\}; \tag{3}$$

$d_{\max}$是区域中节点到sink的最远距离, 当节点部署完之后其为一个确定的常数.

式(1)中的第1项选取节点剩余能量多的节点; 第2项表明邻居节点多的节点更合适; 第3项则尽量选择没有覆盖目标位置的节点作为簇头节点, 从而减小簇头节点的更换频率; 第4项则要求离sink节点的距离要近. 当$\mathrm{Tn}_{\max}^i = 0$时说明节点自身和周围的一跳邻居节点都不覆盖目标位置, 此类节点将在所有其他节点都成簇之后, 以能量最高的节点为簇头成簇, 所以不需要计算CHS值. 各个节点计算得到自身的CHS_i之后, 比较其值与各个邻居节点值的大小, 判断下式是否成立:

$$\mathrm{CHS}_i \geqslant \max_{\forall j, S_j \in n_i}(|\mathrm{CHS}_j|). \tag{4}$$

当节点S_i的CHS_i满足式(4)条件时, 其确定自己为簇头节点并开始向邻居节点广播成簇消息(CF), 包括: 1) 簇头ID; 2) 该簇头的一跳邻居ID列表.

当节点S_j收到S_i发来的CF消息时, 首先判断自己是否位于一跳邻居ID列表内, 即$S_j \in n_i$. 如果属于该列表, 则接着判断自身是否已加入其他簇. 如果没有, 则S_j向S_i发送确认消息ACF, 确认加入该簇, 并向自己的一跳邻居继续转发CF信息; 否则保存该簇头ID, 标记自身为簇间节点. 如果不属于该列表, 则仍然判断自身是否已加入其他簇, 如果已加入, 则丢弃该消息不再转发并保存该簇头ID和转发消息的节点ID, 标记自身为簇间节点; 否则S_j就丢弃该消息不再转发并保存该簇头ID和转发消息的节点ID, 同时根据式(1)更新自己和邻居节点的CHS值来重新成簇. ACF信息包括: 1) 节点ID; 2) 簇头ID; 3) 节点各个感知扇区覆盖的目标集合T_s^i. 当节点S_i收到多个CF消息时, S_i对最先到达的CF信息作出ACF应答, 而后收到CF消息时则按照前面收到CF消息时的步骤操作, 伪码见算法1.

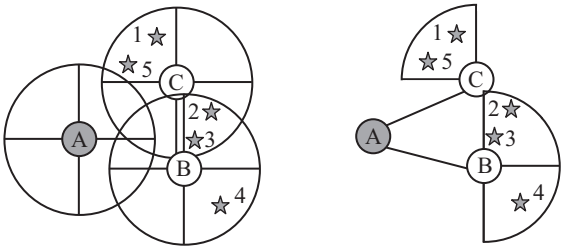
算法1　成簇(各个节点$S_i \in N$).

1) NI: 邻居信息(Tn_i, n_i等)
2) 根据式(1)计算CHS_i
3) if CHS_i满足式(4) then
4)　节点S_i确定为簇头
5)　并向一跳邻居节点发送CF信息
6) else if 节点S_i收到任意簇头S_k的CF信息 then
7)　if S_i是S_k的一跳邻居 then
8)　　if S_i未加入其他簇 then
9)　　S_i向S_k发送ACF信息
10)　　并且向自己的一跳邻居转发该CF信息
11)　　else
12)　　保存该簇头ID, 标记自身为簇间节点
13)　　end if
14)　else if S_i加入了其他簇 then
15)　　保存该簇头ID和转发消息的节点ID
16)　　标记自身为簇间节点并不再转发该消息
17)　else
18)　　保存该簇头ID和转发消息的节点ID
19)　　并丢弃该消息不再转发
20)　　同时根据式(1)更新自己和邻居节点的CHS
21)　　重新选择簇头节点
22)　end if
23) end if.

2.2 簇内最优覆盖集选取

当所有节点都已经成簇后, 簇头节点开始根据簇内节点的感知扇区信息来选取最优覆盖集. 根据定

义3,最优覆盖集应该由各个节点的不冲突扇区组成,但并不是每个簇内都能得到最优覆盖集. 如图1(a)所示,目标位置1、5和4都分别只被节点C和B覆盖,而目标位置2和3则都被节点C和B覆盖,此时簇内不存在最优覆盖集. 为了保证簇内所有的目标位置都能被选取的覆盖集覆盖,簇头节点首先给簇内每个目标位置分配一个权重TW,其值等于该目标位置被簇内节点各个扇区覆盖的次数. 图1(a)中,节点A为簇头节点,节点B和C是簇内节点,圆环表示各个节点的感知半径,每个节点有4个感知扇区,此时$\mathrm{TW}_1 = \mathrm{TW}_4 = \mathrm{TW}_5 = 1, \mathrm{TW}_2 = \mathrm{TW}_3 = 2$.



(a) 簇内覆盖集选取前 (b) 簇内覆盖集选取后

图1 簇内覆盖集选取示例

如果簇内存在权重值为1的目标位置,则簇头节点需要保证这些目标位置的覆盖. 用D_k表示簇内所有覆盖了目标位置的感知扇区的集合,T_k表示簇内目标位置的集合,则覆盖集的选取主要包括以下两种情况.

情况1 D_k中存在不冲突的感知扇区S_s^i. 由定义2可知,节点i只有一个扇区覆盖了目标位置,故把感知扇区S_s^i加入到覆盖集中,并在D_k和T_k中删除相应的感知扇区和目标位置. 如果有多个不冲突的感知扇区,则重复此步骤,直到T_k为空集或者D_k中不存在不冲突的感知扇区,转情况2.

情况2 D_k中不存在不冲突的感知扇区,如图1(a)所示的情况. 此时需要首先保证权重TW为1的目标位置的覆盖,于是先把覆盖目标权重值为1的感知扇区加入到覆盖集中,因此D_k中剩下的感知扇区必然与已选覆盖集中的感知扇区冲突. 为了保证目标位置被全部覆盖,簇头节点需要选取最少数量的感知扇区来覆盖剩余的目标位置,故而簇头节点计算D_k中剩余扇区的覆盖优先级(TCS),即

$$\mathrm{TCS}_s^i = c_1 \times \frac{E_i}{E_0} + c_2 \times \frac{|T_s^i|}{\mathrm{Tn}_i} + c_3 \times \frac{T_s^i}{T_k}. \quad (5)$$

其中:c_1、c_2和c_3是权重值,且$c_1 + c_2 + c_3 = 1$;TCS_s^i表示D_k中剩余扇区S_s^i的覆盖优先级;$|T_s^i|$和$|\mathrm{Tn}_i|$分别表示节点i的第s个感知扇区S_s^i所覆盖目标位置个数和节点i所有扇区覆盖目标位置的总数;$|T_k|$表示此时簇内未被覆盖目标位置的总数.

如图1(b)所示,当TW值为1的目标位置1、4和5被优先覆盖之后,如果节点的剩余能量一样,则由于节点C的$\frac{|T_s^i|}{|\mathrm{Tn}_i|}$值大于节点B,节点C相应扇区的TCS值较大从而该扇区被选入覆盖集. 每次把一个TCS值最大的扇区加入覆盖集,直到T_k为空,从而得到最终的覆盖集. 簇头节点选择完簇内的覆盖集之后,向簇内节点发送TS消息通知各个节点被选入覆盖集的感知扇区ID,簇内节点根据收到的TS激活或者休眠各自的感知扇区,完成对簇内目标位置的覆盖,伪码如下.

算法2 最优覆盖集选取.

1) TW:簇内各个目标位置权重值

2) CS:所选覆盖集,$\mathrm{CS} = \varnothing$

3) D_k:簇内所有覆盖了目标位置的感知扇区集合

T_k:簇内所有目标位置集合

4) while T_k不是空集 do

5) if D_k中存在不冲突的感知扇区S_s^i then

6) 把S_s^i放入覆盖集,$\mathrm{CS} \leftarrow S_s^i$

7) 把S_s^i从D_k中删除,$D_k \leftarrow D_k - S_s^i$

8) 删除S_s^i覆盖的目标位置,$T_k \leftarrow T_k - T_s^i$

9) else if D_k中有TW值为1的目标位置 then

10) 把相应扇区S_s^j放入覆盖集,$\mathrm{CS} \leftarrow S_s^j$

11) 把S_s^j从D_k中删除,$D_k \leftarrow D_k - S_s^j$

12) 删除S_s^j覆盖的目标位置,$T_k \leftarrow T_k - T_s^j$

13) else

14) 根据式(5)计算剩余扇区的TCS值

15) 把TCS值最大S_s^l放入覆盖集,$\mathrm{CS} \leftarrow S_s^l$

16) 把S_s^l从D_k中删除,$D_k \leftarrow D_k - S_s^l$

17) 删除S_s^l覆盖的目标位置,$T_k \leftarrow T_k - T_s^l$

18) end if

19) end while.

选取簇内覆盖集时,各个簇间节点所覆盖的目标位置可能有重叠,如图2所示. 节点A和E为簇头节点,节点B、C和D、F分别属于簇头A和E所在的簇,则节点B和D为簇间节点,都覆盖了目标位置3. 在成簇完成之后,为了减少不必要的重叠覆盖,节点B和D通过比较各自的覆盖信息来决定目标位置3的覆盖:1)若簇间节点都只覆盖这一个目标位置,则由剩余能量多的簇间节点来负责覆盖该目标位置,其他簇间节点忽视该目标位置并更新自身的覆盖信息:2)若簇间节点中存在一个只覆盖这一个目标位置的节点,则直接由该簇间节点覆盖该目标位置,其

余簇间节点忽视该目标位置并更新自身的覆盖信息: 3) 若簇间节点除此之外还覆盖了其他目标位置(如图3),剩余能量多的节点则负责覆盖该目标位置(如节点B): 如果节点D覆盖目标位置3的感知扇区内还覆盖有其他目标位置(假设是目标位置6),则为了选取感知扇区数最少的覆盖集,节点D将负责覆盖目标位置3,其余簇间节点忽视该目标位置并更新自身的覆盖信息.

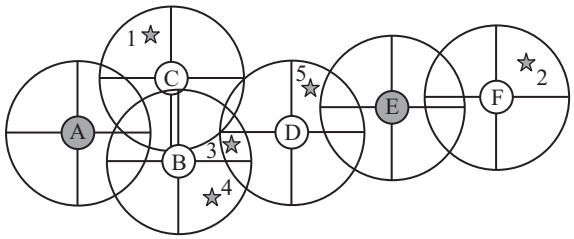


图 2　簇间节点覆盖示例

总之,簇间节点共同覆盖的目标位置T_t的覆盖问题由3个指标决定: 1) 覆盖有目标位置的感知扇区数,数目少者负责覆盖T_t; 2) 覆盖共同目标位置的感知扇区中的目标位置总数,总数大者负责覆盖T_t; 3) 相同情况下,剩余能量多者负责覆盖T_t.

2.3　簇头更新

当簇头节点S_k的剩余能量E_k低于阈值$E_{\rm th}$,即$E_k < E_{\rm th}$时,簇头节点开始收集簇内成员节点的相关信息以重新选取一个簇头节点. 簇头节点根据收集到的信息计算各个成员节点的CHS值,并选取CHS值满足下式的节点S_l作为新的簇头节点:

$$\mathrm{CHS}_l \geqslant \max_{\forall i, S_i \in n_k^c} \mathrm{CHS}_i. \tag{6}$$

由于簇内的成员节点之间并不都是彼此相邻,新选取的簇头节点S_l可能是簇内某些节点的二跳邻居. 为了保证簇和簇内覆盖集的完整性,老的簇头节点S_k在选定新簇头节点之后,通过向簇内成员广播NCI消息通知新簇头节点$S_l(S_l \in n_k^c)$的诞生,同时自身转换为簇内成员节点以保证簇内节点的连通. NCI消息包括: 1) 簇内成员列表; 2) 新簇头节点的ID. 簇头更新阶段,簇头节点在簇内成员节点向其传递数据的同时收集相关信息,从而选定簇头节点,并不需要每个节点再次运行前面的成簇算法. 由于仅需要簇内的局部信息和计算,该阶段的通信和控制开销较小,不会消耗太多的能量.

2.4　算法分析

定理1　成簇算法的计算复杂度为$O(|n_i|)$,选取簇内覆盖集算法的计算复杂度为$O(|n_k^c|^2 \times |\Phi_s|)$.

证明略.

定理2　算法TCCA在各个簇内选取的覆盖集一定能够覆盖各自簇内的所有目标位置并且最大化对簇内目标位置的覆盖周期.

证明略.

3　仿真分析

3.1　仿真和权重参数设置

基于Matlab2009b仿真平台,本文对TCCA算法进行仿真验证,并与ACDA[9]和TCDC[11]算法进行比较,仿真参数如表1所示. 经过一系列仿真,结果表明$w_1 = 0.4$、$w_2 = 0.15$、$w_3 = 0.25$、$w_4 = 0.2$和$c_1 = 0.4$、$c_2 = 0.35$、$c_3 = 0.25$时算法TCCA的效果最佳.

表 1　网络参数

参　数	数　值
仿真区域 / m^2	$1\,000 \times 1\,000$
部署方式	随机均匀部署
部署节点数	200 ~ 800
节点感知扇区数	2 ~ 6
目标位置数	200
节点通信半径 / m	100
节点感知半径 / m	50
节点初始能量 / J	5
|NI|, |CF|, |ACF| / b	240, 192, 160
|TS|, |NCI| / b	160, 160
剩余能量阈值$E_{\rm th}$ / J	1

3.2　性能指标

1) 簇头节点数: 记录各个算法运行完成之后,整个网络分簇所形成的簇头节点数.

2) 激活扇区数: 记录分簇完成以后,各个簇内选取完最优覆盖集之后,各个覆盖集中处于激活状态且用于覆盖目标位置的感知扇区的总数.

3) 目标位置覆盖率: 记录各个簇内最优覆盖集选取完成之后,所有簇内覆盖的目标位置总数与未分簇之前整个网络所覆盖的目标位置总数的比值.

4) 生命周期: 记录从算法初步完成到网络中所有目标位置都不被覆盖的时间周期.

3.3　仿真结果

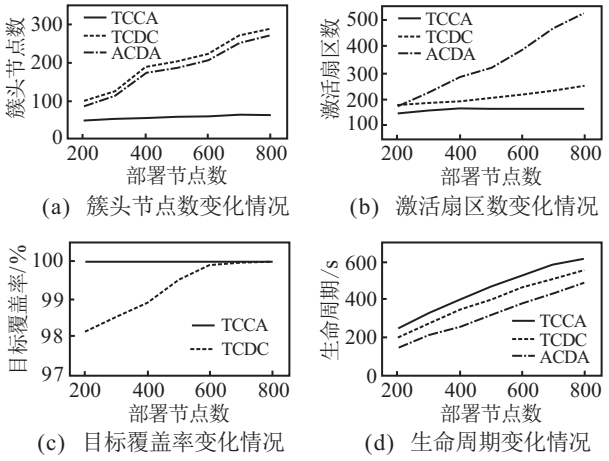
仿真结果主要验证了网络性能随着网络中部署的节点总数和各个节点的感知扇区数的变化情况. TCDC算法和ACDA算法中节点的通信也是有向的,且通信扇区数与感知扇区数一致. 算法ACDA是基于区域覆盖的,所以目标位置覆盖率这一性能参数将不会用于表现该算法的性能.

3.3.1　部署节点数的影响

1) 簇头节点数.

图3(a)显示了簇头节点数随部署节点数的变化

情况,可以看出,TCDC和ACDA算法所构成的簇头节点数明显高于TCCA算法,并且随着部署节点数的增加也相应地增多,而TCCA算法的簇头节点数增加不明显. 这是因为TCCA算法中节点的通信是全向的,而TCDC和ACDA算法中节点的通信是有向的,且每个节点最终只选择一个通信扇区保持通信,各个簇内的成员相对而言少于TCCA算法,所以需要更多的簇来包含所有的节点.



(a) 簇头节点数变化情况 (b) 激活扇区数变化情况

(c) 目标覆盖率变化情况 (d) 生命周期变化情况

图 3 部署节点数影响

2) 激活扇区数.

如图3(b)所示,TCCA算法所得到的簇内覆盖集中激活扇区数随着部署节点数的增加基本保持不变,而TCDC和ACDA算法所得到的簇内覆盖集中激活扇区数随着部署节点数的增加都有所增加. 因为各个簇内的覆盖集都是由各个簇的簇头节点根据各自簇内成员的相关信息选取的,簇内成员越多所选取的最优覆盖集越优,所以簇头节点较少则更容易选取对整个网络最优的覆盖集.

3) 目标位置覆盖率.

图3(c)表明了TCCA和TCDC算法的目标覆盖率随着部署节点数的变化情况. TCCA算法选取覆盖集的最基本条件就是保证目标位置的覆盖,而且每一步都是针对簇内相应的感知扇区的,从而保证了单个成员节点的多个感知扇区分别覆盖多个目标位置时,仍然可以选取合适的冲突扇区来保证目标位置的覆盖,因此,TCCA算法的目标位置覆盖率优于TCDC算法.

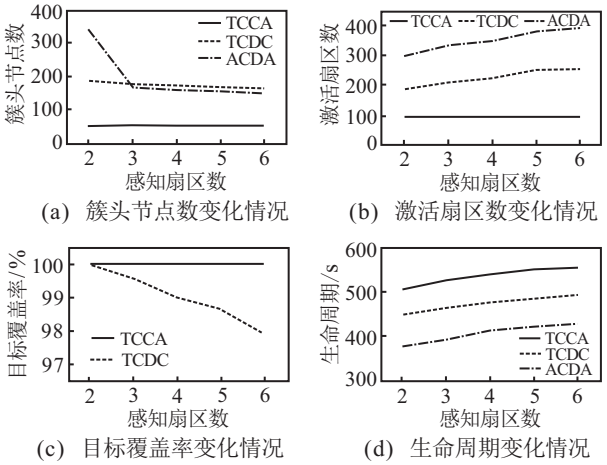
4) 生命周期.

如图3(d)所示,网络的生命周期都随着网络中部署节点数的增加而增大,TCCA算法的网络生命周期最大,TCDC次之,而ACDA算法的网络生命周期最短. 在算法初步完成之后,TCCA算法中节点所剩能量最多,其构成的簇头节点数和所选的激活感知扇区数都低于TCDC和ACDA算法,而且TCCA算法在CHS值的计算中考虑了对不覆盖目标位置的簇头节点的选取,从而能够有效减小簇头节点的更换频率,减少不必要的信息交换,进而延长网络的生命周期.

3.3.2 感知扇区数的影响

1) 簇头节点数.

如图4(a)所示,因为TCCA算法中节点采用全向通信模型,而TCDC和ACDA算法中节点的通信扇区数和感知扇区数一致,所以TCCA所构成的簇头节点数不受扇区数的影响,且少于TCDC和ACDA算法.



(a) 簇头节点数变化情况 (b) 激活扇区数变化情况

(c) 目标覆盖率变化情况 (d) 生命周期变化情况

图 4 感知扇区数影响

2) 激活扇区数.

图4(b)表明了各个算法中簇内覆盖集的激活扇区数随着感知扇区数的变化情况. 可以看出,随着感知扇区数的变化,TCCA算法的激活扇区数并没有太大变化,这可以归因于TCCA算法所构成的簇并不会随着感知扇区数的变化而变化,所以各个簇内所选取的最优覆盖集中激活感知扇区数也相对稳定,使得激活扇区数保持稳定.

3) 目标位置覆盖率.

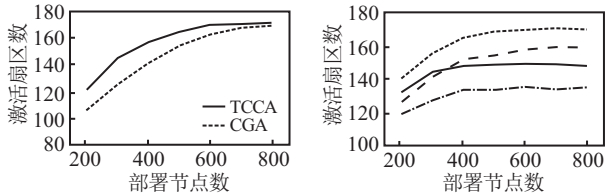
由前面的算法介绍可以看出,TCCA算法在簇内选取最优覆盖集时,首先考虑的是簇内所有目标位置的覆盖,其次尽量选取互不冲突的感知扇区加入覆盖集,所以TCCA算法所得到的覆盖集总是能保证簇内所有目标位置的覆盖,图4(c)也验证了这一点.

4) 生命周期.

如图4(d)所示,TCCA、TCDC和ACDA算法的生命周期都随着感知扇区数的增加而增大,且TCCA算法的生命周期大于其他两个算法. 通过前面的分析可以看出,TCCA算法的运算复杂度和交换的消息数量都小于TCDC和ACDA算法. 因此,运用TCCA算法构建的有向感知网络,节点能够更好地利用自身的能量完成对网络内目标位置的覆盖.

3.3.3 其他比较

下面主要分析TCCA算法与集中式算法CGA[12]的差异以及节点感知半径R_s和节点感知扇区数sn异构对TCCA算法的影响,见图5. 由图5(a)可以看出,由于CGA是集中式算法,其所需激活扇区数要小于TCCA算法,在部署节点数达到一定值时所需激活扇区数接近,这是由于部署节点数多时TCCA算法所选取的覆盖集更加理想. 若部署节点的感知半径和扇区数不同,则假设其分别在$[20, 40, 60, 80, 100]$和$[1, 2, 3, 4, 5]$中随机取值,其他参数如表1所示. 从图5(b)中可以看出(其中: 点线表示R_s和sn都相同; 虚线表示R_s相同、sn不同; 实线表示R_s不同、sn相同; 点划线表示R_s和sn都不同),感知半径和感知扇区数都相同时,所需激活扇区数最多,而两者都不同时所需激活扇区数最少. 这都是节点感知能力异构所带来的正常影响,说明TCCA算法对异构网络同样适用.



(a) 激活扇区数变化情况(1)　(b) 激活扇区数变化情况(2)

图5　部署节点数影响

4 结　论

本文提出的针对目标位置覆盖的分簇算法TCCA,在选取簇头节点时综合考虑了节点剩余能量、邻居节点数和功能的单一性,在簇内最优覆盖集的选择时能够在保证簇内目标位置覆盖的前提下尽量选取互不冲突的感知扇区,从而使得节点的能耗减少,延长网络的生命周期. 理论分析和仿真实验验证了改进方法能够较好地解决目标位置的覆盖问题,其计算复杂度以及运行的有效性都高于已有的算法.

参考文献(References)

[1] Guvensan M A, Yavuz A G. On coverage issues in directional sensor networks: A survey[J]. Ad Hoc Networks, 2011, 9(7): 1238-1255.

[2] Wang B. Coverage problems in sensor networks: A survey[J]. ACM Computing Surveys(CSUR), 2011, 43(4): 32-53.

[3] 赵玫, 杨洪勇,李路伟. 基于权重的目标覆盖控制算法[J]. 控制与决策, 2014, 29(10): 1845-1850.
(Zhao M, Yang H Y, Li L W. Target coverage control algorithm based on weight[J]. Control and Decision, 2014, 29(10): 1845-1850.)

[4] 王力立,吴晓蓓. 传感器网络中陷阱空洞的分布式检测及修复[J]. 控制与决策, 2012, 27(12): 1810-1815.
(Wang L L, Wu X B. Decentralized detection and patching of trap coverage holes for sensor networks[J]. Control and Decision, 2012, 27(12): 1810-1815.)

[5] 丁旭, 吴晓蓓,黄成. 基于改进粒子群算法和特征点集的无线传感器网络覆盖问题研究[J]. 电子学报, 2016, 44(4): 967-973.
(Ding X, Wu X B, Huang C. Area coverage problem based on improved PSO algorithm and feature point set in wireless sensor networks[J]. Acta Electronical Sinica, 2016, 44(4): 967-973.)

[6] Wang Y, Cao G. Barrier coverage in camera sensor networks[C]. Proc of the 12th ACM Int Symposium on Mobile Ad Hoc Networking and Computing. Paris: ACM, 2011: 12-21.

[7] Sharmin S, Nur F N, Razzaque M A, et al. Network lifetime aware area coverage for clustered directional sensor networks[C]. Int Conf on Networking Systems and Security(NSysS). Dhaka: IEEE, 2015: 1-9.

[8] Yu Z, Teng J, Bai X, et al. Connected coverage in wireless networks with directional antennas[J]. ACM Trans on Sensor Networks(TOSN), 2014, 10(3): 51-79.

[9] Chen Y-C ,Wen C-Y. Distributed clustering with directional antennas for wireless sensor networks[J]. Sensors J of IEEE, 2013, 13(6): 2166-2180.

[10] Cai Y, Lou W, Li M, et al. Energy efficient target-oriented scheduling in directional sensor networks[J]. IEEE Trans on Computers, 2009, 58(9): 1259-1274.

[11] Islam M M, Ahasanuzzaman M, Razzaque M A, et al. Target coverage through distributed clustering in directional sensor networks[J]. EURASIP J on Wireless Communications and Networking, 2015, 2015(1): 1-18.

[12] Ai J, Abouzeid A A. Coverage by directional sensors in randomly deployed wireless sensor networks[J]. J of Combinatorial Optimization, 2006, 11(1): 21-41.

[13] Chen A, Kumar S, Lai T H. Designing localized algorithms for barrier coverage[C]. Proc of the 13th Annual ACM Int Conf on Mobile Computing and Networking. Montréal: ACM, 2007: 63-74.

（责任编辑：李君玲）