

# 基于混合蛙跳的无线传感器网络定位方法

江宇波, 赵攀, 邱玲

(四川理工学院计算机学院, 四川 自贡 643000)

**摘要:**为了解决无线传感器网络未知节点的定位问题,提出了一种新的三维空间定位方法。首先给出了未知节点位置的计算方法和误差评价模型,并利用混合蛙跳算法建立了评价模型的求解算法 SFLL。最后,利用仿真实验,对比了与其它算法之间的性能状况,结果表明 SFLL 具有较好的适应性。

**关键词:**无线传感器网络;定位;最小二乘法;混合蛙跳

**中图分类号:**TP393

**文献标志码:**A

## 引言

随着无线传感器网络(Wireless Sensor Network, WSN)的发展和物联网技术的兴起,对未知节点的定位逐渐成为研究的热点和重点<sup>[1-2]</sup>。目前,定位算法可以分为距离和非距离两种方法。距离定位算法根据未知节点与锚节点的距离,并结合三边或者多边角方法来确定未知节点位置,主要有 RSSI 算法<sup>[3]</sup>、TOA 算法<sup>[4]</sup>等。而基于非距离定位算法则根据未知节点与锚节点之间的邻接关系来定位,典型代表有质心算法<sup>[5]</sup>、MDS - MAP 算法<sup>[6]</sup>等。最近,基于距离的定位算法又有了更新的发展,如最小二乘定位算法<sup>[7]</sup>,采用负梯度搜索等优化算法进行定位,不仅所需锚节点比例较少,而且定位精度更高。

在上述工作的基础上,本文提出了一种新的三维空间的定位算法和误差评价模型。该模型利用混合蛙跳算法对评价函数进行求解,同时通过数学仿真对比研究了该算法与 RSSI(Received Signal Strength Indicator)算法、TOA(Time Of Arrival)算法之间的性能状况,以此验证该模型的有效性。

## 1 节点定位方法

假设某无线传感器网络中存在  $N$  个节点,其中  $M$  个

为锚节点,坐标信息通过 GPS 定位系统确定,如图 1 所示,其中  $M$  代表锚节点,  $N$  代表未知节点。由于受到能量、成本等因素限制,只有少量节点可以成为锚节点,其余节点需要通过锚节点进行定位。距离定位算法<sup>[8-9]</sup>利用三边法或最大似然估计法,并结合信号强度来获取未知节点与锚节点之间的距离,但此方法只适用于锚节点个数大于 3 的情况。因此,本文基于三维空间提出了一种新的无线传感器网络节点定位算法,其思路是:首先根据能量和传输距离将网络节点划分成多个簇,其次根据每个簇内的锚节点对未知节点进行定位。

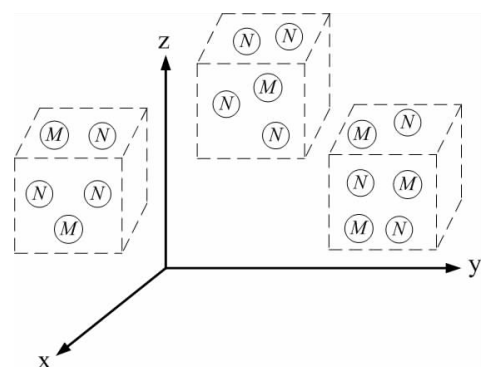


图 1 无线传感器网络拓扑结构

在初始情况下,令所有节点功能相同、能量有限、通

收稿日期:2014-05-05

基金项目:四川省教育厅重点项目(13ZA0118);人工智能四川省重点实验室开放基金项目(2012RYY02);四川理工学院培育项目(2012PY13);企业信息化与物联网检测技术四川省高校重点实验室项目(2013WYJ01)

作者简介:江宇波(1982-),男,四川自贡人,助理实验师,硕士,主要从事计算机网络通信方面的研究,(E-mail)598256382@qq.com

信功率可调。首先将节点划分成簇,簇内节点之间均采用直接通信方式,而簇间则可以选择直接或间接等方式通信,以达到能耗最低和效率最高的目的。根据 LEACH 协议,需要在各簇中产生簇首节点。对于各节点随机生成(0,1)之间的常数  $\varepsilon$  及阈值  $J$ , 如果  $\varepsilon < J$ , 则将当前节点加入候选簇首集合并向周围节点进行广播。如果在相邻区域内存在多个候选簇首节点,那么比较当前这些节点的剩余能量,剩余能量大者作为簇首节点,其余为普通节点。这里,阈值  $J$  定义为:

$$J = \frac{(\ln(1 + \varphi \frac{w(E_e + \eta d^2)}{E_0}))((1 - \lambda))}{(1 - p)\ln(1 + \varphi)} \quad (1)$$

其中,  $p$  为簇首节点占总节点的比例,  $E_0$  为节点  $i$  的初始能量,  $\lambda$  为距离比例,  $d_m$  为最大阈值,  $\varphi$  为常数 ( $0 < \varphi < 1$ ),  $E_e$  为发送或接收 1 比特信息时所消耗的能量,  $\eta$  为在单位距离内的信道上传输 1 比特信息时所消耗的能量,  $d$  为节点之间相距距离。

此时,按照产生的簇首节点集合,以各簇首节点为中心、 $R$  为半径的球体来划分簇结构。如果某节点位于多个球体交界处,那么以距某簇首节点距离最短作为划分标准,如图 1 中虚线代表簇边界。

假设簇内数据转发只需 1 跳,而转发到相邻簇则需要 2 跳。由各个簇内锚节点的分布状况,定位未知节点位置可分为以下几种情况:

#### (1) 簇内存在 2 个以上锚节点

此时,可以通过最小二乘法进行求解。假设未知节点  $N$  的坐标为  $(x, y, z)$ , 锚节点  $M_i (i = 1, 2, \dots, k, k \geq 3)$  的坐标为  $(x_i, y_i, z_i)$ , 那么未知节点  $Node$  与锚节  $M_i$  之间的测量距离  $r_i$  满足:

$$\begin{cases} \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} = r_1 \\ \sqrt{(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2} = r_2 \\ \dots \\ \sqrt{(x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2} = r_k \end{cases} \quad (2)$$

由于存在多于 3 个已知锚节点信息,对上述方程进行求解可得到如下形式:

$$A = (\alpha^T \beta)^{-1} \alpha^T \beta \quad (3)$$

#### (2) 簇内存在 2 个锚节点

假设这 2 个锚节点坐标分别为  $M_1(x_1, y_1, z_1)$  和  $M_2(x_2, y_2, z_2)$ , 根据公式(2)计算它与未知节点之间的测量距离  $r_1$  和  $r_2$ 。同时分别以这两个锚节点为中心、 $r_1$  和  $r_2$  为半径的作出两球区域,在连接球心的轴线上,取其中心点位置为新增临时锚节点  $M_3(x_3, y_3, z_3)$ , 当两球

体相交时,其坐标为:

$$\begin{cases} x_3 = 0.5(2r_1 - x_2 - r_2) \\ y_3 = 0.5(2r_1 - y_2 - r_2) \\ z_3 = 0.5(2r_1 - z_2 - r_2) \end{cases} \quad (4)$$

当两球体不相交时:

$$\begin{cases} x_3 = 0.5(x_1 + x_2 + r_1 - r_2) \\ y_3 = 0.5(y_1 + y_2 + r_1 - r_2) \\ z_3 = 0.5(z_1 + z_2 + r_1 - r_2) \end{cases} \quad (5)$$

根据锚节点  $M_1(x_1, y_1, z_1)$ 、 $M_2(x_2, y_2, z_2)$  和  $M_3(x_3, y_3, z_3)$  信息,利用标准最小二乘法可获得未知节点坐标信息。

#### (3) 簇内存在 2 个以下锚节点

此时由于锚节点距离未知节点较远,只能利用上述 2 个锚节点定位方法进行计算,其定位误差将增大。

## 2 算法描述

对于上述方法,其关键在于求解测量距离  $r_i$ 。但是由于 GPS 定位误差以及计算结果的近似处理,未知节点位置会存在一定偏差。假设位置节点与锚节点之间实际距离为  $d_i$ , 误差为  $\varepsilon_i$ , 那么满足  $|d_i - r_i| \leq \varepsilon_i$ 。  $\varepsilon_i$  越小代表定位精度越高,当  $\varepsilon_i \rightarrow 0$  时,定位算法获得最优解。因此,本文首先建立目标函数:

$$F = \min \left( \sum_{i=1}^k (d_i - r_i) \right) \quad (6)$$

其约束条件为:

$$\begin{cases} |d_1 - r_1| \leq \varepsilon_1 \\ |d_2 - r_2| \leq \varepsilon_2 \\ \dots \\ |d_k - r_k| \leq \varepsilon_k \\ \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} = r_1 \\ \sqrt{(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2} = r_2 \\ \dots \\ \sqrt{(x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2} = r_k \\ d_k \geq 0, r_k \geq 0, \varepsilon_k \geq 0 \end{cases} \quad (7)$$

由此,本文结合混合蛙跳算法对上述方法进行求解。混合蛙跳算法是一种群体智能的生物进化算法<sup>[10-11]</sup>,它模拟了青蛙群体寻找食物时按族群分类进行信息交换的过程,算法将群体分割为多个族群,在族群内部和族群之间进行更新操作。各族群内部搜索和全局信息交换一直持续交替进行到满足收敛条件或达到最大迭代次数为止。本文结合混合蛙跳算法给出目标

函数的求解算法(Shuffled Frog Leaping - based Localization algorithm, SFL),其步骤如下:

(1) 定义混合蛙跳算法的适应度函数来衡量当前解的优劣:

$$\varphi = \frac{1}{1 + F} \quad (8)$$

(2) 初始化,确定  $t = 0$  时 WSN 的各项参数。并令青蛙个体初始数目为  $a$ , 将目标函数  $F$  视作青蛙个体, 测量距离  $r_i$  视作表示青蛙个体移动步长, 最大迭代次数为  $MAXT$ ;

(3) 随机产生初始种群, 计算个体适应度。然后将个体按照适应度值大小进行降序排列, 并按以下原则分配到  $s$  个族群中: 第 1 个个体放入第 1 个族群, 第 2 个个体放入第 2 个族群, ..., 第  $s$  个个体放入第  $s$  个族群, 第  $s + 1$  个个体放入第 1 个族群, 这样一直循环, 直至所有个体分配完毕;

(4) 在每一个族群中, 适应度最优解和最差解是以其适应函数值的最大和最小值来确定的, 记为  $F_{opt}(s)$  和  $F_{wst}(s)$ , 并将所有族群  $F_{opt}(s)$  的最大值作为种群的全局最优解  $F_{opt}$ ;

(5) 在每个族群内部进行如式的内部搜索:

$$rnew_i = \alpha r_i + (1 - \alpha)(F_{opt} - F_{wst}) \quad (9)$$

其中,  $\alpha$  为  $(0, 1)$  之间的系数,  $rnew_i$  表示青蛙个体更新之后的移动步长。如果  $F_{wst}(s)$  发生改变, 则使用新的  $F_{wst}(s)$ , 否则将  $F_{opt}(s)$  换成  $F_{opt}$  来重新执行内部搜索; 如果仍没有改变, 则随机产生  $(F_{wst}(s), F_{opt}(s))$  之间的任意数来替换  $F_{wst}(s)$ ;

(6) 当所有族群内部更新完成后, 令  $t = t + 1$  并跳转到步骤(3), 重新划分族群和执行更新操作, 直到达到最大迭代次数  $MAXT$ ;

(7) 输出当前最优解, 即为预测误差极小值;

(8) 算法结束。

### 3 仿真实验

对于本文提出的 SFL 算法, 利用 NS2 和 MATLAB 进行仿真实验来验证其有效性。这里基于 NS2 建立如图 1 所示的仿真拓扑结构, 设置三维空间大小为  $100 \text{ m} \times 100 \text{ m} \times 100 \text{ m}$ , 节点数目 100 个, 并划分为 15 个栅格, 测距半径为 10 m。同时令青蛙个体初始数目为 100 个, 共有 15 个族群, 最大迭代次数为 200。这里将本文提出的 SFL 算法与传统的 RSSI (Received Signal Strength Indicator) 算法、TOA (Time Of Arrival) 算法进行对比。

首先研究平均定位误差与锚节点通信半径之间的关系。图 2 给出了这三种算法对应的平均定位误差与锚节点通信半径之间的关系。由图 2 可以看出, 随着锚节点通信半径的增加, 平均定位误差逐渐减低, 直至平稳。这与通常理解是一致的, 若锚节点通信半径可以增大到整个区域空间, 那么该区域内将不存在盲区, 那么平均定位误差趋于 0。而当达到相同平均定位误差时, SFL 算法所需的锚节点通信半径更小。

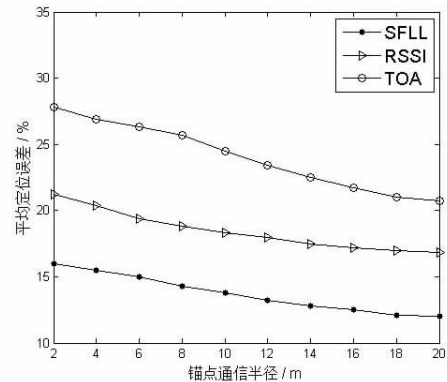


图 2 平均定位误差与锚节点通信半径之间的关系

其次, 对比分析锚节点数量对平均定位误差的影响情况(图 3)。令  $\rho$  表示锚节点数量与总节点数量的比值, 从图 3 可以看出, 当  $\rho$  增加时平均定位误差呈现递减趋势, 这与通常的理解一致。加大锚节点在空间区域中的密度, 使得网络中的盲点区域减少, 可以获得更大的定位精度, 但是此时将会加大系统开销成本。所以在实际应用中, 应综合各种因素来考虑锚节点密度。并且从图 3 中可以看出, 在相同密度下, SFL 算法计算得到的平均定位误差最小, 而 TOA 算法精度最差, 说明 SFL 算法具有一定的精度优势。

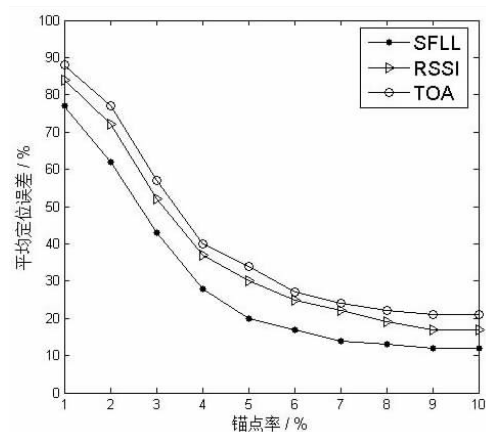


图 3 不同锚节点率下平均定位误差对比

最后, 深入分析影响 SFL 算法的关键因素。图 4

给出了不同青蛙簇群数量  $s$  下平均定位误差的变化趋势。在仿真初始阶段,簇群数越大对应的平均定位误差越小,而在仿真后期情况发生突变,簇群数越大对应的平均定位误差越大。这是因为初始阶段青蛙个体性能普遍较低,此时簇群数越大即意味着簇内个体越小,个体性能更新速度将越快;而在仿真后期,随着整体性能的提供啊,此时簇群数越小可以有效扩大个体寻优的目标,避免陷入局部最优。

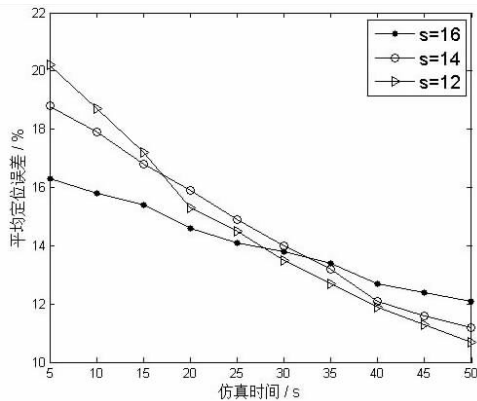


图 4 不同青蛙簇群数量下平均定位误差的变化

#### 4 结束语

为了有效解决无线传感器网络中未知节点定位精度的问题,本文结合最小二乘法建立了三维空间的定位算法 SFLL。首先将 WSN 节点划分成簇,并根据簇内锚节点数量给出了未知节点坐标的具体计算方法和误差评价模型,同时利用混合蛙跳算法对评价模型进行求解。最后通过进行数学仿真,深入研究了该算法与其他算法之间的性能差异,结果表明 SFLL 具有较好的适应性。

## Node Localization Method of Wireless Sensor Network Based on Shuffled Frog Leaping

JIANG Yubo, ZHAO Pan, QIU Ling

(School of Computer Science, Sichuan University of Science & Engineering, Zigong 643000, China)

**Abstract:** In order to mitigate the node localization accuracy in wireless sensor network, a new three-dimensional localization method is proposed. At first, the calculation method and error evaluation model of unknown node is presented. Then, an evaluation algorithm SFLL is proposed by shuffled frog leaping. At last, a mathematic simulation is conducted. Compared to the performances of other algorithms, the results show that SFLL has better adaptability.

**Key words:** wireless sensor network; localization; Least Squares; Shuffled Frog Leaping

#### 参考文献:

- [1] Wang W, Srinivasan V, Wang B, et al. Coverage for target localization in wireless sensor networks[J]. IEEE Transactions on Wireless Communications, 2008,7(2): 667-676.
- [2] 冯 晨.无线传感器网络定位精度优化方法研究[D]. 南京:南京邮电大学,2013.
- [3] 詹 杰,刘宏立,刘述钢,等.基于 RSSI 的动态权重定位算法研究[J].电子学报,2011,39(1):82-88.
- [4] 孟令军,王宏涛,夏善红.WSN 节点声测距 TOA 值频域估计方法[J].电子与信息学报,2010,32(4):993-997.
- [5] 宫娜娜,王缓缓.无线传感器网络质心算法的最佳通信半径[J].计算机仿真,2013,30(5):203-207.
- [6] 马 震,刘 云,沈 波.分布式无线传感器网络定位算法 MDS-MAP(D)[J].通信学报,2008,29(6):57-62.
- [7] 匡兴红,邵惠鹤.基于传感器网络的气体源定位方法研究[J].系统仿真学报,2007,19(7):1464-1467.
- [8] 丁卫平,王建东,管致锦.基于量子蛙跳协同进化的粗糙属性快速约简[J].电子学报,2011,39(11):2597-2603.
- [9] 杨维剑,王梅英.无线网络传感器中超低功耗节点能源技术研究[J].四川理工学院学报:自然科学版, 2010,23(1):44-47.
- [10] 葛 宇,王学平,梁 静.基于蛙跳算法的 DV - Hop 定位改进[J].计算机应用,2011,31(4):922-924.
- [11] 葛 宇,梁 静,许 波,等.基于蛙跳算法的无线传感器网络节点定位[J].计算机工程与应用,2012,48(20): 126-130.