申报	系列：教师系列 教学科研并重型
	专业：计算机科学与技术
	职称：副教授

业绩成果材料

单　位（二级单位）数学与信息学院（软件学院）

姓　名　　**李双娟**

材料核对人：　　单位盖章：

核对时间：

华南农业大学制

目　录

一、科研项目

1、国家自然科学基金项目《移动有向传感器网络的栅栏覆盖研究》的立项通知及有关佐证材……4

2、国家自然科学基金项目《基于异构二部图模型的多源大规模数据聚类集成算法研》的立项通知及有关佐证……6

二、论文、著作等

1、检索证明……10

2、学术论文《Minimizing maximum movement of sensors for line barrier coverage in the plane》……12

3、学术论文《Optimizing the Sensor Movement for Barrier Coverage in a Sink-Based Deployed Mobile Sensor Network》……25

4、学术论文《Reducing Sensor Failure and Ensuring Scheduling Fairness for Online Charging in Heterogeneous Rechargeable Sensor Networks》……39

5、学术论文《Exact algorithms for barrier coverage with line-based deployed rotatable directional sensors》……42

8、教材《Python 语言程序设计教程》……52

三、其他业绩

1.指导学生参加蓝桥杯大赛获奖.....55

关于国家自然科学基金资助项目批准及有关事项的通知

李双娟　先生/女士：

根据《国家自然科学基金条例》的规定和专家评审意见，国家自然科学基金委员会（以下简称自然科学基金委）决定批准资助您的申请项目。项目批准号：61702198，项目名称：移动有向传感器网络的栅栏覆盖研究，直接费用：22.00万元，项目起止年月：2018年01月至2020年12月，有关项目的评审意见及修改意见附后。

请尽早登录科学基金网络信息系统（https://isisn.nsfc.gov.cn），获取《国家自然科学基金资助项目计划书》（以下简称计划书）并按要求填写。对于有修改意见的项目，请按修改意见及时调整计划书相关内容；如对修改意见有异议，须在计划书电子版报送截止日期前提出。**注意：请严格按照《国家自然科学基金资助项目资金管理办法》填写计划书的资金预算表，其中，劳务费、专家咨询费科目所列金额与申请书相比不得调增。**

计划书电子版通过科学基金网络信息系统（https://isisn.nsfc.gov.cn）上传，由依托单位审核后提交至自然科学基金委进行审核。审核未通过者，返回修改后再行提交；审核通过者，打印为计划书纸质版（一式两份，双面打印），由依托单位审核并加盖单位公章后报送至自然科学基金委项目材料接收工作组。计划书电子版和纸质版内容应当保证一致。

向自然科学基金委提交和报送计划书截止时间节点如下：

1、提交计划书电子版截止时间为**2017年9月11日16点**（视为计划书正式提交时间）；

2、提交计划书电子修改版截止时间为**2017年9月18日16点**；

3、报送计划书纸质版截止时间为**2017年9月26日16点**。

请按照以上规定及时提交计划书电子版，并报送计划书纸质版，未说明理由且逾期不报计划书者，视为自动放弃接受资助。

附件：项目评审意见及修改意见表

国家自然科学基金委员会

信息科学部

2017年8月17日

51064201-0932

国家自然科学基金

资助项目准予结题通知

李双娟　同志：

您承担的国家自然科学基金项目：(移动有向传感器网络的栅栏覆盖研究)，批准号：(61702198)按有关规定已审核完毕，准予结题。

与本项目资助有关的后续成果，请您继续及时报送。

祝您在研究工作中取得更好的成绩！

国家自然科学基金委员会

信息科学部

2021年03月29日

国家自然科学基金资助项目批准通知

黄栋　　先生/女士：

根据《国家自然科学基金条例》和专家评审意见，国家自然科学基金委员会（以下简称自然科学基金委）决定批准资助您的申请项目。项目批准号：61976097，项目名称：基于异构二部图模型的多源大规模数据聚类集成算法研究 ，直接费用：61.00万元，项目起止年月：2020年01月至 2023年 12月，有关项目的评审意见及修改意见附后。

请尽早登录科学基金网络信息系统（https://isisn.nsfc.gov.cn），获取《国家自然科学基金资助项目计划书》（以下简称计划书）并按要求填写。对于有修改意见的项目，请按修改意见及时调整计划书相关内容；如对修改意见有异议，须在电子版计划书报送截止日期前向相关科学处提出。

电子版计划书通过科学基金网络信息系统（https://isisn.nsfc.gov.cn）上传，依托单位审核后提交至自然科学基金委进行审核。审核未通过者，返回修改后再行提交；审核通过者，打印纸质版计划书（一式两份，双面打印），依托单位审核并加盖单位公章后报送至自然科学基金委项目材料接收工作组。电子版和纸质版计划书内容应当保证一致。向自然科学基金委提交和报送计划书截止时间节点如下：

1、提交电子版计划书截止时间为**2019年9月11日16点**（视为计划书正式提交时间）；

2、提交电子修改版计划书截止时间为**2019年9月18日16点**；

3、报送纸质版计划书截止时间为**2019年9月26日16点**。

请按照以上规定及时提交电子版计划书，并报送纸质版计划书，未说明理由且逾期不报计划书者，视为自动放弃接受资助。

附件：项目评审意见及修改意见表

国家自然科学基金委员会

2019年8月16日

项目批准号	61976097
申请代码	F060204
归口管理部门	
依托单位代码	51064208A0499-0932

国家自然科学基金委员会

资助项目计划书

资助类别：面上项目

亚类说明：

附注说明：

项目名称：基于异构二部图模型的多源大规模数据聚类集成算法研究

直接费用：61万元　　执行年限：2020.01-2023.12

负 责 人：黄栋

通讯地址：广东省广州市天河区五山华南农业大学数学与信息学院610室

邮政编码：510640　　电　　话：020-85280320-610

电子邮件：huangdong06@163.com

依托单位：华南农业大学

联 系 人：倪慧群　　电　　话：020-85280070

填表日期：2019年08月21日

国家自然科学基金委员会制

简表

申请者信息	姓　　名	黄栋	性 别	男	出生年月	1987年11月	民 族	汉族
	学　　位	博士			职称	副教授		
	是否在站博士后	否		电子邮件		huangdong06@163.com		
	电　　话	020-85280320-610		个人网页				
	工 作 单 位	华南农业大学						
	所 在 院 系 所	数学与信息(软件)学院						
依托单位信息	名　　称	华南农业大学					代码	51064208A0499
	联 系 人	倪慧群		电子邮件		kycjhk@scau.edu.cn		
	电　　话	020-85280070		网站地址		http://kjc.scau.edu.cn/		
合作单位信息	单 位 名 称							
项目基本信息	项 目 名 称	基于异构二部图模型的多源大规模数据聚类集成算法研究						
	资 助 类 别	面上项目			亚 类 说 明			
	附 注 说 明							
	申 请 代 码	F060204:无监督学习			F060206:集成学习			
	基 地 类 别							
	执 行 年 限	2020.01-2023.12						
	直 接 费 用	61万元						

项目组主要成员

编号	姓名	出生年月	性别	职称	学位	单位名称	电话	证件号码	项目分工	每年工作时间（月）
1	黄栋	1987.11	男	副教授	博士	华南农业大学	020-85280320-610	441622198711022098	项目负责人	10
2	崔金荣	1984.12	女	讲师	博士	华南农业大学	18588528183	410927198412094042	视频特征提取	5
3	李双娟	1983.10	女	讲师	博士	华南农业大学	13729829618	430482198310260047	图模型设计与优化	5
4	王国华	1988.04	男	讲师	博士	华南农业大学	13602436769	441781198804130114	多源关联分析与建模	5
5	徐莎莎	1994.12	女	硕士生	学士	华南农业大学	13725051754	430528199412243063	二部图构建与分析	10
6	梁健恒	1994.10	男	硕士生	学士	华南农业大学	15521165257	442000199410240934	多源癌症基因数据分析	10
7	何囝囡	1994.08	女	硕士生	学士	华南农业大学	18673880415	432524199408061925	多源数据特征提取	10
8	钟景润	1995.07	男	硕士生	学士	华南农业大学	18819258392	44162119950728323X	视频数据采集与预处理	10
9	李泽轩	1995.08	男	硕士生	学士	华南农业大学	15019762183	440582199508036616	社交网络数据采集与预处理	10

总人数	高级	中级	初级	博士后	博士生	硕士生
9	1	3	0	0	0	5

SCAULIB202626896

检索证明

根据委托人提供的论文材料，委托人华南农业大学数学与信息学院 李双娟(学科类型:自然科学) 2 篇论文收录情况如下表。

序号	论文名称	发表刊物及发表的年月卷期/页码等	作者排名	论文等级	作者文中单位	收录情况	影响因子	中科院大类分区	引用
1	Minimizing maximum movement of sensors for line barrier coverage in the plane	COMPUTER NETWORKS 出版年：2019 出版日期：NOV 9 卷期：163 页码：- 文献号：106841 文献类型：Article	第一作者	B 类	华南农业大学数学与信息学院	SCI	IF2-year=3.111 IF5-year=3.146 （2019）	计算机科学 3 区 Top 期刊：否 OA 期刊：否 （2019）	SCI 核心合集 总引：6
2	Optimizing the Sensor Movement for Barrier Coverage in a Sink-Based Deployed Mobile Sensor Network	IEEE ACCESS 出版年：2019 卷期：7 页码：156301-156314 文献类型：Article	第一作者	A 类	华南农业大学数学与信息学院	SCI	IF2-year=3.745 IF5-year=4.076 （2019）	计算机科学 2 区 Top 期刊：否 OA 期刊：是 （2019）	SCI 核心合集 总引：2

说明：论文等级和中科院大类分区按《华南农业大学学术论文评价方案（试行）》划分。

报告免责声明:如未盖章,报告无效

检索员：尹银怀

华南农业大学图书馆

2026-04-08

检索证明

根据委托人提供的论文材料，委托人**华南农业大学李双娟**2篇论文收录情况如下表。

序号	论文名称	发表刊物及发表的年月卷期/页码等	作者排名	论文等级	作者文中单位	收录情况	影响因子	中科院大类分区
1	Exact algorithms for barrier coverage with line-based deployed rotatable directional sensors	IEEE Wireless Communications and Networking Conference, WCNC 出版年：2020 May 2020 卷期：2020-May 文献号：9120836 文献类型：Conference article (CA)	通讯作者	/	华南农业大学	EI		
2	Reducing Sensor Failure and Ensuring Scheduling Fairness for Online Charging in Heterogeneous Rechargeable Sensor Networks	Proceedings - IEEE Symposium on Computers and Communications 出版年：2020 July 2020 卷期：2020-July 文献号：9219569 文献类型：Conference article (CA)	通讯作者	/	华南农业大学	EI		

说明：论文等级和中科院大类分区按《华南农业大学学术论文评价方案（试行）》划分。

报告免责声明:如未盖章,报告无效

检索员：刘汉忠
华南农业大学图书馆
2022-04-21

Computer Networks 163 (2019) 106841

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Minimizing maximum movement of sensors for line barrier coverage in the plane

Shuangjuan Li[a], Hong Shen[b,c,*]

[a] College of Mathematics and Informatics, South China Agricultural University, China
[b] School of Data and Computer Science, Sun Yat-Sen University, China
[c] School of Computer Science, University of Adelaide, Australia

ARTICLE INFO

Article history:
Received 8 January 2018
Revised 6 June 2019
Accepted 24 June 2019
Available online 25 July 2019

Keywords:
Barrier coverage
Mobile sensor network
Minmax
Arbitrary sensing range
Uniform sensing range

ABSTRACT

Border surveillance for intrusion detection is an important application of wireless sensor networks (WSNs). Given a set of mobile sensors and their initial positions, how to move these sensors to a region border to achieve barrier coverage energy-efficiently is challenging. This paper studies the 2-D MinMax barrier coverage problem of moving n sensors in a two-dimensional plane to form a barrier coverage of a specified line segment in the plane while minimizing the maximum sensor movement for the sake of balancing battery power consumption. Previously, this problem was shown to be NP-hard for the general case when the sensing ranges are arbitrary. It was an open problem whether the problem is polynomial-time solvable for the case when sensors have a fixed number of sensing ranges. We first study the barrier coverage problem for the general case and propose a two-phase algorithm and a greedy algorithm. The approximation factor of the two-phase algorithm is proved to be $\sqrt{2}$ for the case when the sensors can exactly cover the barrier. We then study a special case of great practical significance that the sensors have the same sensing range and present an $O(n^2 \log n)$ time algorithm. We obtain $\Theta(n^3)$ candidate values for the minimum maximum sensor movement by deriving a necessary condition of the optimal sensor deployment and find the optimal sensor movement among them by using an efficient search procedure and a decision algorithm. To the best of our knowledge, this is the first result for solving the 2-D MinMax barrier coverage problem both for the general case and the uniform case.

1. Introduction

Border surveillance for intrusion detection is an important application category for wireless sensor networks. For example, sensors can be deployed along international borders to form a barrier for detecting illegal intruders as they cross the border. When only stationary sensors are used, after randomly deployment, it is difficult to form barriers for barrier coverage, allowing some intruders crossing the border without being detected. With recent technical advances, sensors equipped with a great degree of mobility (e.g., Packet [24] and Robomote [8]) have been developed. Mobile sensors can move to cover particular sections of the border as needed, enabling fewer sensors deployed to achieve barrier coverage than deploying stationary sensors. However, mobile sensors equipped with batteries with limited energy consume much more energy during the movement, which is known quadratic on the moving distance, than that during the communication or sensing process, thus it is very important to design an energy-efficient sensor mobility scheme that maximizes the lifetime of the mobile sensor network for achieving barrier coverage. Mobile sensors can also be used in the stationary sensor network to improve barrier coverage because they can move to cover the barrier gaps left by the stationary sensors after initial randomly deployment. In such a hybrid stationary-sensor network, it is also important to design an energy-efficient sensor mobility scheme for maximize the network lifetime.

Given a set of mobile sensors and their initial positions, how to move these sensors to a region border to achieve barrier coverage energy-efficiently is challenging, because the sensors can move to arbitrary points of the barrier line. As shown in Fig. 1(a), the work [17] studied how to move mobile sensors to the grid points of a barrier for achieving barrier coverage while minimizing the maximum sensor movement and solved it using the maximum flow algorithm. However, as shown in Fig. 1(b), in this paper, the sensors can move to arbitrary points of the barrier line for achieving barrier coverage rather than the grid points, which may reduce the sensor movement. Fig. 1 gives an example, which shows that the

* Corresponding author at: North Terrace, Adelaide, Australia.
E-mail addresses: lishj2013@hotmail.com (S. Li), hongsh01@gmail.com (H. Shen).

https://doi.org/10.1016/j.comnet.2019.06.019
1389-1286/

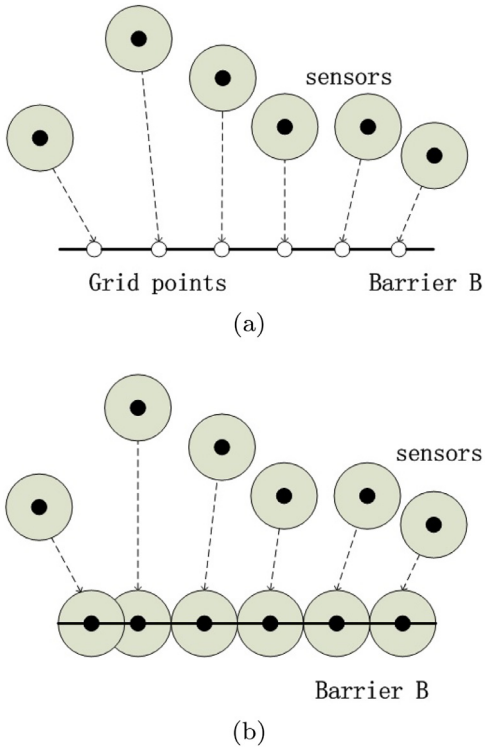


Fig. 1. (a) Sensors move to cover the grid points of barrier; (b) Sensors move to cover the barrier.

maximum sensor movement in Fig. 1(b) is reduced compared to that in Fig. 1(a).

This paper studies how to move the mobile sensors to cover a barrier represented by a line segment with the minimum maximum sensor movement, which is referred to as the 2-D MinMax problem. As discussed in [10], the 2-D MinMax problem has been shown to be NP-hard for the general case when the sensor ranges are arbitrary. Besides, it was proposed as an open question in [10] whether there is a polynomial time algorithm for the case when sensors have a fixed number of sensing ranges. It is also unknown whether the 2-D MinMax problem for the uniform case when the sensing ranges are uniform is polynomial-time solvable. This paper tries to solve the 2-D MinMax problem for the general case, and also this problem for the uniform case. Since the sensors can move to arbitrary points of the barrier line, there are infinite candidate values for the minimum maximum sensor movement. It is therefore challenging to find the minimum maximum movement. Besides, another challenging issue is also studied how to determine whether the sensors can move to cover the barrier when the sensors' maximum movement is not greater than a fixed value, which is referred to as the decision problem. It's challenging to determine the final positions of the sensors forming a barrier coverage. The algorithm of the decision problem can help to solve the 2-D MinMax problem.

This paper addresses the 2-D MinMax problem and also the decision problem. The main contributions of this paper are summarized as follows:

1) For the general case when the sensors' sensing ranges are arbitrary, the 2-D MinMax problem has been proved to be NP-hard in [10]. A two-phase algorithm and a greedy fast algorithm are presented. The approximation factor of the two-phase algorithm is proved to be $\sqrt{2}$ when the sensors can exactly cover the barrier.

2) For the uniform case when the sensing ranges are uniform, a greedy algorithm is proposed to solve the decision problem of determining whether the sensors can move to cover the barrier when the sensors' maximum movement is not larger than a fixed value.

3) For the case when the number of the sensors' sensing ranges is k (k is a constant), it was an open question in [10] whether there is a polynomial time algorithm for this MinMax problem. We show that there is an $O(n^2 \log n)$ time algorithm for a special case when $k = 1$. We can obtain $\Theta(n^3)$ candidate values for the minimum maximum sensor movement by deriving a necessary condition of the optimal sensor deployment and find the optimal value among them by using an efficient search procedure and a decision algorithm.

4) Extensive simulation experiments are conducted to evaluate the proposed algorithms. The results demonstrate that our algorithms are quite scalable and energy-efficient.

Our algorithms can be directly applied in various real applications, such as intrusion detection in the battlefield, boundary protection across protected areas and wildlife cross-zone activity monitoring. In these applications, since mobile sensors are usually air-dispersed and their deployment under a random distribution does not cover the targeted barrier initially in most cases, they need to move the desirable positions to form a barrier coverage. Our algorithms provide a way how these sensors shall move to cover the barrier represented by a line segment such that the maximum movement is minimized and hence the lifetime of the coverage (the time that the first sensor drops off) is maximized.

The rest of the paper is organized as follows. Section 2 reviews the related work is. In Section 3, the problem formulation and notation are presented. In Section 4, two algorithms are proposed to solve the 2-D MinMax problem for the general case. In Section 5, an algorithm is presented to solve the decision problem for the uniform case. In Section 6, a polynomial time algorithm is proposed to solve the 2-D MinMax problem for the uniform case. The performance of our algorithms are evaluated in Section 7. Section 8 concludes the paper.

2. Related work

Barrier coverage in stationary sensor network has been studied intensively in [2,12,14,18,20,22,25]. Since random sensor dispersal costs a large number of sensors, several researchers have started to study how to use fewer sensors with mobility for achieving barrier coverage in an energy-efficient way, especially minimizing some aspect of the relocation cost.

Several work studied the 1-D barrier coverage problem under the assumption that the barrier is a line segment and all sensors are initially located on the line containing the barrier. The work [6] first studied the problem of minimizing the maximum sensor movement (1-D MinMax) and presented an $O(n^2)$ time algorithm for the case when the sensor ranges are uniform. The key idea of the algorithm is to cover the gaps one by one from left to right, while balancing the costs of covering from left and right as much as possible. The work [3] improved this time complexity to $O(n \log n)$ for the uniform case, and also solved the problem for the general case when the sensing ranges are arbitrary by giving an $O(n^2 \log n)$ time algorithm. To help solve this problem, the work [3] also developed a greedy algorithm for the decision problem of determining whether the sensors can move to cover the barrier when the sensors' maximum movement is not larger than a fixed value, which takes $O(n \log n)$ time. The work [15] considered the problem of minimizing the number of sensors moved on the line (1-D MinNum), which is proved to be NP-hard for the general case while efficient algorithms are designed for the uniform case. The work [7] studied the problem of minimizing the sum of sensor movements (1-D MinSum), which is shown to be NP-complete by reduction to the 3-partition problem for the general case and also presented an $O(n^2)$ time algorithm for the uniform case.

Several work studied the 2-D barrier coverage problem under the assumption that the barrier is a line segment and all sensors are initially located in a 2-D plane containing the barrier. The 2-D MinMax problem was first studied in [10] and shown to be NP-hard for the general case. When sensor movements are limited to be perpendicular to the barrier, this problem can be solved in $O(n\log n)$ time by applying dynamic programming. Similar to the 2-D MinMax problem, the 2-D MinSum problem is also proved to be NP-hard for the general case and an solvable in $O(n^2)$ time for the perpendicular movement case. The work [13] studied the 2-D MinMax problem for the uniform case and proposed an $O(n^3\log n)$ time algorithm. The work [5] studied the 2-D MinSum problem for the uniform case and proposed a $\sqrt{2}$-approximation algorithm.

Other types of the barriers than straight lines, such as circles or simple polygons, were studied in [1]. To move the sensors from the interior of a monitored region to the region's perimeter that forms a barrier of the region, the MinMax problem can be solved in $O(n^{3.5}\log n)$ time for cycle barriers and $O(mn^{3.5}\log n)$ time for polygon barriers, where m is the number of edges of the simple polygon [1]. Besides, the MinSum problem can be solved by a PTAS. The work [19] improved the MinMax result for polygon barriers to $O(n^{2.5}\log n)$ and also presented an $O(n^4)$ time algorithm for the MinSum problem on the circle when the sensors are moved from the circle perimeter to a regular n-gon, which is an open question in [1].

Most of the previous work assumed that the positions of the barriers are pre-specified, but the work [16,17] studied the barrier coverage problem under the assumption that the positions of the barriers are not known a priori and presented a two-phase sensor mobility scheme, which was proved order optimal in achieving the maximum number of barriers while minimizing the maximum sensor moving distances. The work [27] studied the 2-D MinMax problem when the y-coordinate of the barrier is determined and proposed an efficient algorithm to find the optimal barrier location. The work [11] studied how to achieve barrier coverage in the sensor scarcity scenario by dynamic sensor patrolling and presented a coordinated sensor patrolling (CSP) algorithm.

The work [23] studied how to form a barrier using mobile sensors and stationary sensors. When the stationary sensors are predeployed, the mobile sensors are moved to fill in the coverage gaps. An algorithm was proposed to find the minimum number of mobile sensors required and also another algorithm is presented to minimize the cost of moving the sensors to fill in the gaps. The work [9] studied how to use the mobile sensors with adjustable sensing range to mend the gaps. It presented a scheme to maximize the lifetime of barrier coverage after mending the gaps.

3. Problem formulation

This section first formally describes the barrier coverage problems in mobile sensor network and then introduces relevant notation.

3.1. Problem statement

A barrier B is a line segment on the x-axis in the two-dimensional plane with starting point [0, 0] and ending point [L, 0]. A set of n mobile sensors $S=\{s_1, s_2, \ldots, s_n\}$ with the sensing range $\{r_1, r_2, \ldots, r_n\}$ is located in this plane with initial positions $\{p_1, p_2, \ldots, p_n\}$, where $p_i=(x_i, y_i)$. Let $R=\Sigma_{i=1}^{n} 2r_i (R \geq L)$. Each sensor of S is assumed to be mobile and can relocate itself from the initial position to another specified position. Suppose the final position of a mobile sensor s_i is $p_i'=(x_i', y_i')$. The movement of a sensor s_i is defined as the Euclidean distance between its initial position and its final position, denoted by $d_i=d(p_i, p_i')$.

This paper studies the problem of how to schedule the sensors to move to a barrier line for achieving barrier coverage while minimizing the maximum moving distance of sensors, which can balance the power consumption among the sensors thus prolonging the network lifetime. The formal definition of this problem is given as follows:

Definition 1. The 2-D MinMax barrier coverage problem (2-D MinMax problem): Given a barrier B and a set of sensors $S=\{s_1, s_2, \ldots, s_n\}$ with sensing range $\{r_1, r_2, \ldots, r_n\}$ and initial positions $\{p_1, p_2, \ldots, p_n\}$, the problem is to find the sensors' final positions $\{p_1', p_2', \ldots, p_n'\}$ on the barrier B so that each point of B is covered by at least one sensor and the maximum movement of the sensors i.e. $max_{1\leq i\leq n}\{d(p_i, p_i')\}$ is minimized.

This paper first studies the 2-D MinMax problem for the general case when the sensing ranges are arbitrary and then the 2-D MinMax problem for the uniform case when the sensing ranges are uniform.

To help design the 2-D MinMax problem for the uniform case, we also study the decision problem of determining whether the sensors can move to cover the barrier when the sensors' maximum movement is not larger than a fixed value. The formal definition of this problem is given as follows:

Definition 2. The decision problem of barrier coverage (The decision problem): Given a barrier B, a set of sensors $S=\{s_1, s_2, \ldots, s_n\}$ with the same sensing range r and initial positions $\{p_1, p_2, \ldots, p_n\}$ and a constant $\lambda \geq 0$, the problem is to determine whether there exist sensors' final positions $\{p_1', p_2', \ldots, p_n'\}$ on the barrier B so that each point of B is covered by at least one sensor and the maximum movement of the sensors is at most λ.

3.2. Notation

In this subsection some notation is presented which will be used in the following sections. Note that some of our notation is adopted from [3].

Let λ^* denote the minimum maximum movement of sensors in S obtained by the algorithm for the 2-D MinMax problem.

A sensor deployment scheme on λ denoted by $D(\lambda)$ is defined as a specification of the final positions of the sensors on the barrier B when the maximum sensor movement is λ.

A sensor deployment scheme is said to be a feasible sensor deployment scheme if each point of barrier line B is covered by at least one sensor in a sensor deployment scheme.

A feasible sensor deployment scheme is said to be an optimal sensor deployment if the maximum sensor movement of each sensor is minimized.

Given a positive constant λ as the maximum sensor movement, for each sensor $s_i \in S$, s_i's left and right endpoint of the moving range, short for s_i's left and right moving endpoint, is defined as the smallest and largest possible final position of sensor s_i on barrier B. Sensor s_i's left moving endpoint is denoted by $x_i - d_i$, while s_i's right moving endpoint is denoted by $x_i + d_i$, where $d_i = \sqrt{\lambda^2 - y_i^2}$. Fig. 2(a) shows s_i's left and right moving endpoint.

Given a positive constant λ as the maximum sensor movement, for each sensor $s_i \in S$, s_i's left and right extension of the moving range, short for s_i's left and right moving extension, is defined as the smallest and largest point of barrier B covered by sensor s_i when it is located at the left and right moving endpoint. Sensor s_i's left moving extension is denoted by $x_i - d_i - r_i$, while s_i's right moving extension is denoted by $x_i + d_i + r_i$, where $d_i = \sqrt{\lambda^2 - y_i^2}$. Fig. 2(b) shows s_i's left and right moving extension.

Sensor s_i is said to be in attaching position in $D(\lambda)$ if s_i is two sensing ranges distance away from the left adjacent sensor in $D(\lambda)$,

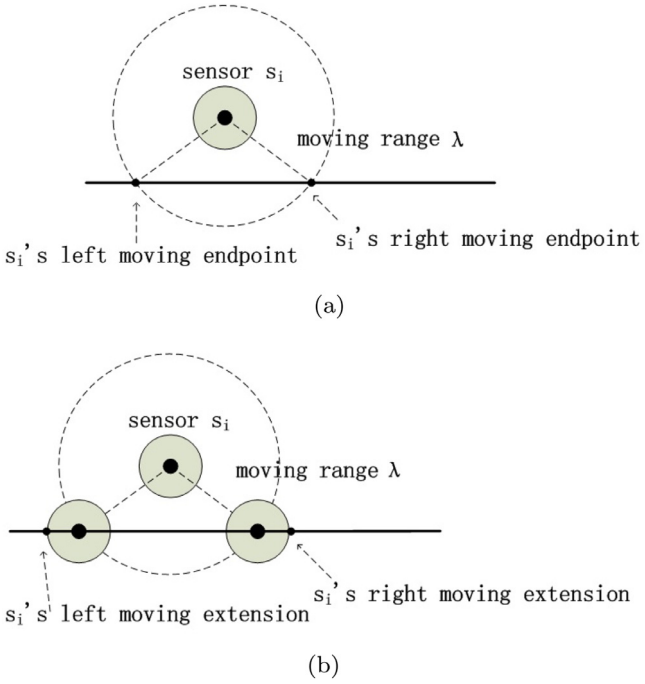


Fig. 2. (a) sensor s_i's left and right moving endpoint (b) sensor s_i's left and right moving extension.

or one sensing range distance away from the left endpoint of barrier B if no left adjacent sensor in $D(\lambda)$.

4. The 2-D minmax problem for the general case

This section studies the 2-D MinMax problem for the case when the sensing ranges are arbitrary. This problem has been proved to be NP-hard in [10] but no approximation algorithm has been designed so far. A two-phase algorithm is first proposed, whose approximation factor is $\sqrt{2}$ for a special case when sensing ranges of the sensors can exactly cover the barrier line together, that is $R = L$. Then a greedy heuristic algorithm is presented.

4.1. Two-phase algorithm

A two-phase sensor movement scheme is proposed to solve the 2-D MinMax problem for the general case. This scheme is based on the algorithm of BCLS problem in [3], which is called 1-D MinMax algorithm. The 1-D MinMax algorithm in [3] is to schedule the sensors located on the line containing the barrier line to move to the barrier line for barrier coverage while minimizing the maximum sensor movement.

In the first phase, sensors move to the line containing the barrier B in the vertical direction. The maximum sensor movement is denoted by y_m, which is the largest y-coordinate of the initial positions of all the sensors. The running time of this phase is $O(n)$.

In the second phase, sensors move to cover the barrier B in the horizontal direction by using the 1-D MinMax algorithm in [3]. Suppose d_m is the minimum maximum sensor movement computed by the 1-D MinMax algorithm. Since 1-D MinMax algorithm runs in $O(n^2 \log n)$ time, the running time of this phase is $O(n^2 \log n)$.

Fig. 3 shows an example of the two-phase sensor movement scheme.

This algorithm costs $O(n^2 \log n)$ time. The maximum sensor movement can be computed by this two-phase sensor movement scheme, which is $\sqrt{y_m^2 + d_m^2}$, since sensors can move from the initial positions to the final positions directly.

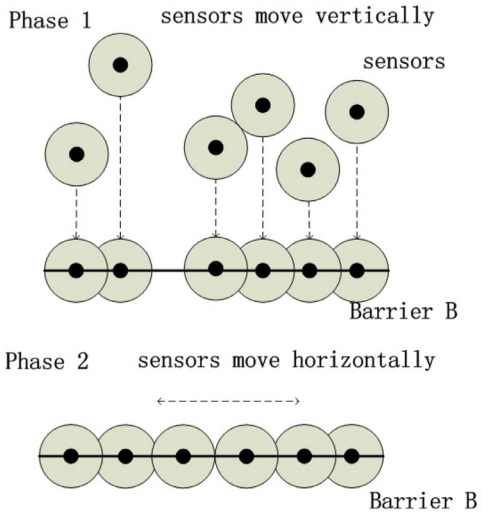


Fig. 3. The two-phase algorithm.

Below we'll prove that this two-phase algorithm is a good approximation algorithm for a special case when $R = L$.

Lemma 3. *The two-phase algorithm is a $\sqrt{2}$ -approximation algorithm for the 2-D MinMax problem when the sensing ranges are arbitrary and $R = L$.*

Proof. Let *OPT* denote the optimal algorithm for the 2-D MinMax problem and λ^* denote the maximum sensor movement by *OPT*. Let *SOL* denote the two-phase algorithm for the 2-D MinMax problem and $\bar{\lambda}$ denote the maximum sensor movement computed by *SOL*.

In the first phase of *SOL*, all the sensors should move to cover the barrier line since $R = L$. Thus it can be seen that

$$y_m \le \lambda^* \tag{1}$$

where y_m is the largest y-coordinate of the initial positions of all the sensors.

In the second phase of *SOL*, it can be proved that $d_m \le \lambda^*$, where d_m is the minimum maximum sensor movement computed by the 1-D MinMax algorithm in [3].

Let y_t denote the smallest y-coordinate of the initial positions of all the sensors. In the second phase, a feasible sensor deployment scheme can be obtained when the maximum sensor movement is $\sqrt{(\lambda^*)^2 - y_t^2}$. Since d_m is the optimal sensor movement in the second phase, $d_m \le \sqrt{(\lambda^*)^2 - y_t^2}$ holds. Thus it can be seen that

$$d_m \le \lambda^* \tag{2}$$

By inequality (1)(2), it can be seen that

$$\bar{\lambda} = \sqrt{y_m^2 + d_m^2} \le \sqrt{(\lambda^*)^2 + (\lambda^*)^2} = \sqrt{2}\lambda^*$$

Therefore, the Lemma is proved. □

4.2. Greedy algorithm

A faster greedy algorithm called the GreedyDiff algorithm is proposed to solve the 2-D MinMax problem for the general case. In this algorithm, we always choose the closest available mobile sensor to cover the leftmost uncovered point of the barrier line and put it in attaching position. This process continues until the whole of the barrier line is covered. The detail of the algorithm is described in Algorithm 1.

Algorithm 1 The GreedyDiff algorithm.

INPUT: S, L
OUTPUT: λ
1: Let $R_0 \leftarrow 0$, $i \leftarrow 0$, $S_c \leftarrow \phi$;
2: $\lambda \leftarrow 0$;
3: while $R_i < L$
4: $m \leftarrow infinity$;
5: for each sensor $s_j \in S/S_c$
6: $d_j \leftarrow \sqrt{(R_i + r_j - x_j)^2 - y_j^2}$;
7: if $d_j < m$
8: $m \leftarrow d_j$;
9: k$\leftarrow$j;
10: endif
11: endfor
12: $S_c \leftarrow S_c \cup \{s_k\}$;
13: $i \leftarrow i + 1$;
14: $R_i \leftarrow R_{i-1} + 2r_k$;
15: if $\lambda < m$
16: $\lambda \leftarrow m$;
17: endif
18: endwhile;
19: return λ;

5. The decision problem for the uniform case

This section studies the decision problem for the uniform case, which is to determine whether the sensors can move to cover the barrier when the sensors' maximum movement is not larger than a fixed value λ. The challenging issue is how to obtain the final positions of the sensors forming a barrier coverage. Motivated by the algorithm in [3], we determine the critical sensors and their final positions in the sensor deployment scheme. The difference is that we determine the critical sensors by considering sensors' moving ranges rather than sensors' right-side extensions as in [3]. The algorithm for decision problem, short for the decision algorithm, will be used as a decision procedure by our algorithm to solve the 2-D MinMax problem for the uniform case in Section 6.

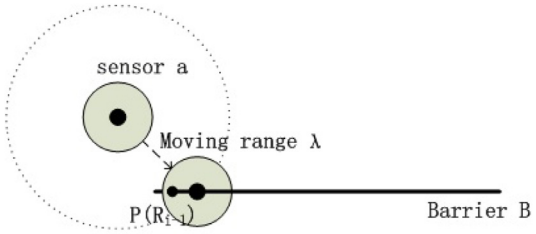
5.1. Algorithm description

The main idea of the decision algorithm is to choose a set of critical sensors to move to cover the barrier from the left to the right.

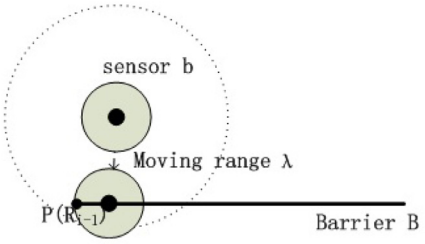
In each step of the algorithm, a critical sensor is chosen to cover the leftmost uncovered point of the barrier. The critical sensor is chosen as follows: first, check whether there is a non-critical sensor which can cover the leftmost uncovered point of the barrier and can not be in attaching position when it is located at its right moving endpoint. If yes, choose the satisfied sensor with the largest right moving endpoint as the critical sensor and put it located at its right moving endpoint; otherwise, check whether there is a non-critical sensor which can cover the leftmost uncovered point of the barrier. If yes, choose the satisfied sensor with the smallest right moving endpoint as a critical sensor and put it in attaching position; otherwise, the barrier cannot be covered by these sensors.

The algorithm works as follows:

Let the set $S_c \subseteq S$ denote the set of critical sensors, i.e. $S_c = \{s_{c(i)}\}(1 \le i \le \gamma)$, where $s_{c(i)}$ is the sensor chosen in step i to cover the leftmost uncovered point on B. In ith step, suppose the leftmost uncovered point of the barrier is $R_{i-1}(R_{i-1} < L)$, a critical sensor $s_{c(i)}$ is chosen as follows: First, scan all the non-critical sensors and put the satisfied sensors which can move to cover the point R_{i-1} and also can not be in attaching position, which implies that its right moving endpoint is smaller than $R_{i-1} + r$ into the set S_{i1}. That is $S_{i1} = \{s_i | R_{i-1} - r < x_i + d_i < R_{i-1} + r\} - S_c$. If $S_{i1} \neq \phi$, choose the sensor with the largest right moving endpoint from S_{i1} as a critical sensor $s_{c(i)}$ and put it located at its right moving endpoint, as shown in Fig. 4(a); otherwise, scan all the non-critical sensors and put the satisfied sensors which can move to cover the point R_{i-1} and also whose right moving endpoint is not smaller than $R_{i-1} + r$ into the set S_{i2}. That is $S_{i2} = \{s_i | x_i - d_i - r \le R_{i-1} \le x_i + d_i - r\} - S_c$. If $S_{i2} \neq \phi$, choose the sensor with the smallest right moving endpoint from S_{i2} as a critical sensor $s_{c(i)}$ and put it in attaching position, as shown in Fig. 4(b); otherwise, the algorithm stops and return $\lambda < \lambda^*$. If $R_{i-1} \ge L$, the algorithm stops and return $\lambda \ge \lambda^*$.



(a) If $S_{i1} \neq \phi$, sensor a is chosen as a critical sensor to cover the point $P(R_{i-1})$



(b) If $S_{i1} = \phi$ and $S_{i2} \neq \phi$, sensor b is chosen as a critical sensor to cover the point $P(R_{i-1})$

Fig. 4. Illustration of the critical sensor covering the point $p(R_{i-1})$.

The detail of the algorithm is described in Algorithm 2. Since there are n sensors in S, the algorithm runs at most n steps.

According to the decision algorithm, the following Lemma can be obtained which specifies the property of the sensors in the sensor deployment, which will be used in the next section.

Lemma 4. *Every sensor of S is either at the right moving endpoint or not. If not, then it is in attaching position.*

Proof. According to the decision algorithm, the critical sensors is either at the right moving endpoint or not. If not, then it is in attaching position. According to Algorithm 2, the sensors which are not critical sensors are located at the right moving endpoint. Thus, the Lemma is proved. □

Theorem 5. *The decision problem is solvable in O(nlogn) time.*

Proof. The decision problem can be solved by decision algorithm in Algorithm 2. Step 5 through 24 of Algorithm 2 dominate the running time of decision algorithm.

Step 6 through 11 obtain the set S_{i1} and find the sensor with the largest right moving endpoint from S_{i1} as a critical sensor. According to decision algorithm, put the sensor s_i which satisfies $R_{i-1} - r < x_i + d_i < R_{i-1} + r$ into set S_{i1}. Recall that $x_i + d_i$ denotes the right moving endpoint of sensor s_i. Actually, don't need to obtain all the sensors in the set S_{i1}. Sort the sensors increasingly by

Algorithm 2 decision algorithm.

INPUT: λ, S
OUTPUT: $\lambda < \lambda^*$ or $\lambda \geq \lambda^*$
1: Let $R_0 \leftarrow 0$, $i \leftarrow 1$, $S_c \leftarrow \phi$;
2: For each sensor $s_j \in S$
3: $d_j \leftarrow \sqrt{\lambda^2 - y_j^2}$;
4: endfor;
5: while $R_{i-1} < L$
6: $S_{i1} \leftarrow \{s_i | R_{i-1} - r < x_i + d_i < R_{i-1} + r\} - S_c$;
7: if $S_{i1} \neq \phi$
8: Select the sensor s_k with the largest $x_k + d_k + r$ from S_{i1} as $s_{c(i)}$;
9: $x'_{c(i)} \leftarrow x_{c(i)} + d_{c(i)}$;
10: $S_c \leftarrow S_c \cup \{s_{c(i)}\}$;
11: $R_i \leftarrow x'_{c(i)} + r$;
12: else
13: $S_{i2} \leftarrow \{s_i | x_i - d_i - r \leq R_{i-1} \leq x_i + d_i - r\} - S_c$;
14: if $S_{i2} \neq \phi$
15: Select the sensor s_k with the smallest $x_k + d_k + r$ from S_{i2} as $s_{c(i)}$;
16: $x'_{c(i)} \leftarrow R_{i-1} + r$;
17: $S_c \leftarrow S_c \cup \{s_{c(i)}\}$;
18: $R_i \leftarrow x'_{c(i)} + r$;
19: else
20: return $\lambda < \lambda^*$;
21: endif;
22: endif;
23: $i \leftarrow i + 1$;
24: endwhile;
25: for each sensor $s_k \in S \backslash S_c$
26: $x'_k \leftarrow x_k + d_k$;
27: endfor;
28: return $\lambda \geq \lambda^*$;

their right moving endpoint, which costs $O(n\log n)$ time. Then find the sensor with the largest right moving endpoint which satisfies $R_{i-1} - r < x_i + d_i < R_{i-1} + r$ by doing binary search on these sorted sensors, which costs $O(\log n)$ time. There are n sensors, thus step 6 through 11 perform at most n times, which cost $O(n\log n)$ time.

Step 13 through 18 obtain the set S_{i2} and find the sensor with the smallest right moving endpoint from S_{i2} as a critical sensor. According to decision algorithm, choose the sensor s_i which satisfies $x_i - d_i - r \leq R_{i-1} \leq x_i + d_i - r$ into set S_{i2}. Recall that $x_i - d_i$ denotes the left moving endpoint of sensor s_i. Sort the sensors increasingly by their left moving endpoint, which costs $O(n\log n)$ time. To obtain set S_{i2}, construct a new binary search tree using the sensors' right moving endpoints as keys. Then scan the sorted sensors until the sensor satisfies $x_i - d_i - r > R_{i-1}$ and put the sensors which satisfy $x_i - d_i - r \leq R_{i-1} \leq x_i + d_i - r$ into the binary search tree. Obviously, the sorted sensors are scanned once in all the steps, which costs $O(n)$ time in the whole algorithm. Meanwhile, delete the sensors which don't satisfy $x_i - d_i - r \leq R_{i-1} \leq x_i + d_i - r$ from the binary search tree. To delete the unsatisfied sensors, just do binary search on the search tree finding the sensors which satisfies $x_i + d - r < R_{i-1}$, which cost $O(\log n)$ time. There are n sensors, thus step 13 through 18 perform at most n times, which cost $O(n\log n)$ time.

Step 2 through 4 cost $O(n)$ time. Besides, step 25 through 27 cost $O(n)$ time.

Thus, decision algorithm cost $O(n\log n)$ time. □

5.2. Correctness

The correctness of the decision algorithm is proved in this subsection. Our analysis is similar to the one in [3]. Note that the left-aligned line segment is an line segment with the starting point at 0 and the ending point at a nonnegative number.

Lemma 6. *Suppose S_i is a subset of sensors in S. If the line segment $[0, R_i]$ is covered by the sensors of S_i by decision algorithm, then $[0, R_i]$ is the largest left-aligned line segment that can be covered by the sensors of S_i.*

Proof. Suppose there is a sensor deployment D' of the sensors in S_i which can cover the line segment $[0, x_m]$. It can be claimed that we can obtain the sensor deployment D of the critical sensors in S_i by decision algorithm which can also cover $[0, x_m]$.

Let $S_c = \{s_{c(j)}\}(1 \leq j \leq i) \subseteq S_i$ denote the critical set of S_i in the sensor deployment D. Suppose the final position of the sensor $s_{c(j)}$ in D is $x'_{c(j)}$. Suppose the final position of $s_{c(j)}$ in D' is $t_{c(j)}$. It can be proved that sensors of S_i can still cover the line segment $[0, x_m]$ if each critical sensor $s_{c(j)}$ in S_c moves from $t_{c(j)}$ to $x'_{c(j)}$.

First, it can be proved that sensor $s_{c(1)}$ can move from $t_{c(1)}$ to $x'_{c(1)}$. Suppose $t_{c(1)} \neq x'_{c(1)}$. Two cases will be considered:

Case 1: If $s_{c(1)} \in S_{11}$, then $x'_{c(1)} = x_{c(1)} + d_{c(1)}$ and $t_{c(1)} < x'_{c(1)}$. Since sensor $s_{c(1)}$ located at $x'_{c(1)}$ can cover the point 0 in D, sensor $s_{c(1)}$ can move from $t_{c(1)}$ to $x'_{c(1)}$ and sensors of S_i can still cover $[0, x_m]$.

Case 2: If $s_{c(1)} \in S_{12}$, then $S_{11} = \phi$. It implies that $s_{c(1)}$ is the sensor with the smallest right moving endpoint in S_{12}. If sensor $s_{c(1)}$ located at $t_{c(1)}$ covers the point 0 in D', then $t_{c(1)} < x'_{c(1)} = r$, thus sensor $s_{c(1)}$ can move from $t_{c(1)}$ to $x'_{c(1)}$ and sensors of S_i can still cover $[0, x_m]$. If sensor $s_{c(1)}$ located at $t_{c(1)}$ doesn't cover the point 0 in D', two cases will be considered: if $t_{c(1)} < -r$, sensor $s_{c(1)}$ can move from $t_{c(1)}$ to $x'_{c(1)}$ and sensors of S_i can still cover $[0, x_m]$; otherwise, it can be claimed that the sensor s_r which cover the point 0 and sensor $s_{c(1)}$ can switch their positions. Since $S_{11} = \phi$ and $s_{c(1)}$ is the sensor with the smallest right moving endpoint in S_{12}, the right moving endpoint of s_r is not smaller than that of $s_{c(1)}$. Then switch these two sensors by moving s_r to $t_{c(1)}$ and move $s_{c(1)}$ to $x'_{c(1)}$. Sensors of S_i can still cover $[0, x_m]$.

Thus, sensor $s_{c(1)}$ can move from $t_{c(1)}$ to $x'_{c(1)}$. Similarly, sensor $s_{c(j)}(1 < j \leq i)$ can be moved from $t_{c(j)}$ to $x'_{c(j)}$ one by one. Therefore, we can obtain the sensor deployment D of the critical sensors in S_i by decision algorithm which can also cover $[0, x_m]$.

By decision algorithm, the sensors of S_i can cover $[0, R_i]$ in D. Thus $x_m \leq R_i$ holds. Therefore, $[0, R_i]$ is the largest left-aligned line segment that can be covered by the sensors of S_i. □

Now we'll prove the correctness of decision algorithm by using Lemma 6.

Theorem 7. *If decision algorithm reports $\lambda \geq \lambda^*$, there is a feasible sensor deployment scheme on λ; If it reports $\lambda < \lambda^*$, there is no feasible sensor deployment scheme on λ.*

Proof. If decision algorithm reports $\lambda \geq \lambda^*$, the set of critical sensors can cover the barrier, which implies that there is a feasible sensor deployment scheme on λ.

If decision algorithm reports $\lambda < \lambda^*$, the sensors of S can cover $[0, R_i]$, where $R_i < L$. By Lemma 6, $[0, R_i]$ is the largest left-aligned line segment that can be covered by the sensors of S. Therefore, there is no feasible sensor deployment scheme on λ. □

6. The 2-D MinMax problem for the uniform case

This section presents a polynomial-time algorithm to solve the 2-D MinMax problem when the sensors' sensing ranges are uniform. This section first describes the basic idea of our algorithm, then presents a Theorem of necessary condition for the optimal sensor deployment and finally proposes an algorithm and an improved algorithm for the 2-D MinMax problem for the uniform case.

6.1. Basic idea of our algorithm

Mobile sensors are initially deployed randomly in the plane containing the barrier and then move to the barrier for achieving barrier coverage. The objective of the 2-D MinMax problem is to minimize the maximum movement of mobile sensors to achieve barrier coverage. It is challenging to solve the 2-D MinMax problem for the uniform case, since the sensors can move to arbitrary points of the barrier and there are infinite candidate values for the minimum maximum sensor movement. It's unknown whether the minimum maximum sensor movement can be found in a polynomial time. To overcome this difficulties, a necessary condition of the optimal sensor deployment is derived so that the search space of the optimal movement is narrowed down from infinite candidate values to $O(n^3)$ candidate values. Obviously, the minimum maximum sensor movement is the smallest feasible value among these candidate values which can achieve barrier coverage. The smallest feasible value can be found by first sorting these candidate values and then doing binary search on these sorted value using the decision algorithm as the decision procedure.

6.2. Necessary condition for optimality

In this subsection a necessary condition of the optimal sensor deployment is presented to help compute $\Theta(n^3)$ candidate values for the minimum maximum sensor movement.

Lemma 8. *Suppose $D(\lambda)$ is a feasible sensor deployment on λ by decision algorithm. If $\lambda = \lambda^*$, then one of the following conditions is true.*

(a) There exists a sensor which is at the right moving endpoint and the last sensor is at the position of $L - r$ in $D(\lambda)$.
(b) There exists two sensors which are both at the right moving endpoint and one of which with larger position is also in attaching position in $D(\lambda)$.
(c) There exists a sensor which is located at the left moving endpoint in $D(\lambda)$.

Proof. This Lemma can be proved by contradiction. Suppose none of the two conditions is true, it can be proved that $\lambda > \lambda^*$.

We choose an small positive constant ε(e.g. ε is 10^{-10}). If λ is decreased to be $\lambda - \varepsilon$, the right moving endpoints of the sensors are shifted to the left a bit in $D(\lambda - \varepsilon)$ compared to $D(\lambda)$. Since ε is a very small positive constant, the final positions of the sensors which are not at the left or right moving endpoint in $D(\lambda)$ can keep unchanged or be shifted to the left a bit in $D(\lambda - \varepsilon)$, since these sensors are still located within their moving range in $D(\lambda - \varepsilon)$.

If none of the conditions is true, two cases are studied as follows:

Case 1: None of the sensors is located at left or right moving endpoint in $D(\lambda)$. Suppose λ is decreased to be $\lambda - \varepsilon$. The final positions of these sensors can keep unchanged. It implies that $D(\lambda - \varepsilon)$ is a feasible sensor deployment, thus $\lambda > \lambda^*$.

Case 2: None of the sensors is located at the left moving endpoint in $D(\lambda)$ and there exists a sensor which is at the right moving endpoint in $D(\lambda)$. Besides, the last sensor is at the position larger than $L - r$ and there doesn't exist two sensors which are both at the right moving endpoint and one of which with larger position is also in attaching position in $D(\lambda)$. Note that if the last sensor is at the position smaller than $L - r$, $D(\lambda)$ is not a feasible sensor deployment, thus the last sensor is at the position larger than $L - r$. Below a feasible sensor deployment can be produced on $\lambda - \varepsilon$.

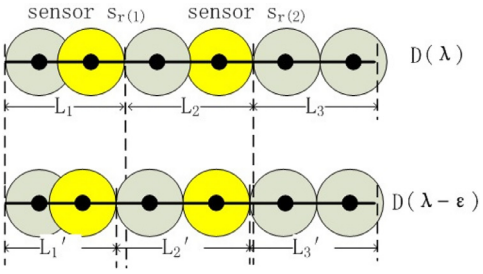


Fig. 5. Illustration of the proof of Lemma 8 (Yellow circle represented as sensors located at the right moving endpoint. Compared to $D(\lambda)$, the sensors' final positions are shifted to the left a bit in $D(\lambda - \varepsilon)$ except the first sensor).

Let $S_r = \{s_{r(1)}, s_{r(2)}, \ldots, s_{r(k)}\}$, where $s_{r(i)}$ denotes the sensor located at the right moving endpoint. The sensors of S_r are sorted by their positions in $D(\lambda)$ increasingly.

Let L_1 denote the subsegment of barrier covered by sensor $s_{r(1)}$ and the sensors preceding $s_{r(1)}$ in $D(\lambda)$. Let L_i denote the subsegment of barrier covered by sensor $s_{r(i)}$ and the sensors between $s_{r(i-1)}$ and $s_{r(i)}$ in $D(\lambda)$, where $1 < i \le k$. Let L_{k+1} denote the subsegment of barrier covered by the sensors following sensor $s_{r(k)}$ in $D(\lambda)$. Obviously, $\Sigma_{i=1}^{k+1} L_i > L$ holds because the last sensor is at the position larger than $L - r$. Fig. 5 shows an example of L_i.

Suppose λ is decreased to be $\lambda - \varepsilon$. Then the right moving endpoints of sensors in S_r are shifted to the left a bit, thus the final positions of sensors in S_r should be also shifted to the left a bit.

Let L_1' denote the sub-segment of barrier which starts from the position 0 to the right moving extension of sensor $s_{r(1)}$ in $D(\lambda - \varepsilon)$. Let L_i' denote the sub-segment of barrier which starts from the right moving extension of sensor $s_{r(i-1)}$ to that of sensor $s_{r(i)}$ in $D(\lambda - \varepsilon)$, where $1 < i \le k$. Let L_{k+1}' denote the sub-segment of barrier which starts from the right moving extension of sensor $s_{r(i)}$ to the position L. Fig. 5 shows an example of L_i'.

It'll be proved that there is a feasible sensor deployment $D(\lambda - \varepsilon)$ to cover the subsegments of barrier $\{L_i' | 1 \le i \le k+1\}$. These subsegments will be considered from back to forth by using induction method.

First, it can be claimed that L_{k+1}' can be covered by the sensors following sensor $s_{r(k)}$. When λ is decreased to be $\lambda - \varepsilon$, the final position of the sensor $s_{r(k)}$ should be shifted to the left a bit. Recall that none of the sensors is located at the left moving endpoint in $D(\lambda)$. The final positions of the sensors following sensor $s_{r(k)}$ can all be shifted to the left a bit to fill the gap left by sensor $s_{r(k)}$. Recall that the position of the last sensor is larger than $L - r$ in $D(\lambda)$. The final positions of the last sensor can be also shifted to the left a bit such that it still covers the point L. Thus the claim is proved.

It can be assumed inductively that the sensors following sensor $s_{r(i)} (1 \le i < k)$ can cover the subsegment $\Sigma_{j=i+1}^{k+1} L_j'$ when λ is decreased to be $\lambda - \varepsilon$. Depending on whether $i > 1$, two cases are considered:

If $i > 1$, it can be claimed that L_i' can be covered by sensor $s_{r(i)}$ and the sensors between $s_{r(i)}$ and $s_{r(i-1)}$.

When λ is decreased to be $\lambda - \varepsilon$, the final positions of the sensor $s_{r(i-1)}$ and $s_{r(i)}$ should be shifted to the left a bit. Let $d_{r(i-1)}(\varepsilon)$ and $d_{r(i)}(\varepsilon)$ be the movement of the final positions of sensor $s_{r(i-1)}$ and sensor $s_{r(i)}$ when λ is decreased by ε. Two cases are considered.

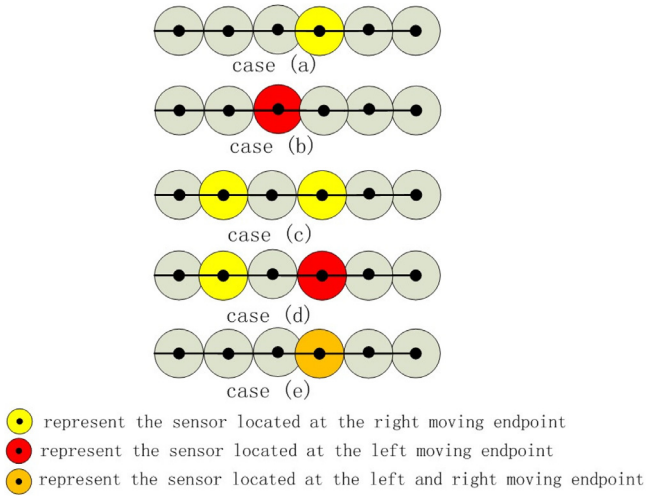


Fig. 6. The illustration of Theorem 9.

Case 1: $d_{r(i-1)}(\varepsilon) \le d_{r(i)}(\varepsilon)$. Obviously, $L'_i \le L_i$. It implies the sensing ranges of sensors may still cover L'_i. Recall that none of these sensors is located at the left moving endpoint in $D(\lambda)$. The final positions of the sensors between sensor $s_{r(i-1)}$ and sensor $s_{r(i)}$ can all be shifted to the left a bit to fill the gap left by sensor $s_{r(i-1)}$. Thus L'_i can be covered by sensor $s_{r(i)}$ and the sensors between $s_{r(i)}$ and $s_{r(i-1)}$.

Case 2: $d_{r(i-1)}(\varepsilon) > d_{r(i)}(\varepsilon)$. Obviously, $L'_i > L_i$. It can be claimed that L'_i can be covered by sensor $s_{r(i)}$ and the sensors between $s_{r(i)}$ and $s_{r(i-1)}$. Let L_m denote double the sum of the sensing ranges of these sensors. Recall that $s_{r(i)}$ is not in attaching position. Thus, $L_m > L_i$ holds. Since ε is very small, $L'_i \le L_m$ can hold. The final positions of the sensors between sensor $s_{r(i-1)}$ and sensor $s_{r(i)}$ can all be shifted to the left a bit to fill the gap left by sensor $s_{r(i-1)}$, thus L'_i can be covered by sensor $s_{r(i)}$ and the sensors between $s_{r(i)}$ and $s_{r(i-1)}$.

If $i = 1$, it can be claimed that L'_1 can be covered by sensor $s_{r(1)}$ and the sensors preceding $s_{r(1)}$. When λ is decreased to be $\lambda - \varepsilon$, the final position of the sensor $s_{r(1)}$ must be shifted to the left a bit. The final positions of the sensors following sensor $s_{r(k)}$ can keep unchanged. Thus L'_1 can be covered by sensor $s_{r(1)}$ and the sensors preceding $s_{r(1)}$.

Therefore, there is a feasible sensor deployment $D(\lambda - \varepsilon)$ to cover $\Sigma_{j=1}^{k+1} L'_j$, which is the barrier. Then $\lambda > \lambda^*$ holds. The Lemma is proved by contradiction. □

Below, Lemma 8 can be rewritten to obtain a Theorem as follows, which is illustrated in Fig. 6.

Theorem 9. *Suppose $D(\lambda)$ is a feasible sensor deployment on λ by the decision algorithm. If $\lambda = \lambda^*$, then one of the following cases is true.*

- *(a) There is a sensor located at the right moving endpoint in $D(\lambda)$ such that the sensors following it is in attaching position and the last sensor is at the position of $L - r$.*
- *(b) There exists a sensor s_i located at the left moving endpoint in $D(\lambda)$ such that all the sensors preceding sensor s_i are in attaching position.*
- *(c) There exist two sensor s_i and s_j such that both of them are at the right moving endpoint in $D(\lambda)$ and the sensors between sensor s_i and s_j, including s_j, are all in attaching position.*
- *(d) There exists a sensor s_j located at the left moving endpoint in $D(\lambda)$ such that there is a sensor s_i preceding it which is at the right moving endpoint in $D(\lambda)$ and the sensors between sensor s_i and s_j are all in attaching position.*
- *(e) There exists a sensor s_i located at the left moving endpoint and also the right moving endpoint in $D(\lambda)$.*

Proof. By Lemma 4, every sensor is either in attaching position or not. If not, then it is at the right moving endpoint.

By Lemma 8, there exists a sensor which is at the right moving endpoint and the last sensor is at the position of $L - r$ in $D(\lambda)$. It implies that there is a sensor located at the right moving endpoint in $D(\lambda)$ such that the sensors following it is in attaching position and the last sensor is at the position of $L - r$.

By Lemma 8, there exists two sensors which are both at the right moving endpoint and one of which with larger position is also in attaching position in $D(\lambda)$. It implies that there exist two sensor s_i and s_j such that both of them are at the right moving endpoint in $D(\lambda)$ and the sensors between sensor s_i and s_j, including s_j, are all in attaching position.

By Lemma 8, there exists a sensor which is located at the left moving endpoint in $D(\lambda)$. Three cases will be considered.

Case 1: If none of the sensors is located at the right moving endpoint, it implies that there exists a sensor s_i located at the left moving endpoint in $D(\lambda)$ such that all the sensors preceding sensor s_i are in attaching position.

Case 2: If there is a sensor s_i preceding it which is at the right moving endpoint, it implies that there exists a sensor s_j located at the left moving endpoint in $D(\lambda)$ such that there is a sensor s_i preceding it which is at the right moving endpoint in $D(\lambda)$ and the sensors between sensor s_i and s_j are all in attaching position.

Case 3: There exists a sensor s_i located at the left moving endpoint and also the right moving endpoint in $D(\lambda)$.

Thus, the Theorem is proved. □

6.3. computing λ^*

According to Theorem 9, the candidate values for λ^* are computed in this subsection, where λ^* is the minimum maximum sensor movement. Corresponding to the five cases of Theorem 9, the candidate values for λ^* are computed as follows:

(1) for every sensor $s_i (1 \le i \le n)$ and every $k \in [0, n-1]$, compute λ as $\lambda_1(i, k)$ which satisfies that $x_i + \sqrt{\lambda^2 - y_i^2} = L - (k \times 2r + r)$, where the position of s_i's right moving endpoint is $x_i + \sqrt{\lambda^2 - y_i^2}$;

(2) For each sensor s_i $(1 \le i \le n)$, each $k \in [0, n-1]$, compute λ as $\lambda_2(i, k)$ which satisfies that $x_i - \sqrt{\lambda^2 - y_i^2} = k \times 2r + r$, where the position of s_i's left moving endpoint is $x_i - \sqrt{\lambda^2 - y_i^2}$;

(3) For each sensor s_i $(1 \le i \le n)$, each sensor s_j $(1 \le j \ne i \le n)$ and each $k \in [0, n-2]$, compute λ as $\lambda_3(i, j, k)$ which satisfies that $x_i + \sqrt{\lambda^2 - y_i^2} + k \times 2r + 2r = x_j + \sqrt{\lambda^2 - y_j^2}$.

(4) For each sensor s_i $(1 \le i \le n)$, each sensor s_j $(1 \le j \ne i \le n)$ and each $k \in [0, n-2]$, compute λ as $\lambda_4(i, j, k)$ which satisfies that $x_i + \sqrt{\lambda^2 - y_i^2} + k \times 2r + 2r = x_j - \sqrt{\lambda^2 - y_j^2}$.

(5) For each sensor s_i $(1 \le i \le n)$, $\lambda_5(i) = y_i$.

Let Λ denote the set of $\lambda_1(i, k)$, $\lambda_2(i, k)$, $\lambda_3(i, j, k)$, $\lambda_4(i, j, k)$ and $\lambda_5(i)$. Clearly, $|\Lambda| = O(n^3)$. By Theorem 9, $\lambda^* \in \Lambda$ holds. Obviously, λ^* is the minimum feasible value among all the values in set Λ which can produce a feasible sensor deployment scheme. Recall that the decision algorithm can be adopted to determine whether there is a feasible sensor deployment on the value in set Λ. To compute λ^* is to first sort all the values of Λ increasingly and then find the minimum feasible value by doing binary search on them and using decision algorithm as the decision procedure. The algorithm runs in $O(n^3 \log n)$ time.

6.4. An improved algorithm

To reduce the runtime of the algorithm, the technique called binary search on sorted arrays is adopted from the work [4]. The runtime of the algorithm can be reduced to $O(n^2 \log n)$ time.

First introduce the technique of binary search in sorted arrays [4].

Lemma 10. *[4] Given M arrays, each array $Y_i (1 \le i \le M)$ containing $O(N)$ sorted elements, it takes $O(M+T)log(NM)$ time to find the element δ in $Y = \cup_{i=1}^{M} Y_i$, where $O(T)$ is the time consumed by reporting $a \le \delta$ or $a > \delta$ (a is a constant).*

Below we'll order the values of the set Λ into $O(n^2)$ sorted lists.

Lemma 11. *The elements of set Λ can be ordered into $O(n^2)$ sorted lists such that each list has $O(n)$ elements in $O(n^2)$ time.*

Proof. Recall that set Λ is the set of $\lambda_1(i, k)$, $\lambda_2(i, k)$, $\lambda_3(i, j, k)$, $\lambda_4(i, j, k)$ and $\lambda_5(i)$.

The set of $\lambda_1(i, k)$ has $O(n^2)$ elements. By simple calculation, $\lambda_1(i, k)$ is a decreasing function of k when i is fixed. That is $\lambda_1(i, k_1) \ge \lambda_1(i, k_2)$ when $k_1 \le k_2$. Thus for each i $(1 \le i \le n)$, there is a sorted list of at most $n-1$ elements $\lambda_1(i, 0)$, $\lambda_1(i, 1)$,...,$\lambda_1(i, n-2)$. Thus, the elements in the set of $\lambda_1(i, k)$ can be sorted into $O(n)$ sorted lists denoted by arrays $C_i (1 \le i \le n)$, and each list has $O(n)$ values. For example, $C_1 = \{\lambda_1(1, 0), \lambda_1(1, 1) \ldots, \lambda_1(1, n-1)\}$. Note that the elements of each array are not computed explicitly but computed when needed. It costs $O(n)$ time.

Similarly, the set of $\lambda_2(i, k)$ can be sorted in the same way. The elements in the set of $\lambda_2(i, k)$ can be sorted into $O(n)$ sorted lists denoted by arrays $D_i (1 \le i \le n)$, and each list has $O(n)$ values. It costs $O(n)$ time.

By simple calculation, $\lambda_3(i, j, k)$ is a decreasing function of k, when both i and j are fixed. That is $\lambda_3(i, j, k_1) \ge \lambda_3(i, j, k_2)$ when $k_1 \le k_2$, thus for each pair of i and $j (1 \le i \ne j \le n)$, there is a sorted list of at most $n-1$ elements $\lambda_3(i, j, 0)$, $\lambda_3(i, j, 1)$,...,$\lambda_3(i, j, n-2)$. Therefore, the elements in the set of $\lambda_3(i, j, k)$ is sorted into $O(n^2)$ sorted lists, denoted by arrays $E_{ij} (1 \le i \ne j \le n)$, such that each list has $O(n)$ elements. It costs $O(n^2)$ time.

Similarly, the set of $\lambda_4(i, j, k)$ is sorted in the same way. The elements in the set of $\lambda_4(i, j, k)$ can be sorted into $O(n^2)$ sorted lists, denoted by arrays $F_{ij} (1 \le i \ne j \le n)$, and each list has $O(n)$ values. It costs $O(n^2)$ time.

The set of $\lambda_5(i)$ has $O(n)$ elements, which can be sorted into Array G, which costs $O(n \log n)$ time.

Therefore, the elements of set Λ can be ordered into $O(n^2)$ sorted lists such that each list has $O(n)$ elements in $O(n^2)$ time. □

By lemma 11, we have the following results:

1) The elements in the set of $\lambda_1(i, k)$ can be ordered into $O(n)$ sorted arrays $C_i (1 \le i \le n)$, and each list has $O(n)$ values. Besides, the kth element λ of the array C_i can be explicitly computed, which satisfies $x_i + \sqrt{\lambda^2 - y_i^2} = L - (k \times 2r + r)$.

2) The elements in the set of $\lambda_2(i, k)$ can be ordered into $O(n)$ sorted arrays $D_i (1 \le i \le n)$, and each list has $O(n)$ values. Besides, the kth element λ of the array D_i can be explicitly computed, which satisfies $x_i - \sqrt{\lambda^2 - y_i^2} = k \times 2r + r$.

3) The elements in the set of $\lambda_3(i, j, k)$ can be ordered into $O(n^2)$ sorted arrays $E_{ij} (1 \le i \ne j \le n)$ and each list includes $O(n)$ elements. Besides, the kth element λ of the array E_{ij} can be explicitly computed, which satisfies n $x_i + \sqrt{\lambda^2 - y_i^2} + k \times 2r + 2r = x_j + \sqrt{\lambda^2 - y_j^2}$.

4) The elements in the set of $\lambda_4(i, j, k)$ can be ordered into $O(n^2)$ sorted arrays $F_{ij} (1 \le i \ne j \le n)$ and each list has $O(n)$ values. Besides, the kth element λ of the array F_{ij} can be explicitly computed, which satisfies $x_i + \sqrt{\lambda^2 - y_i^2} + k \times 2r + 2r = x_j - \sqrt{\lambda^2 - y_j^2}$.

5) The elements in the set of $\lambda_5(i)$ can be ordered into array G, and the kth element λ of the array G is y_k.

Then the smallest feasible value λ_c can be found in arrays $\cup_{i=1}^{n} C_i$ using binary search on sorted arrays and decision algorithm. Similarly, the smallest feasible value λ_d can be found in arrays $\cup_{i=1}^{n} D_i$, or λ_e in arrays $\cup_{i,j=1}^{n} E_{ij}$, or λ_f in arrays $\cup_{i,j=1}^{n} F_{ij}$, or λ_g array G. Then $\lambda^* = min\{\lambda_c, \lambda_d, \lambda_e, \lambda_f, \lambda_g\}$.

Theorem 12. *The 2-D MinMax problem when the sensors' sensing range are uniform can be solved in $O(n^2 logn)$ time.*

Proof. According to Lemma 11, the elements of the set Λ are sorted into $O(n^2)$ arrays and each array includes $O(n)$ sorted elements. Recall that the decision algorithm costs $O(nlogn)$ time. By Lemma 10, it takes $O(n^2 + nlogn)\log(n^* n^2)$ time to find the smallest feasible element in Λ. Thus, the 2-D MinMax problem for the uniform case can be solved in $O(n^2 logn)$ time. □

7. Performance evaluation

This section evaluates the performance of the proposed algorithms for 2-D MinMax problem by conducting a set of simulation experiments using Matlab. We first investigate the performance of the algorithms for the general case that sensors' sensing ranges are arbitrary, and then study the performance of the algorithm for the uniform (sensing range) case.

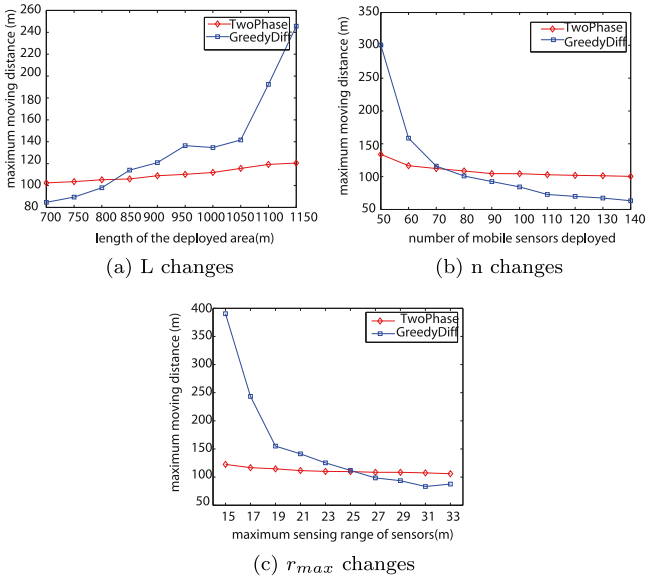
We use robomotes as mobile sensors. Each robomote is smaller than $0.000047m^3$, costs less than 150 dollars and is equipped with two AA batteries with an initial amount of energy 24,172 Joules [8]. IEEE 802.11b is chosen as the medium access control protocol and a shortest path routing is adopted [26]. This robomote is with the differential drive and moves two coaxial wheels with two independent electric motors. It consumes 27.96 Joule to move one meter and 0.1 Joule per second to sense and communicate [21,28]. The final positions of all robomotes are computed by a sink node, which is rechargeable and with strong computing and storage capabilities. The robomotes carry little computing power and storage, which can be ignored.

All sensors are deployed randomly and uniformly in a rectangular area of length $L = 1000m$ and width $w = 100m$. The barrier B is located in the same area with the starting point [0, 0] and the ending point [1000, 0]. We only consider that case that the sensors are sufficient to cover the barrier. Note that each data point of our experiment results is an average value of the data collected by running the experiments 100 times. The metric studied is the maximum moving distance for mobile sensors to achieve barrier coverage, the average moving distance for mobile sensors to achieve barrier coverage and the running time of the algorithm.

7.1. Evaluation for the general case

We solve the 2-D MinMax barrier coverage problem for the general case of arbitrary sensor sensing ranges. The performance of the two phase algorithm called the TwoPhase algorithm is compared with the GreedyDiff algorithm in three scenarios by measuring the metric of the maximum moving distance, the average moving distance and the running time. Evaluation of these performance metrics is conducted on different parameters, such as length of the deployed area(L), number of mobile sensors deployed(n) and maximum sensing range of sensors(r_{max}). The default setting of the parameters are L=1000, n=75 and r_{max}=25.

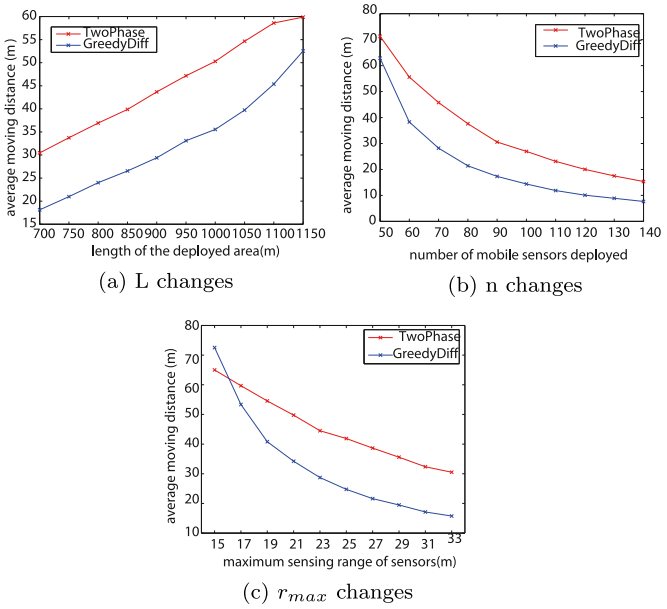
Fig. 7 shows the effects of different parameters on the maximum moving distance for the general case. The red curve represents the maximum moving distance obtained by the TwoPhase



(a) L changes (b) n changes

(c) r_{max} changes

Fig. 7. Performance evaluation of algorithms for the general case on the maximum moving distance.



(a) L changes (b) n changes

(c) r_{max} changes

Fig. 8. Performance evaluation of algorithms for the general case on the average moving distance.

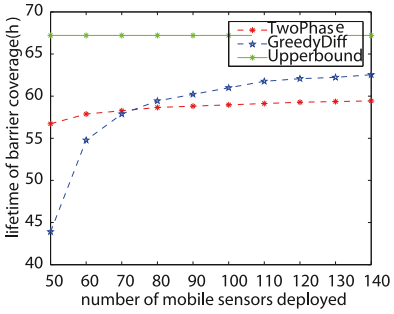


Fig. 9. Performance evaluation of algorithms for the general case on the network lifetime.

algorithm while the blue curve represents the maximum moving distance obtained by the GreedyDiff algorithm. It can observed that no matter how the parameters change, the maximum sensor movement by using the TwoPhase algorithm increases slowly while that by using the GreedyDiff algorithm increases fast. It can be seen that the TwoPhase algorithm is more scalable.

As shown in Fig. 7(a), the maximum moving distance by the two algorithms both increases as the length of the deployed area increases. Since the number of mobile sensor deployed is fixed, the density of the sensors decreases as the length of the deployed area increases. Thus, the sensors need to move a larger distance to cover the barrier. When the length of barrier is smaller than 800, the result by the GreedyDiff algorithm is smaller than that by the TwoPhase algorithm. When the length of barrier is larger than 800, the result by the GreedyDiff algorithm is larger than that by the TwoPhase algorithm. TwoPhase algorithm is more suitable if the deployed area is long.

As shown in Fig. 7(b), the maximum moving distance by the two algorithms decreases as number of mobile sensor deployed increases. That's because nearer sensors can be chosen to cover the barrier. Thus, deploying more mobile sensors can reduce the maximum sensor movement. When the number of mobile sensors is less than 70, the result by the TwoPhase algorithm is smaller than that by the GreedyDiff algorithm. When the number of mobile sensors is larger than 70, the GreedyDiff algorithm outperforms the TwoPhase algorithm. When the number of mobile sensors is big, the GreedyDiff algorithm is more suitable.

As shown in Fig. 7(c), the maximum moving distance by the two algorithms decreases as the maximum sensing range of sensors increases. That's because sensors with larger sensing range can cover larger subsegment of the barrier, which implies that sensors can move a smaller distance to cover the barrier. When the maximum sensing range of sensors is less than 25, the result by the TwoPhase algorithm is smaller than that by the GreedyDiff algorithm. When the maximum sensing range of sensors is larger than 25, the GreedyDiff algorithm outperforms the TwoPhase algorithm. The TwoPhase algorithm is more suitable when the maximum sensing range of sensors is small.

Fig. 8 shows the effects of different parameters on the average moving distance for the general case. The red curve represents the average moving distance obtained by the TwoPhase algorithm while the blue curve represents the average moving distance obtained by the GreedyDiff algorithm. As shown in Fig. 8(a), the average moving distance by the two algorithms increase as the length of the deployed area increases. The result by the TwoPhase algorithm is larger than that by the GreedyDiff algorithm, which is because the GreedyDiff algorithm uses less sensors than TwoPhase algorithm, which implies that the total sensor movement by the GreedyDiff algorithm is less than that by TwoPhase algorithm. As shown in Fig. 8(b), the average moving distance by the two algorithms decreases as the maximum sensing range of sensors increases. The result by the TwoPhase algorithm is larger than that by the GreedyDiff algorithm. As shown in Fig. 8(c), the average moving distance by the two algorithms decreases as the maximum sensing range of sensors increases. In most of the cases, the result by the TwoPhase algorithm is larger than that by the GreedyDiff algorithm. Therefore, the GreedyDiff algorithm outperforms the TwoPhase algorithm for the metric of the average moving distance.

Fig. 9 shows the network lifetime for barrier coverage for the negeral case. The green line denotes the upperbound of network lifetime, which is the lifetime of a working sensor without moving. The upperbound of network lifetime is the initial energy of batteries divided by the energy for sensing and communicating, which is 67.2 hours. From this figure, it can be seen that the network lifetime by the TwoPhase algorithm, denoted by the red curve, is almost 57 hours, which is almost 84% of the upperbound of the

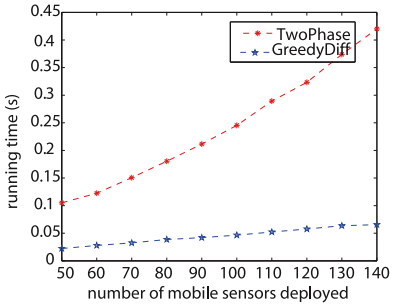


Fig. 10. Performance evaluation of algorithms for the general case on the running time.

network lifetime. The network lifetime by the GreedyDiff algorithm, denoted by blue curve, increases from 44 hours to 63 hours, when the number of sensors increases from 50 to 140, which implies that the moving distance of sensors has an important effect on the network lifetime. These results shows that the TwoPhase algorithm can result a longer network lifetime than the GreedyDiff algorithm when the number of sensors is small, while the latter would be better when the number of sensors is large.

Fig. 10 shows the comparison of the running time of these two algorithms for the general case. The running time of GreedyDiff algorithm is smaller than that of the TwoPhase algorithm. Thus, GreedyDiff algorithm is more suitable for practical applications in this aspect.

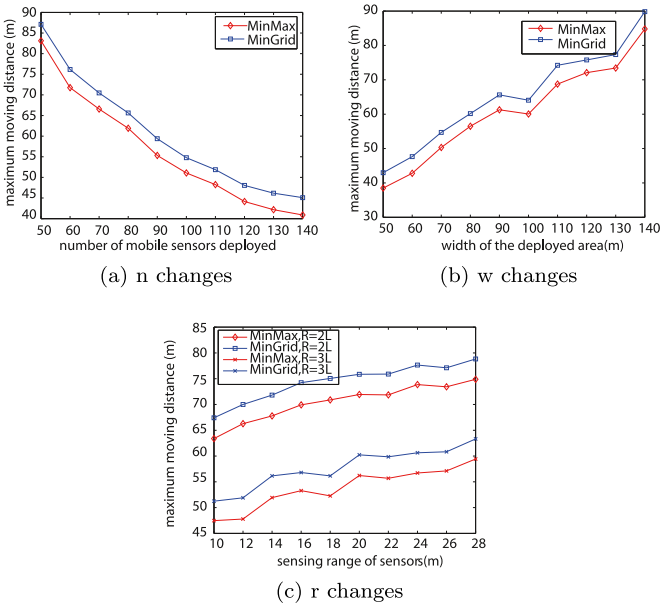
7.2. Evaluation for the uniform case

We solve the 2-D MinMax barrier coverage problem for the case that all sensors have the same sensing range and propose an algorithm called the MinMax algorithm. We choose a grid_based algorithm adopted from [17] for comparison, which is called MinGrid. In this algorithm, all the sensors are only allowed to move to grid points of coordinates $(2i+1)\times r$ on barrier B, where $0 \le i \le l/2r-1$. The MinGrid algorithm tries to match a mobile node to each grid point and minimizes the maximum moving distance among all sensors. The maximum flow algorithm used by the MinGrid algorithm as the decision procedure runs in a lot of time. Thus, we use the decision algorithm in our paper instead as the decision procedure to reduce the time complexity.

We compare the performance of the MinMax algorithm with that of the MinGrid algorithm in three scenarios by measuring the metric of the maximum moving distance, the average moving distance and the running time. Evaluation of these performance metrics is conducted on different parameters, such as number of mobile sensors deployed(n), width of the deployed area(w) and sensing range of sensors(r). The default setting of the parameters are L=1000, w=100, n=100 and r=15.

Fig. 11 shows the effects of different parameters on the maximum moving distance for the uniform case. The red curve represents the maximum moving distance obtained by the MinMax algorithm while the blue curve represents the maximum moving distance obtained by the MinGrid algorithm. It can observed that in all the cases, the maximum moving distance by the MinMax algorithm is smaller than the MinGrid algorithm, which validates the optimality of the MinMax algorithm. The reason is that the MinMax algorithm may choose a closer final position than the MinGrid on the barrier for each sensor.

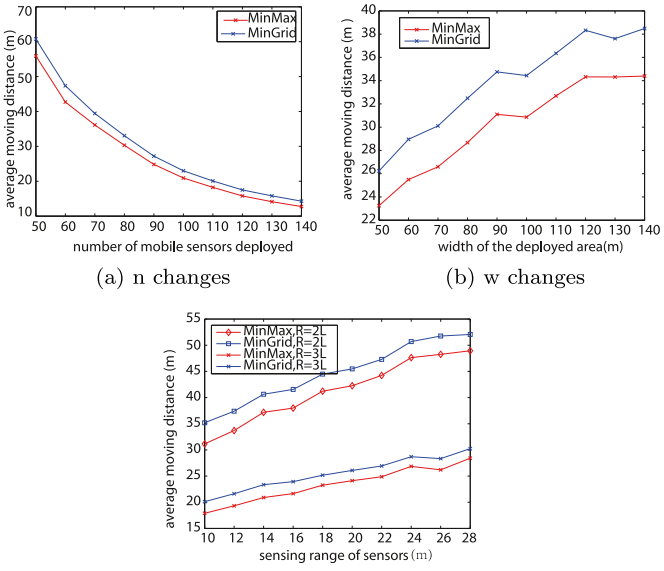
As shown in Fig. 11(a), the maximum moving distance decreases as the number of sensors increases. That's because when more sensors are deployed, there will be more sensors closer to choose from to cover the barrier. As shown in Fig. 11(b), the maximum moving distance increases as the width of the deployed area



(a) n changes (b) w changes

(c) r changes

Fig. 11. performance evaluation of algorithms for the uniform case on the maximum moving distance.



(a) n changes (b) w changes

(c) r changes

Fig. 12. Performance evaluation of algorithms for the uniform case on the average moving distance.

increases. That's because the sensors are randomly distributed in a wider area and they have to move a larger distance to cover the barrier line. As shown in Fig. 11(c), the maximum movement for the case of $R = 3L$ is almost 80 percent of that for the case of $R = 2L$. The maximum moving distance increases as the sensing range of sensors increases. That's because as the sensing range of sensors increases, the number of sensors decreases. Thus the sensors have to travel a larger distance. This motivates us to deploy small sensing range of sensors to prolong the lifetime of wireless sensor network.

Fig. 12 shows the effects of different parameters on the average moving distance for the uniform case. The red curve represents the average moving distance obtained by the MinMax algorithm while the blue curve represents the average moving distance obtained by the MinGrid algorithm. In all the cases, the average moving dis-

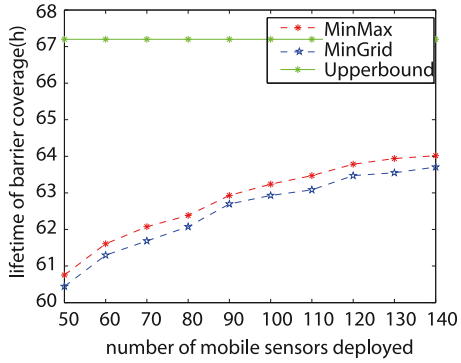


Fig. 13. Performance evaluation of algorithms for the uniform case on the network lifetime.

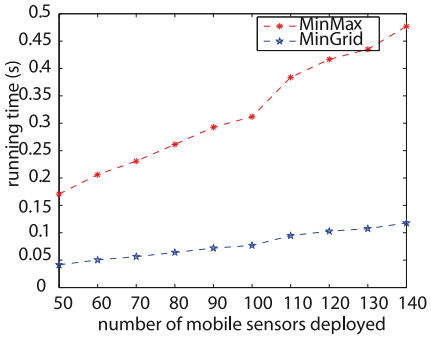


Fig. 14. Performance evaluation of algorithms for the uniform case on the running time.

tance by the MinMax algorithm is smaller than the MinGrid algorithm. As shown in Fig. 12(a), the average moving distance decreases as the number of sensors increases. As shown in Fig. 12(b), the average moving distance increases as the width of the deployed area increases. As shown in Fig. 12(c), the average moving distance increases as the sensing range of sensors increases.

Fig. 13 shows the network lifetime for barrier coverage for the uniform case. The upperbound of network lifetime is 67.2 hours, which is denoted by the green line. The network lifetime by MinMax algorithm, denoted by the red curve, is greater than that by MinGrid algorithm denoted by the blue curve. As the number of sensors increases, these curves show that the network lifetimes by these two algorithms both increase. So in order to prolong the network lifetime, more sensors should be deployed so that the maximum sensor movement can be reduced.

Fig. 14 shows the comparison of the running time of these two algorithms for the uniform case. Although the running time of MinGrid algorithm is smaller than that of the MinMax algorithm, our MinMax algorithm can find the minimum maximum sensor movement.

8. Conclusions

This paper studied the 2-D MinMax barrier coverage problem under the assumption that the barrier is a line segment in a two-dimensional plane and the sensors are initially located in this plane. Two algorithms are proposed for the general case of arbitrary sensor sensing ranges and then an $O(n^2 \log n)$ time algorithm is presented for the uniform sening-range case. It would be interesting to see whether our techniques can be extended to solve the 2-D MinMax problem for the case that sensor ranges are within a set of fixed values. Besides, this 2-D MinMax problem requires that the final positions of sensors should be on the barrier. It is challenging to design an algorithm for achieving barrier coverage and also minimizing the maximum sensor movement without the assumption that the final positions of the sensors must be on the barrier.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by National Key Research & Development Program of China Project #2017YFB0203201, and Australian Research Council Discovery Project DP150104871, and National Natural Science Foundation of China (Grant No. 61702198). The corresponding author is Hong Shen.

References

[1] B. Bhattacharya, M. Burmester, Y. Hu, E. Kranakis, Q. Shi, A. Wiese, Optimal movement of mobile sensors for barrier coverage of a planar region, Theor. Comput. Sci. 410 (52) (2009) 5515–5528.
[2] A. Chen, S. Kumar, T.H. Lai, Local barrier coverage in wireless sensor networks, Mobile Comput. IEEE Trans. 9 (4) (2010) 491–504.
[3] D.Z. Chen, Y. Gu, J. Li, H. Wang, Algorithms on minimizing the maximum sensor movement for barrier coverage of a linear domain, Discrete Comput. Geom. 50 (2) (2013) 374–408.
[4] D.Z. Chen, C. Wang, H. Wang, Representing a functional curve by curves with fewer peaks, Discrete Comput. Geom. 46 (2) (2011) 334–360.
[5] A. Cherry, J. Gudmundsson, J. Mestre, Barrier coverage with uniform radii in 2d, in: International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, Springer, 2017, pp. 57–69.
[6] J. Czyzowicz, E. Kranakis, D. Krizanc, I. Lambadaris, L. Narayanan, J. Opatrny, L. Stacho, J. Urrutia, M. Yazdani, On minimizing the maximum sensor movement for barrier coverage of a line segment, in: Ad-Hoc, Mobile and Wireless Networks, Springer, 2009, pp. 194–212.
[7] J. Czyzowicz, E. Kranakis, D. Krizanc, I. Lambadaris, L. Narayanan, J. Opatrny, L. Stacho, J. Urrutia, M. Yazdani, On minimizing the sum of sensor movements for barrier coverage of a line segment, in: Ad-Hoc, Mobile and Wireless Networks, Springer, 2010, pp. 29–42.
[8] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, G.S. Sukhatme, Robomote: enabling mobility in sensor networks, in: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, in: IPSN '05, IEEE Press, Piscataway, NJ, USA, 2005.
[9] X. Deng, B. Wang, C. Wang, H. Xu, W. Liu, Mending barrier gaps via mobile sensor nodes with adjustable sensing ranges, in: Wireless Communications and Networking Conference (WCNC), 2013 IEEE, IEEE, 2013, pp. 1493–1497.
[10] S. Dobrev, S. Durocher, M. Eftekhari, K. Georgiou, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, S. Shende, J. Urrutia, Complexity of barrier coverage with relocatable sensors in the plane, in: Algorithms and Complexity, Springer, 2013, pp. 170–182.
[11] S. He, J. Chen, X. Li, X.S. Shen, Y. Sun, Mobility and intruder prior information improving the barrier coverage of sparse sensor networks, IEEE Trans. Mob. Comput. 13 (6) (2014) 1268–1282.
[12] J. Li, J. Chen, T.H. Lai, Energy-efficient intrusion detection with a barrier of probabilistic sensors, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 118–126.
[13] S. Li, H. Shen, Minimizing the maximum sensor movement for barrier coverage in the plane, in: Computer Communications (INFOCOM), 2015 IEEE Conference on, IEEE, 2015, pp. 244–252.
[14] H. Ma, M. Yang, D. Li, Y. Hong, W. Chen, Minimum camera barrier coverage in wireless camera sensor networks, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 217–225.
[15] M. Mehrandish, L. Narayanan, J. Opatrny, Minimizing the number of sensors moved on line barriers, in: Wireless Communications and Networking Conference (WCNC), 2011 IEEE, IEEE, 2011, pp. 653–658.
[16] A. Saipulla, J.-H. Cui, X. Fu, B. Liu, J. Wang, Barrier coverage: foundations and sesign, in: The Art of Wireless Sensor Networks, Springer, 2014, pp. 59–115.
[17] A. Saipulla, B. Liu, G. Xing, X. Fu, J. Wang, Barrier coverage with sensors of limited mobility, in: Proceedings of the 11th ACM international symposium on Mobile ad hoc networking and computing, ACM, 2010, pp. 201–210.
[18] A. Saipulla, C. Westphal, B. Liu, J. Wang, Barrier coverage of line-based deployed wireless sensor networks, in: INFOCOM 2009, IEEE, IEEE, 2009, pp. 127–135.
[19] X. Tan, G. Wu, New algorithms for barrier coverage with Mobile sensors, in: Frontiers in Algorithmics, Springer, 2010, pp. 327–338.
[20] D. Tao, T.-Y. Wu, A survey on barrier coverage problem in directional sensor networks, IEEE Sens. J. 15 (2) (2015) 876–885.

[21] G. Wang, G. Cao, T. La Porta, W. Zhang, Sensor relocation in mobile sensor networks, in: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., 4, IEEE, 2005, pp. 2302–2312.
[22] Y. Wang, G. Cao, Barrier coverage in camera sensor networks, in: Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2011, p. 12.
[23] Z. Wang, J. Liao, Q. Cao, H. Qi, Z. Wang, Barrier coverage in hybrid directional sensor networks, in: Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on, IEEE, 2013, pp. 222–230.
[24] B.M. Yamauchi, Packbot: a versatile platform for military robotics, 2004, doi:10.1117/12.538328.
[25] G. Yang, D. Qiao, Multi-round sensor deployment for guaranteed barrier coverage, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.
[26] Y. Yoo, D. Agrawal, Mobile sensor relocation to prolong the lifetime of wireless sensor networks, in: VTC Spring 2008-IEEE Vehicular Technology Conference, IEEE, 2008, pp. 193–197.
[27] X. Zhang, M.L. Wymore, D. Qiao, Optimized barrier location for barrier coverage in mobile sensor networks, in: Wireless Communications and Networking Conference (WCNC), 2015 IEEE, IEEE, 2015, pp. 1554–1559.
[28] D. Zorbas, T. Razafindralambo, Prolonging network lifetime under probabilistic target coverage in wireless mobile sensor networks, Comput. Commun. 36 (9) (2013) 1039–1053.

Shuangjuan Li received the BS degree in Computer School and the MS degree in State Key Laboratory of Software Engineering from Wuhan University in China in 2005 and 2007, respectively. She was a engineer in No.7 Research Institute of China Electronics Technology Group Corporation in Guangzhou from 2007 to 2013 and worked on the design of protocols for the wireless network system. She received the PhD degree in School of Data and Computer Science from Sun Yat-sen University in China. Now she is a lecturer in South China Agricultural University. Her research interests include optimization algorithms, wireless sensor networks.

Hong Shen is s currently a specially-appointed endowed professor at Sun Yat-sen University, China, and a tenured Professor (Chair) of Computer Science at University of Adelaide, Australia. He received the B.Eng. degree from Beijing University of Science and Technology, M.Eng. degree from University of Science and Technology of China, Ph.Lic. and Ph.D. degrees from Abo Akademi University, Finland, all in Computer Science. He was Professor and Chair of the Computer Networks Laboratory in Japan Advanced Institute of Science and Technology (JAIST) during 2001–2006, and Professor (Chair) of Compute Science at Griffith University, Australia, where he taught 9 years since 1992. With main research interests in parallel and distributed computing, algorithms, data mining, privacy preserving computing, high performance networks and multimedia systems, he has published more than 300 papers including over 100 papers in international journals such as a variety of IEEE and ACM transactions. Prof. Shen received many honours and awards.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 18, 2019, accepted October 8, 2019, date of publication October 28, 2019, date of current version November 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2949025

Optimizing the Sensor Movement for Barrier Coverage in a Sink-Based Deployed Mobile Sensor Network

SHUANGJUAN LI[1], HONG SHEN[2,3], QIONG HUANG[1], AND LONGKUN GUO[4,5]

[1]College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[2]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China
[3]School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia
[4]School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China
[5]Shandong Computer Science Center (National Supercomputer Center in Jinan), Jinan 250101, China

Corresponding author: Hong Shen (hongsh01@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702198, Grant 61402184 and Grant 61872152, in part by the National Key Research and Development Program of China Project under Grant 2017YFB0203201, in part by the Australian Research Council Discovery Project under Grant DP150104871, and in part by Guangdong Program for Special Support of Topnotch Young Professionals under Grant 2015TQ01X796.

ABSTRACT Barrier coverage is an important coverage model for intrusion detection. Clearly energy consumption of sensors is a critical issue to the design of a sensor deployment scheme. In mobile sensor network, it costs the sensors much energy to move. In this paper, we study how to optimize the sensor movement while scheduling the mobile sensors to achieve barrier coverage. Given a line barrier and n sink stations that can supply a required number of mobile sensors, we study how to find the mobile sensors' final positions on the line barrier so that the barrier is covered and the total sensor movement is minimized. We first propose a fast algorithm for determining the nearest sink for the given point on the barrier. We then propose a greedy algorithm and an optimal polynomial-time algorithm for calculating the optimal sensor movement. To obtain an optimal algorithm, we first introduce a notion of the virtual-cluster which represents a subset of sensors covering a specified line segment of the barrier and their sensor movements are minimized. Then we construct a weighted barrier graph with the virtual-clusters modeled as vertexes and the weight of each vertex as the total sensor movements of the virtual-cluster. We also prove that the minimum total sensor movements for achieving barrier coverage is the minimum total weights of the path between the two endpoints of the line barrier in this graph. We also solve this barrier coverage problem for the case when the barrier is a cycle by extending the techniques used for the line barrier. Finally, we demonstrate the effectiveness and efficiency of our algorithms by simulations.

INDEX TERMS Barrier coverage, MinSum, mobile sensors, sink-based deployment.

I. INTRODUCTION

Wireless sensor network has received a great interest in applications such as border surveillance, battlefield surveillance and critical infrastructure security. Barrier coverage is an important coverage model in wireless sensor network, which can provide a sensor-chain barrier for detecting the intruders crossing the boundaries of the surveillance area. Sensors are often randomly dispersed from the airplane. However, it is difficult to achieve barrier coverage only using stationary sensors. Recently, with the development of mobile sensors (e.g., Robomote [1], Packbot [2] and Khepera [3]), it is possible to deploy mobile sensors in practical applications. Mobile sensors can move to the desired positions for constructing a sensor-chain barrier, which can save a lot of stationary sensors. However, the movement of mobile sensors consumes much more energy than sensor's sensing and communication. Most of sensors are equipped with batteries, thus it is a critical problem how to minimize the sensor movement for saving the energy, thus prolonging the network's lifetime.

A widely-used random sensor deployment model assumes that sensors, dispersed from an airplane, are randomly distributed in a deployed area [4]. This deployment model requires that sensors are dispersed by a airplane flying at a low altitude because of sensors' small measurement and light weight; otherwise, sensors may have a very large offset from their target landing points, and some sensors cannot

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenliang Zhang.

communicate with other sensors. In some applications such as the battlefield surveillance and nuclear leakage monitoring, it is impossible that the airplane is flying at a low altitude and thus a new deployment model should be explored. We assume that a sink station and some mobile sensors are packed up as a package. These packages are dispersed by a airplane, which are distributed randomly in a deployed area. After these packages are deployed, the sink stations communicate with their neighbor sink stations, then compute the final positions of mobile sensors and schedule the sensors to move to the desired positions for achieving barrier coverage. Since the mobile sensors may have quite different moving distances, the mobile sensors in a package are equipped with different initial energies. The sensor with more initial energy is sent to a farther final position. This kind of deployment model is called as the sink-based deployment model, which has some advantages. First, sinks can communicate with neighbor sinks easily for their large communication range. Second, sinks can compute the final positions of sensors for their powerful computing ability and storage capacity. Third, fewer mobile sensors are used than the random deployment model.

This model was first studied in the target coverage problem in the work [5]. It assumed that initially all the sensors are located at k sink stations and proposed a polynomial-time approximation scheme to minimize the moving distance of sensors to cover all targets in the surveillance region. However, little work has been studied in the barrier coverage problem based on this deployment model and we are the first to study this problem. Given a line barrier and n sink stations that can supply a required number of sensors, we study the barrier coverage problem which aims to find the mobile sensors' final positions on the line barrier so that the barrier is covered and the total sensor movement is minimized. We find that the barrier coverage problem under the sink-based deployment model can be solved by a polynomial-time algorithm. The challenge of this problem is that the sensors can move to arbitrary point of the line barrier.

The main contributions of this paper are summarized as follows:

1) To the best of our knowledge, we are the first to study the barrier coverage problem under the sink-deployment model. This model is suitable for the scenario that sensors can only be dispersed from a airplane flying at a high altitude.

2) We present a fast algorithm for determining the nearest sink for a given point on the barrier and a greedy algorithm for finding the final positions of sensors to cover the line barrier energy-efficiently.

3) We propose an optimal algorithm for finding the final positions of sensors to cover the line barrier while minimizing the total sensor movements. First, we define a concept of virtual-cluster which represents a subset of sensors covering a specified line segment of the barrier and their sensor movements are minimized. Then we construct a weighted barrier graph with the virtual-clusters modeled as vertexes and the weight of each vertex as the total sensor movements of the virtual-cluster. Two vertexes have an edge if these two corresponding sensor-clusters overlap. We prove that the minimum total sensor movements for achieving barrier coverage is the minimum total weights of the path between the two endpoints of the line barrier in this graph.

4) By extending the techniques used for the line barrier coverage, we also propose a polynomial-time algorithm for the barrier coverage problem when the barrier is a cycle.

5) We conduct simulations to evaluate the performance of our algorithms.

The remainder of this paper is organized as follows. We present the previous work in Section 2. The network model and the problem definition are given in Section 3. We present a fast algorithm for determining the nearest sink for a given point and then propose a greedy algorithm for the line barrier coverage problem in Section 4. An optimal polynomial-time algorithm is also proposed for the line barrier coverage problem in Section 5. We propose an optimal algorithm for the cycle barrier coverage problem in Section 6. Extensive performance evaluations of our algorithms are presented in Section 7. Finally, we conclude this paper in Section 8.

II. PREVIOUS WORK

Barrier coverage in wireless sensor network has been studied for over ten years. Most of the work focuses on stationary sensor network [6]–[11]. Since the mobile sensors can move to the desired positions, fewer sensors are used to achieve barrier coverage. Recently, the researchers turn to study how to deploy the mobile sensor to achieve barrier coverage in mobile sensor network, especially minimizing the sensor movement.

Some work studied the barrier coverage problem in one-dimensional setting. Suppose the barrier is a line segment and all sensors are initially located on the line containing the barrier. Some work studied the minimum maximum sensor movement problem (MinMax). The work [12] first proposed an $O(n^2)$ time algorithm for solving this problem when the sensing range of sensors are uniform. The work [13] improved the time complexity to $O(n \log n)$ for the uniform case, and also proposed an $O(n^2 \log n)$ time algorithm when the sensing ranges of sensors are arbitrary. The work [14] studied the minimum total sensor problem (MinSum). It proposed an $O(n^2)$ time algorithm when the sensing ranges of sensors are uniform and proved the problem is NP-complete by reduction to the 3-partition problem for the general case. The work [15] studied the minimum sensor number problem (MinNum). It proved the problem is NP-hard for the general case and proposed efficient algorithms for the uniform case.

Some work studied the barrier coverage problem in two-dimensional setting. Suppose the barrier is a line segment and all sensors are initially located in a plane containing the barrier. The minimum total sensor problem (MinSum) was studied in [16] and was proved to be NP-hard when the sensing range of sensors are arbitrary and can be solved in $O(n^2)$ time when the sensors can only move vertically

to the barrier. The minimum maximum sensor movement problem (MinMax) was studied in [16] and was proved be NP-hard when the sensing range of sensors are arbitrary. This problem can be solved in $O(n \log n)$ time when the sensors can only move vertically to the barrier. The work [17] first studied the MinMax problem when the sensing ranges of sensors are uniform and presented an optimal algorithm. The work [18] studied how to form the maximum number of barriers and also minimize the maximum sensor movement when the number of sensors is given and the positions of the barriers are not known a priori. It proposed a two-phase sensor mobility scheme, which was proved order optimal.

Some work studied the case when the barrier is a circle or simple polygon. The work [19] studied the MinMax problem and proposed an $O(n^{3.5} \log n)$ time algorithm for cycle barriers and proposed an $O(mn^{3.5} \log n)$ time for polygon barriers, where m is the number of edges of the polygon. The work [19] studied the MinSum problem and proposed a PTAS. The MinMax result for polygon barriers was improved by [20], which proposed an $O(n^{2.5} \log n)$ algorithm. It also proposed an $O(n^4)$ time algorithm for the MinSum problem when the sensors are moved from the circle perimeter to a regular n-gon. The work [21] studied how to cover the targets on the line by mobile sensors while minimizing the maximum sensor movement and it was proved that this problem is NP-hard when the sensors are initially on this line and the sensing ranges of sensors are non-uniform.

The work [22] studied the hybrid sensor network composed by stationary sensors and mobile sensors. The problem is how to move mobile sensors to fill the gaps while balancing the energy consumption when the stationary sensors are deployed under the line-based deployment. The work [23] studied how many mobile sensors are needed to form k barriers when the stationary sensors are deployed and proposed an optimal algorithm. The work [24] studied the problem when there are not enough sensors to form a barrier and proposed a dynamic sensor patrolling algorithm. It proved that the total sensor movement during one time slot is minimized. The work [25] studied the dynamic barrier coverage problem by combining an inspection robot and stationary sensors and proposed a heuristic algorithm based on game theory.

III. PRELIMINARIES AND PROBLEM STATEMENT

In this section, we present the network model and the problem formulation.

A. NETWORK MODEL

We assume that the deployed area is a two-dimensional rectangular area with the length L and the width W. A barrier is a line segment starting from $[0, 0]$ to $[L, 0]$ in the deployed area. Suppose n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ are randomly deployed along the barrier, whose positions are $\{(x_i, y_i) | i = 1, 2, \ldots, n\}$. The sink stations can supply a required number of sensors and send the sensors to locate at one point on the barrier for achieving barrier coverage. Sensors can move in any direction. The movement of sensor s_i, denoted by d_i, is the Euclidean distance of the sensor's initial position p_i and its final position p_i', that is $d_i = dist(p_i, p_i')$. The sensing range of the sensors are denoted as r.

We also study the case when the barrier is a cycle. The cycle barrier is located in the deployed area with the center at $[x_0, y_0]$ and the radius R. Let p denote the point on the circle barrier, then it can be represented by a 4-tuple $< R, \theta, x_i, y_i >$, where (R, θ) is polar coordinate of p and the (x_i, y_i) is the rectangular coordinate of p. We choose the point with its polar coordinate $[R, 0]$ as the origin of B denoted as p_0. Suppose n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ are randomly deployed along the cycle. The sink stations can supply a required number of sensors and send the sensors to locate at one point on the cycle for achieving barrier coverage.

B. PROBLEM DEFINITION

Our barrier coverage problem focuses on how to schedule the sensors to cover the barrier so that the total sensor movement is minimized.

Suppose the barrier is a line segment, we define n-Sink Minimum Sum of Movement for Line Barrier Coverage (L-MSBC) problem as follows:

Definition 1 (n-Sink Minimum Sum of Movement for Line Barrier Coverage Problem, Short for L-MSBC Problem): There are n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ which send mobile sensors to cover the line barrier. The L-MSBC problem is to schedule the sinks to send the mobile sensors $S = \{s_1, s_2, \ldots, s_m\}$ to locate on the line barrier so that this line barrier is covered by the sensing ranges of the mobile sensors and the total sensor movement $\sum_{i=1}^{m} \{dist(p_i, p_i')\}$ is minimized.

Suppose the barrier is a cycle, we define n-Sink Minimum Sum of Movement for Cycle Barrier Coverage (C-MSBC) problem as follows:

Definition 2 (n-Sink Minimum Sum of Movement for Cycle Barrier Coverage Problem, Short for C-MSBC Problem): There are n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ which send mobile sensors to cover the cycle barrier. The C-MSBC problem is to schedule the sinks to send the mobile sensors $S = \{s_1, s_2, \ldots, s_m\}$ to locate on the cycle barrier so that this barrier is covered by the sensing ranges of the mobile sensors and the total sensor movement $\sum_{i=1}^{m} \{dist(p_i, p_i')\}$ is minimized.

IV. THE FAST ALGORITHM FOR THE L-MSBC PROBLEM

In this section, we first present a fast algorithm for determining the nearest sink for a given point on the barrier and then propose a greedy and fast algorithm for the L-MSBC problem.

A. THE ALGORITHM FOR DETERMINING THE NEAREST SINK

Given a point on the barrier, how to find the nearest sink to send a sensor to locate at this point?

A trivial way is to compute all the distances between this target point and each sink, and then find the nearest sink for

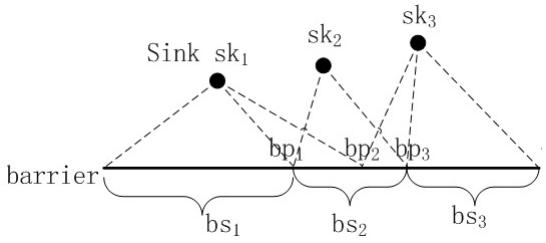


FIGURE 1. Illustration of b_segments.

this point. Obviously, at least $\lceil L/2r \rceil$ sensors are needed to cover the barrier. Thus, it costs $O(nL/2r)$ time to find the nearest sinks for all the sensors covering the barrier. If L is a big number, this method is time consuming. We'll show a fast method to find the nearest sinks.

The main idea of our algorithm is to compute the set of equaldistant points on the line barrier for every pair of sinks, then find a subset of equaldistant points, called boundary-points, such that any point between two consecutive boundary-points in this subset has a same nearest sink, and compute the nearest sink for the subsegment between such two consecutive points. Given a target point, we first identify the subsegment which include it and then return the nearest sink for this subsegment as the nearest sink for this target point.

Before showing the detail of this algorithm, we first introduce some definitions.

Definition 3 (balance-Point): A point on the line barrier is called balance-point, denoted as bp_k, if the distance from sink s_i to it is equal to that from sink s_j to it.

A balance-point is an equaldistant point on the line barrier for a pair of sinks. Let BP denote the set of balance-points. For a pair of sinks s_i and s_j, we compute the balance-point bp_k by drawing the vertical bisector of the line segment $\overline{s_i s_j}$, and bp_k is the intersection point between this vertical bisector and the line barrier. For every pair of sinks, we can compute all the balance-points and sort them increasingly. As shown in Fig.1, bp_1, bp_2, bp_3 are balance-points.

Definition 4 (boundary-Point): A balance-point is called boundary-point, denoted as dp_i, if the two points dp_i-ε and $dp_i + \varepsilon$ have a different nearest sink, where ε is a very small positive number.

Let DP denote the set of boundary-points. We can compute all the boundary-points by binary searching all the balance-points and computing their nearest sink.

Lemma 5: *Any point on the subsegment bounded by two consecutive boundary-points has the same nearest sink.*

Proof: Suppose there exist two points p_{i_1}, p_{i_2} on the subsegment bounded by two consecutive boundary-points dp_{j_1}, dp_{j_2} has different nearest sink sk_{t_1}, sk_{t_2}.

It is easy to know that there is a balance-point bp_k between p_{i_1} and p_{i_2} which has the same distance to sk_{t_1} and sk_{t_2}. It implies that bp_k is also a boundary-point. It is a contradiction, since dp_{j_1}, dp_{j_2} are two consecutive boundary-points.

Therefore, the lemma is proved. ■

Definition 6 (b_Segment): A line segment is called a b_segment if its endpoints are two consecutive boundary-points.

Let bs_i denote the ith b_segment and $BS = \{bs_1, bs_2, \ldots, bs_\tau\}$ denote the set of b_segments. Let $BE = \{be_i | 1 \leq i \leq \tau\} = \{b_0, b_1\} \cup DP$ denote the set of the endpoints of b_segments, which consists of the endpoints of the line barrier and the set DP. That is $bs_i = \overline{dp_{i-1} dp_i}$.

Definition 7 (Assigned Sink): Sink s_k is said to be assigned sink of b_segment bs_i if s_k is the nearest sink of one point on bs_i.

Let as_i denote the assigned sink of b_segment bs_i and $AS = \{as_1, as_2, \ldots, as_\tau\}$ be the set of these assigned sinks. As shown in Fig.1, $\{bp_1, bp_2, bp_3\}$ is a set of balance-points. Since bp_2 is not a boundary-point, $\{bp_1, bp_3\}$ is the set of boundary-points. Thus, $\{bs_1, bs_2, bs_3\}$ is the set of b_segments and $BE = \{b_0, bp_1, bp_3, b_1\}$, where b_0 and b_1 are the endpoints of the barrier and $\{sk_1, sk_2, sk_3\}$ is the set of assigned sinks.

These b_segments have the following property:

Lemma 8: *Any two b_segments have different assigned sinks.*

Proof: We'll prove it by contradiction. Suppose there are two b_segments which have the same assigned sink.

Suppose b_segment bs_i and $bs_{(i+1)}$ have a common endpoint be_i. Sink sk_i is the assigned sink of bs_i while sink $sk_{(i+1)}$ is the assigned sink of $bs_{(i+1)}$. We claim that the line segment $sk_{i+1}be_i$ is on the right of sk_ibe_i. If not, the distance from sink sk_{i+1} to be_{i+1} is larger than that from sink sk_i to be_{i+1}, which is contradiction.

Suppose there is one b_segment bs_{i+1} between these two b_segments bs_i and bs_{i+2}. Suppose bs_i and bs_{i+2} have the same assigned sink sk_i while sk_{i+1} is the assigned sink of bs_{i+1}. As shown in Fig.2(a), we draw a circle c_i with be_i as its center and be_isk_i as its radius. We also draw another circle c_{i+1} with be_{i+1} as its centre and $be_{i+1}sk_i$ as its radius. By the claim, $sk_{i+1}be_i$ is on the right of sk_ibe_i. Thus, sk_{i+1} is in the circle c_{i+2}, which implies that $be_{i+1}sk_{i+1}$ is shorter than $be_{i+1}sk_i$. However, $be_{i+1}sk_{i+1}$ is equal to $be_{i+1}sk_i$ since be_{i+1} is the common point of bs_{i+1} and bs_{i+2}, which is a contradiction. Thus, if there is one b_segment between two b_segment, these three b_segments have different assigned sinks.

Suppose there are two b_segments bs_{i+1} and bs_{i+2} between these two b_segments bs_i and bs_{i+3}. As shown in Fig.2(b), bs_i and bs_{i+3} have the same assigned sink sk_i, while sk_{i+1} and sk_{i+2} are the assigned sinks of bs_{i+1} and bs_{i+2} respectively. We draw a circle c_i with be_i as its centre and be_isk_i as its radius. We also draw another circle c_{i+1} with be_{i+2} as its centre and $be_{i+2}sk_i$ as its radius. By the claim, sk_ibe_{i+2} is on the right of $sk_{i+2}be_{i+2}$. Thus, sk_{i+2} is in the circle c_i, which implies be_isk_{i+2} is shorter than be_isk_i. Since sk_i is the assigned sink of bs_i, be_isk_{i+2} is not shorter than be_isk_i, which is a contradiction. Thus, if there are two b_segments between two b_segments, these four b_segments have different assigned sinks.

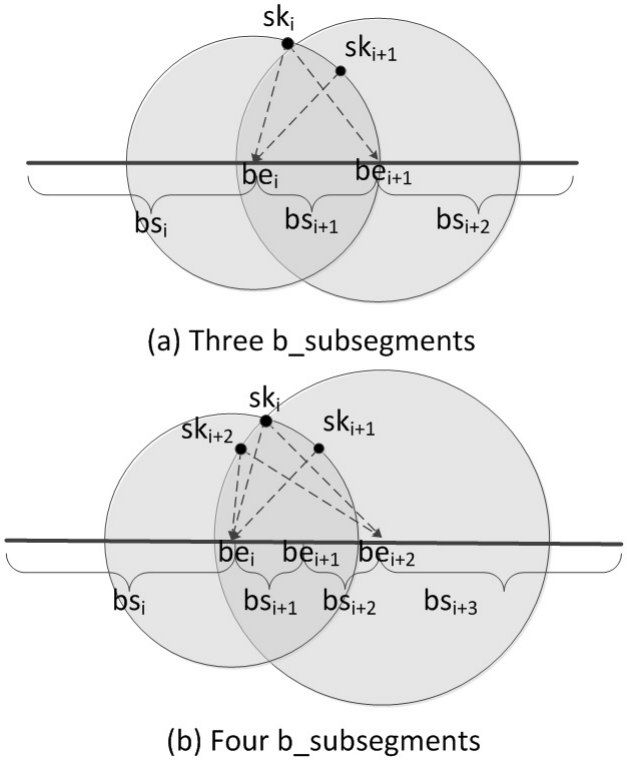


FIGURE 2. Illustration of lemma 8.

Suppose there are $m(m > 2)$ b_segments between these two b_segments. Similarly, we can prove that these two b_segments have different assigned sinks using the above method.

Therefore, the lemma is proved. ■

Next, we propose an algorithm to find all the b_segments and their assigned sinks by computing the boundary-points.

First, compute all the balance-points and sort them increasingly, denoted as $bp_1, bp_2 \ldots bp_m$; Let bp_0, bp_{m+1} denote the endpoints of the line barrier; Second, let $i = 0, j = 0$ and $be_i = bp_j$, compute the nearest sink of the point $bp_j + \varepsilon$ as sk, where ε is a small positive number; Third, use binary search technique to find the largest balance-point bp_l whose nearest sink is also sk. Let $i = i+1$ and $be_i = bp_l$. Next, if l is $m+1$, the algorithm stops; Otherwise, compute the nearest sink of the point $be_i+\varepsilon$ as sk. Go to the third step. The pseudocode of the algorithm is presented in Algorithm 1. It costs $O(n^2 \log n)$ time.

Given one point of the line barrier $(x_p, 0)$, we propose a method to find its nearest sink by binary searching the set BE and finding the index i such that $be_{i-1} < x \leq be_i$. The nearest sink of this point is as_i. The detail of this algorithm is in Algorithm 2. It costs $O(\log n)$ time.

B. GREEDY ALGORITHM FOR THE L-MSBC PROBLEM

Given the locations of sinks and a line barrier, the L-MSBC problem is how to calculate the final locations of sensors sent by sinks such that the line barrier is covered and the total sensor movements are minimized. We propose a greedy and fast algorithm for the L-MSBC problem. In the greedy algorithm, we send sensors to cover the grid points on the line barrier $G = \{r, 3r, 5r, \ldots, \}$. We always choose the closest sink to send the sensor to locate at each grid points until all the grid points are covered. This algorithm runs in $max\{O(L\log n/2r), O(n^2 \log n)\}$ time.

Algorithm 1 Algorithm for Computing the Set of b_Segments

INPUT: $SK = \{sk_1, sk_2, \ldots, sk_n\}, L$
OUTPUT: $BE = \{be_0, be_1, \ldots, be_\tau\}$, $AS = \{as_1, as_2, \ldots, as_\tau\}$
1: $bp_0 \leftarrow 0, k \leftarrow 1, AS \leftarrow \phi$;
2: for each sk_i in SK
3: for each sk_j $(i < j)$ in SK
4: calculate bp_k;
5: if $bp_k < L$
6: $k++$;
7: endif
8: endfor
9: endfor
10: $bp_k \leftarrow L$;
11: sort $bp_0, bp_1, \ldots, bp_k$ increasingly;
12: $i \leftarrow 0, be_0 \leftarrow bp_0, mid \leftarrow 0, BE \leftarrow BE \cup \{be_0\}$;
13: while $be_i ! = L$
14: $l \leftarrow mid + 1; r \leftarrow k$;
15: find the nearest sink sk_1 for the point $be_i + 0.001$;
16: while $l \leq r$
17: $mid \leftarrow (l + r)/2$;
18: find the nearest sink sk_2 for the point $bp_{mid} + 0.001$;
19: if $sk_1 \neq sk_2$
20: $r \leftarrow mid - 1$;
21: else
22: $l \leftarrow mid + 1$;
23: endif
24: endwhile
25: $be_{i+1} \leftarrow bp_{mid}$;
26: $BE \leftarrow BE \cup \{be_{i+1}\}$;
27: $AS \leftarrow AS \cup \{sk\}$;
28: $i \leftarrow i + 1$;
29: endwhile
30: return BE, AS;

V. THE OPTIMAL ALGORITHM FOR THE L-MSBC PROBLEM

In this section, we propose an optimal algorithm for the L-MSBC problem by introducing a weighted barrier graph model.

A. WEIGHTED BARRIER GRAPH

We first give some definitions as follows.

Definition 9 (Projective-Point): A point on the line barrier, denoted as pp_i, is called the projective-point of sink s_i if sink s_i's x-coordinate equals to pp_i's x-coordinate.

Algorithm 2 Algorithm for Finding the Nearest Sink When the Barrier Is a Line Segment

INPUT: BE, AS, x_p
OUTPUT: sk
1: $l \leftarrow 1$; $r \leftarrow |BE|$;
2: while $l \leq r$
3: $mid \leftarrow (l + r)/2$;
4: if $BE_{mid} > x_p$
5: $r \leftarrow mid - 1$;
6: else
7: $l \leftarrow mid + 1$;
8: endif
9: endwhile
10: return sk_{mid};

Algorithm 3 Greedy Algorithm for the L-MSBC Problem

INPUT: $SK = \{sk_1, sk_2, \ldots, sk_n\}, L, r$
OUTPUT: P, d
1: compute BE, AS using Algorithm 1;
2: $p \leftarrow r$, d$\leftarrow 0$;
3: while $p < L + r$
4: find the nearest sink sk_i to p using Algorithm 2;
5: $d+ = dist(p, sk_i)$;
6: put p into P;
7: $p = p + 2r$;
8: endwhile
9: return P, d;

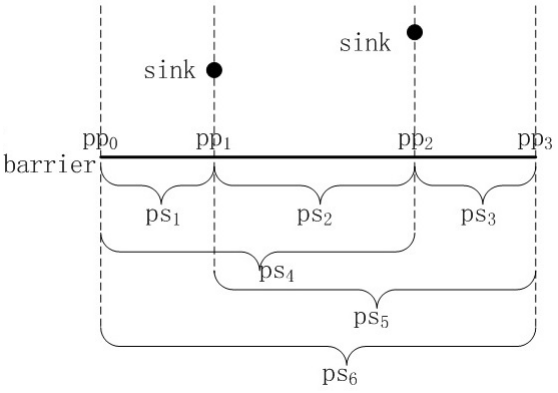


FIGURE 3. Illustration of projective-subsegments.

The line segment between two projective-points is called a projective-subsegment.

Definition 10 (Projective-Subsegment): A line segment on the line barrier, denoted as $ps_i = \overline{pp_i pp_j}$, is called a projective-subsegment if both endpoints are also the projective-points.

There are $O(n^2)$ projective-subsegments. As shown in Fig.3, pp_0, pp_1, pp_2, pp_3 are the projective-points and $ps_1, ps_2, \ldots ps_6$ are the projective-subsegments.

Lemma 11: *For a projective-subsegments $ps_k = \overline{pp_i pp_j}$, the minimum number of sensors needed for covering it, denoted as n_k^1, is $\lceil((pp_j - pp_i) - 2r)/2r\rceil$, while the maximum number of sensors needed, denoted as n_k^2, is $\lceil((pp_j - pp_i) + 2r)/2r\rceil$.*

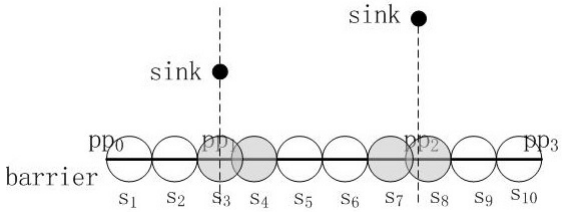


FIGURE 4. Illustration of sensor-clusters.

Proof: The line segment starting from pp_i to $pp_i + r$, denoted as $\overline{pp_i\,(pp_i + r)}$, can be covered by the sensor located within its left neighbor projective-subsegment. Similarly, $\overline{(pp_j - r)\,pp_j}$ can be covered by the sensor located within the right neighbor projective-subsegment. Thus, the minimum number of sensors needed for covering ps_k is the maximum number of sensors needed to cover the line segment $\overline{(pp_i + r)\,(pp_j - r)}$. Since these sensors are in attaching positions, $\lceil((pp_j - pp_i) - 2r)/2r\rceil$ sensors are needed. Thus, the minimum number of sensors needed is $\lceil((pp_j - pp_i) - 2r)/2r\rceil$.

Suppose the line segment $\overline{(pp_i - r)\,pp_i}$ is covered by a sensor located at ps_k and $\overline{pp_j\,(pp_j + r)}$ is covered by a sensor located at ps_k. Thus, the maximum number of sensors needed is $\lceil((pp_j - pp_i) + 2r)/2r\rceil$. ■

Definition 12: Sensors $s_{i_1}, s_{i_2}, \ldots, s_{i_q}$ are said to be in attaching positions if any pair of neighbor sensors overlap at only one point, that is $\forall 1 \leq j < q, x_{i_{j+1}} = x_{i_j} + 2r$ holds.

Lemma 13: *In a sensor deployment of minimizing the total sensor movements, the sensors are in attaching positions if their positions are within, not including, two consecutive projective-points.*

Proof: Suppose there is a sensor in these sensors which overlaps with its left neighbor sensor at more than one point. If the x-coordinate of this sensor's final position is smaller than that of this sensor's initial position, the sensor's final position can be shifted to the right a bit, which can reduce its movement; otherwise, the final position of its left neighbor sensor can be shifted to the left a bit, which can reduce its movement. It is a contradiction of the optimality.

Suppose there is a sensor in these sensors which overlaps with its right neighbor sensor at more than one point. We can also prove that it is a contradiction of the optimality.

Thus, the lemma is proved. ■

Now we define a notion of sensor-cluster.

Definition 14 (Sensor-Cluster): A set of sensors $\{s_{i_1}, \ldots, s_{i_\tau}\}$ with final positions $\left\{p'_{i_1}, \ldots, p'_{i_\tau}\right\}$ is called a sensor-cluster within $\overline{pp_i pp_j}$ if $pp_i \leq p'_{i_1} < \ldots < p'_{i_\tau} \leq pp_j$ holds and for any $k\,(1 \leq k < \tau)$ which satisfies that $p'_{i_{k+1}} - p'_{i_k} = 2r$ holds, where $n_k^1 \leq \tau \leq n_k^2$.

As shown in Fig.4, there are three sensor-clusters, which are the set of sensors s_1, s_2, s_3, the set of sensors s_4, s_5, s_6, s_7 and the set of sensor s_8, s_9, s_{10}.

Suppose the sensors' initial positions are fixed and their final positions should be within $\overline{pp_ipp_j}$, we can calculate a sensor-cluster such that the sensors' total movements are minimized.

Definition 15 (Virtual-Cluster): A sensor-cluster, consisting of sensors $\{s_{i_1}, \ldots, s_{i_\tau}\}$ with final positions $\left\{p'_{i_1}, \ldots, p'_{i_\tau}\right\}$, is called a virtual-cluster, denoted as vc_{ijl}, within $\overline{pp_ipp_j}$ if initial positions of sensors $\{p_{i_1}, \ldots, p_{i_\tau}\}$ are fixed and $\sum_{j=1}^{\tau}\left(p'_{i_j} - p_{i_j}\right)$ is minimized, where $1 \le l \le \gamma$.

We'll define the notion of a weighted barrier graph as follows.

Definition 16 (A Weighted Barrier Graph): $G = (V, E, W)$ of a mobile sensor network is constructed as follows: The set V consists of vertices corresponding to the left endpoint of line barrier (s), the virtual-clusters $\left(\{vc_{ijl}|1 \le i \le j \le n, 1 \le l \le n\}\right)$ and the right endpoint of line barrier (t). That is, $V = \{s \cup t \cup \{vc_{ijl}|1 \le i \le j \le n, 1 \le l \le n\}\}$. The set $E = \{e(v_i, v_j)\}$ consists of edges which are between two vertices if their corresponding sensor-clusters overlap. A vertex has an edge with s or t if the corresponding sensor-cluster covers the left or right endpoint of line barrier. $W : V \to \mathbb{R}$ is the set of the weights of each vertices, where the weight $w(v_i)$ of vertex v_i is the total movement of sensors in the corresponding sensor-cluster.

Then we have the following lemma.

Lemma 17: *Any path from s to t on the weighted barrier graph G means that the barrier can be covered by the corresponding sensor-clusters of the vertices on the path.*

Proof: By the definition of a weighted barrier graph, two vertices have an edge if their corresponding sensor-clusters overlap. A vertex has an edge with s or t if the corresponding sensor-cluster covers the left or right endpoint of line barrier. Thus, if there is a path from s to t on the weighted barrier graph G, there are sensor-clusters which cover the barrier from left to right. ■

Theorem 18: *The minimum total sensor movement needed to form a barrier under the sink-deployment model is exactly the minimum total weight of $s-t$ path on the weighted barrier graph G.*

Proof: Suppose the minimum total weight of $s - t$ path on the weighted barrier graph G is larger than the minimum total sensor movement needed to form a barrier. Recall that the path with the minimum total weight is composed of virtual-clusters which cover the whole line barrier.

By lemma 13 and the definition of the sensor-cluster, it implies that the sensor deployment of minimizing the total sensor movements, called the optimal sensor deployment, is composed of sensor-clusters.

Now we can produce a sensor deployment whose total sensor movement is smaller than this optimal sensor deployment. We check the sensor-clusters in the optimal sensor deployment from the left to the right.

If the ith sensor-cluster is not a virtual-cluster, we can replace this sensor-cluster by the corresponding virtual-cluster. If there is a gap between it and its left neighbor virtual-cluster, then these two virtual-clusters can be replaced by the corresponding virtual-cluster; If there is still a gap between this new virtual-cluster and its left neighbor, continue replacing these two virtual-clusters by the corresponding virtual-cluster until there is no gap between it and its left neighbor. Since the virtual-cluster is a sensor-cluster with the minimum total sensor movement, thus the total sensor movement can be decreased by the replacement of virtual-clusters. Thus, a sensor deployment consisting of virtual-clusters is produced, whose total sensor movement is smaller than the optimal sensor deployment, which is a contradiction.

Thus, the Theroem is proved. ■

B. ALGORITHM DESCRIPTION

In this subsection, we'll present an optimal algorithm of the L-MSBC problem.

The main idea of this algorithm is to compute all possible projective-subsegments, then compute the virtual-clusters for each projective-subsegment, and construct a weighted barrier graph with the virtual-clusters as the vertices. The optimal sensor deployment is the vertices on the path of the minimum total weights.

1) COMPUTING THE VIRTUAL-CLUSTERS

In this subsection, we'll show how to compute the virtual-clusters after the projective-subsegments are calculated.

Recall that the virtual-clusters are the optimal deployments of sensors located within the projective-subsegment. The difficulty to compute the virtual-clusters is that the initial positions of sensors deployed are unfixed. We solve this problem by dividing the range of the sensors' final positions into some subranges such that the initial-positions of sensors are fixed. In each subrange, we can compute the optimal sensor deployment by using optimization technique as a virtual-cluster.

First, we calculate the range of sensors' final positions. For a projective-subsegments $ps_k = \overline{pp_ipp_j}$, the final positions of the sensors within it should satisfy the two conditions:

1) Their final positions should be within the projective-subsegment;

2) These sensors should cover the line segment starting from $pp_i + r$ to $pp_j - r$;

By lemma 11, the minimum number of sensors needed for covering the projective-subsegment $ps_k = \overline{pp_ipp_j}$ is n_k^1. We calculate the range of the sensors' final position as follows:

1) Suppose these sensors are in attaching positions. We consider the final position of first sensor, which is called archor-sensor. We can calculate the range of the archor-sensor's final position, called as sensor-range, as follows:

Case 1: Suppose the number of sensors in the virtual-cluster is n_k^1. If $\left(pp_j - pp_i\right)\%2r = 0$, then the sensor-range is $[2r + pp_i, 2r + pp_i]$; Otherwise, the sensor-range is $\left[\left(pp_j - pp_i\right)\%2r + pp_i, 2r + pp_i\right]$.

Case 2: Suppose the number of sensors in the virtual-cluster is $n_k^1 + 1$. If $\left(pp_j - pp_i\right)\%2r = 0$, then the

sensor-range is $[pp_i, 2r + pp_i]$; Otherwise, the sensor-range is $\left[pp_i, \left(pp_j - pp_i\right) \%2r + pp_i\right]$.

2) Suppose the first sensor is located the projective-point and the other sensors are in attaching positions. Then we choose the second sensor as the archor-sensor. This case can be calculated as Case 1.

3) Suppose the last sensor is located the projective-point and the other sensors are in attaching positions. Then we choose the first sensor as the archor-sensor. This case can be calculated as Case 1.

Second, we divide the sensor-range into some subranges, called as sensor-subrange, such that the initial-positions of sensors are fixed when the archor-sensor is located in this subrange.

The sensor-subranges are calculated as follows:

1) check out all the boundary-points within this subsegment;

2) calculate the anchor-sensor's final position, called change-point, such that there is a sensor in this sensor-cluster exactly located at one boundary-point;

3) these change-points divide the sensor-range into sensor-subranges.

Suppose sensor-range is $[a, b]$ and the boundary-point is bp_i. Then the change-point cp_j can be calculated as $(bp_i - a)\%2r + a$. If cp_j is larger than b, this change-point is not within the sensor-range and ignored.

Third, we compute the optimal sensor deployment when the archor-sensor is located with a sensor-subrange.

Suppose the sensors for covering $ps_k = \overline{pp_i pp_j}$ are $s_1, s_2, \ldots s_m$. The archor-sensor is located in one sensor-subrange $[a_1, b_1]$ with initial position (x_1, y_1) and final position $(x, 0)$. Let y_{p_1}, y_{p_2} be the sensor movement when a sensor is located at the left and right endpoints of for ps_k respectively. Depending on four cases, we calculate the minimum total sensor movement as follows:

(1) When the number of sensors is $m = n_k^1$, the minimum total sensor movement M_1 is calculated as follows:

If $\left(pp_j - pp_i\right) \%2r = 0$,

$M_1 \leftarrow \sum_{i=1}^{m} \sqrt{(x_1 - (pp_i + 2r) - (i-1) * 2r)^2 + y_1^2}$

else

$M_1 \leftarrow Minimize \sum_{i=1}^{m} \sqrt{(x_1 - x - (i-1) * 2r)^2 + y_1^2}$

s.t. $x \in [a_1, b_1] \subseteq \left[\left(pp_j - pp_i\right) \%2r + pp_i, 2r + pp_i\right]$

Then optimization method, such as interpolation method, can be adopted to find the minimum total sensor movement value.

(2) When the number of sensors is $m = n_k^1 + 1$ and a sensor is located at the left endpoint of the projective-subsegment, the minimum total sensor movement $M_2 = M_1 + y_{p_1}$.

(3) When the number of sensors is $m = n_k^1 + 1$ and there is no projective-sensor, the minimum total sensor movement M_3 is calculated as follows:

If $\left(pp_j - pp_i\right) \%2r = 0$,

$M_3 \leftarrow \sum_{i=1}^{m} \sqrt{(x_1 - (pp_i + r) - (i-1) * 2r)^2 + y_1^2}$

else

$M_3 \leftarrow minimize \sum_{i=1}^{m} \sqrt{(x_1 - x - (i-1) * 2r)^2 + y_1^2}$

s.t. $x \in [a_1, b_1] \subseteq \left[pp_i, \left(pp_j - pp_i\right) \%2r + pp_i\right]$

Then optimization method, such as interpolation method, can be adopted to find the minimum total sensor movement value.

(4) When the number of sensors is $m = n_k^1 + 1$ and a sensor is located at the right endpoint of the projective-subsegment, the minimum total sensor movement $M_4 = M_1 + y_{p_2}$.

Finally, we'll compute all the virtual-clusters.

The virtual-cluster is denoted by $< M, l, r, indx >$. Let M denote the minimum total sensor movement of this cluster, while l(resp. r) is represented as the leftmost(resp. rightmost) covering point of this cluster. Let $indx$ denote the index of the projective-subsegment. The left endpoint of the line barrier is also regarded as a virtual-cluster denoted by $< 0, 0, 0, 0 >$, while the right endpoint of the line barrier is also regarded as a virtual-cluster denoted by $< 0, L, L, size(PS) + 1 >$. Note that $size(PS)$ is the number of the projective-subsegments.

There are $O(n^2)$ projective-subsegments. Besides, there are $O(n)$ sensor-subranges for each projective-subsegment. Thus, there are $O(n^3)$ virtual-clusters and it costs $O(n^3 C)$ time to calculate all virtual-clusters, where C is the time consumed by the optimization method adopted to find the minimum total sensor movement value.

2) CONSTRUCTING WEIGHTED BARRIER GRAPH

Now we'll show how to construct the weighted barrier graph.

We construct a weighted barrier graph $G = (V, E)$. Each node in V represents a virtual-cluster and its weight is the virtual-cluster's M value. Let s and t be the virtual-clusters of the left and right endpoint of the line barrier respectively. The weight of each edge is set as follows:

Initially, the weight of each edge is set to be infinity.

For two nodes $vc_i = \{M_i, l_i, r_i, indx_i\}$ and $vc_j = \left\{M_j, l_j, r_j, indx_j\right\}$, the weights of the edges $e(i, j)$ and $e(j, i)$ are calculated as follows:

1) If $indx_i \neq indx_j$, $r_j \geq r_i$, $r_i \geq l_j$ and $l_j > l_i$ hold, $e(i, j)$ is set to be M_j;

2) if $indx_i \neq indx_j$, $r_j < r_i$, $r_i \leq l_j$ and $l_j < l_i$ hold, $e(j, i)$ is set to be M_i;

Our optimization problem is converted into finding a path with the minimum total weights from s to t in G. We use Djikstra algorithm to find the minimum total weight path, which represents the minimum total sensor movement. The Djikstra algorithm costs $O(E + V\lg V)$ time.

Theorem 19: *The L-MSBC problem can be solved in* $O(n^3 lgn)$ *time.*

Proof: By Theorem 18, Algorithm 4 can find the minimum total sensor movements for the L-MSBC problem.

There are $O(n^3)$ virtual-clusters, thus there are $O(n^3)$ vertices. There are $O(n^3)$ edges, since only two vertices belonging to the neighbor projective-subsegments have an edge.

Thus, Algorithm 4 runs in $O(n^3 \lg n)$ time.

Therefore, the Theorem is proved. ■

Algorithm 4 The Optimal Algorithm for the L-MSBC Problem

Input: $SK = \{sk_1, \ldots sk_n\}, L$
Output: M // the total sensor movement
1: Generate all possible projective-subsegments;
2: Generate all virtual-clusters;
3: Generate the weighted barrier graph $G = (V, E)$.
4: Run djikstra algorithm on this graph and find a path with the minimum total weights from s to t in G.
5: return the minimum total weights of the path as M;

VI. THE OPTIMAL ALGORITHM FOR THE C-MSBC PROBLEM

In this section, we study the C-MSBC problem for achieving barrier coverage when the barrier is a cycle by extending the technique used for line barrier coverage.

A. OVERVIEW

Although the cycle barrier coverage problem is as similar as the line barrier coverage problem, the solution for line barrier coverage cannot be directly applied to solve the cycle barrier coverage case. There are some challenging issues. First, how to determine the starting point and ending point of the cycle barrier as the line barrier? Second, even though these two points are determined, there exists a sensor in the sensor deployment which may cover both the starting point and ending point. How to solve this special case? Third, we cannot sort the points on the circle barrier by their x-coordinates like the line barrier.

We'll study these issues and try to extend the technique used for line barrier coverage to solve the circle barrier coverage problem. The basic idea is to introduce weighted barrier graph and find the minimum weight cycle on the weighted barrier graph to obtain the minimum total sensor movement needed to form a cycle barrier.

B. DETERMINING THE NEAREST SINK

In this subsection, we propose a method to determine the nearest sink for one point of the cycle barrier.

We adopt the algorithm from the line barrier coverage, which is to first compute the boundary-points and then find all the b_segments and the assigned sinks. However, there are two differences between the line barrier coverage and cycle barrier coverage. First, there may be two boundary-points for each pair of sinks, thus the algorithm for the line barrier can not be directedly applied to the cycle barrier coverage. It implies that two b_segments may have the same assigned sinks. Second, there may exist a b_segment which crosses the starting point and ending point of the cycle barrier.

To handle these two challenges, we propose an algorithm.

We choose the point with its polar coordinate $[R, 0]$ as the origin of B denoted as bp_0 and $bp_{\mu+1}$. Then calculate all the boundary-points, and then sort these boundary-points by their polar angles, denoted by $bp_1, \ldots bp_\mu$. Our algorithm considers the set of boundary-points $BP = \{bp_0, bp_1, \ldots bp_\mu, bp_{\mu+1}\}$. For simplicity, $bp_i = \theta_i$, where θ_i is the polar angle of the point bp_i. The basic idea is to compute all b_segments satisfying that the neighbor b_segments share different assigned sinks. Let bs_i denote the ith b_segment and $BS = \{bs_1, bs_2, \ldots, bs_\tau\}$ denote the set of b_segments. Let $BE = \{be_0, be_1, \ldots be_\tau\}$ denote the set of the b_segments' endpoints. That is $bs_i = \overset{\frown}{be_{i-1}be_i}$. The procedures of the algorithm are described as follows:

1) Initialize BE as an empty set. Let $i = 0, j = 0$ and $be_0 = bp_0$.

2) Calculate the assigned sink sk which is the nearest sink of the point $be_i + 0.001$.

3) Let $j = j + 1$. Check the boundary-point $bp_j + 0.001$ whether its nearest sink is also sk. If yes, go to 3); Otherwise, let $i = i + 1$ and $be_i = bp_j$.

4) If j is $\eta + 1$, stop; otherwise, go to 2).

The pseudocode of the algorithm is presented in Algorithm 5. It costs $O(n^3)$ time.

Algorithm 5 Algorithm of Finding the Nearest Sink When the Barrier Is a Cycle

INPUT: $SK = \{sk_1, sk_2, \ldots, sk_n\}$
OUTPUT: $BE = \{be_0, be_1, \ldots, be_\tau\}$, $AS = \{as_1, as_2, \ldots, as_\tau\}$
1: $bp_0 \leftarrow 0, k \leftarrow 1, BE \leftarrow \phi, AS \leftarrow \phi$;
2: for each sink sk_i
3: for each sink sk_j $(i < j)$
4: calculate bp_k;
5: $k++$;
6: endfor
7: endfor
8: $bp_k \leftarrow 2\pi$;
9: sort $bp_0, bp_1, \ldots, bp_k$ increasingly by their angles;
10: $i \leftarrow 0, j \leftarrow 0, be_0 \leftarrow bp_0$; $BE \leftarrow BE \cup \{be_0\}$;
11: calculate the nearest sink sk_1 of the point $be_0 + 0.001$.
12: while $bp_j! = bp_k$
13: calculate the nearest sink sk_2 of the point $bp_j + 0.001$.
14: if $sk_1 \neq sk_2$
15: $BE \leftarrow BE \cup \{bp_j\}$;
16: $AS \leftarrow AS \cup \{sk_1\}$;
17: $i++$;
18: $sk_1 \leftarrow sk_2$;
19: endif
20: $j \leftarrow j + 1$;
21: endwhile
22: return BE, AS;

C. CALCULATING PROJECTIVE-SUBSEGMENTS

In this subsection, we present a method to divide the cycle barrier into segments called projective-subsegments.

We first calculate the projective-points. Note that for each sink, there are two projective-points and we only choose

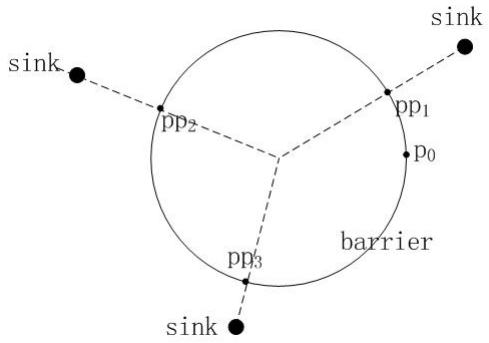


FIGURE 5. Illustration of projective-points on the cycle barrier.

the nearer projective-point. Let $PP = \{pp_1, \ldots pp_\tau\}$ denote the set of projective-points. The p_segments are the arcs satisfying that the endpoints of the arc are two projective-points. Notes that for any pair of two projective-points, there may exist two p_segments and one p_segment may cross the starting point and ending point of the cycle barrier, which is named crossed p_segment. As seen in Fig 5, the projective-subsegment $\widehat{pp_3pp_1}$ crosses the point p_0. To handle the crossed p_segments, we add a duplication of the projective-points to the set of projective-points, with the polar angle added by 2π. Sort the projective-points by their polar angles and the set of projective-points is denoted by $PP = \{pp_1, pp_2, \ldots, pp_{2(n-1)}, pp_{2n}\}$ with their polar angles $\{\theta_1, \theta_2, \ldots 2\pi + \theta_{n-1}, 2\pi + \theta_n\}$. A projective-subsegment is denoted by $ps_i = \widehat{pp_jpp_k}$ $(j < k)$ and Let $PS = \{ps_1, ps_2, \ldots\}$ denote the set of p_segments.

Then we enumerate all possible projective-subsegments and there are four types of projective-subsegments. Let pp_i denote the starting endpoint of one projective-subsegment and pp_j be the ending endpoint.

For the first type of projective-subsegment, $pp_i < 2\pi$ and $pp_j < 2\pi$ hold.

For the second type of projective-subsegment, $pp_i \geq 2\pi$ and $pp_j \geq 2\pi$ hold.

For the third type of projective-subsegment, $pp_i < 2\pi$, $pp_j > 2\pi$ and $pp_j - pp_i \leq 2\pi$ hold.

For the fourth type of projective-subsegment, $pp_i < 2\pi$, $pp_j > 2\pi$ and $pp_j - pp_i > 2\pi$ hold.

It is easy to know that the second type and the fourth of projective-subsegment can be transformed into the first type of projective-subsegment. Thus, we only choose the first type and the third type of projective-subsegments and put them into the set PS, which has $O(n^2)$ projective-subsegments.

The procedures of the algorithm are described as follows:

1) Initialize PS and PP as an empty set.

2) For each sink, compute its projective-points. Choose the nearer projective-point and add it to PP. We also add a duplication of this projective-point to PP, with the polar angle added by 2π.

3) Sort these projective-points by their polar angles.

4) For each pair of pp_i and pp_j $(i < j)$, if $pp_i < 2\pi$ and $pp_j - pp_i \leq 2\pi$ hold, then add the p_segment $\overline{pp_jpp_k}$ into PS.

D. DEFINING VIRTUAL-CLUSTER

Now we compute all the virtual-clusters after calculating the projective-subsegments. First, we compute the minimum number of sensors within each projective-subsegment. Similar as the line barrier, we divide the range of sensors' final positions into subranges such that the sensors' initial positions are fixed. We use the polar angle of the archor-sensor' final position instead of its x-coordinate and use $arcsin(r/R)$ instead of r. The sensor-range and the sensor_subrange is calculated as the line barrier.

Suppose the archor-sensor is located in one sensor-subrange $[a_1, b_1]$ with final position (R, θ_{t_1}). Let y_{p_1}, y_{p_2} be the polar radius of the initial position of the sensors which will be located at the left and right endpoints of the projective-subsegment. Let $rr = arcsin(r/R)$. We calculate the ith sensor's initial position and final position. The ith sensor's final position is calculated as $(R, \theta_{t_1} + (i-1) * 2rr)$. That is, the x-coordinate of the ith sensor is $x_i' = Rcos(\theta_{t_1} + (i-1) * 2rr)$ and the y-coordinate is $y_i' = Rsin(\theta_{t_1} + (i-1) * 2rr)$. The ith sensor's initial position is the position of its assigned sink denoted by (x_{t_i}, y_{t_i}). The virtual-cluster is calculated as the line barrier. There are $O(n^3)$ virtual-clusters and it costs $O(n^3C)$ time to calculate all virtual-clusters, where C is the time consumed by the optimization method adopted to find the minimum total sensor movement.

E. CONSTRUCTING WEIGHTED BARRIER GRAPH

We construct a directed weighted barrier graph $G = (V, E)$ and the C-MSBC problem can be converted into finding a cycle with the minimum total weights in G. Now we'll show how to construct the directed weighted barrier graph $G = (V, E)$. Each node in V represents a virtual-cluster and its weight is the virtual-cluster's M value. The weight of the edge of two nodes is not infinity if the corresponding virtual-clusters belongs to different projective-subsegments and they are connected. The weight of the edges are set as follows:

Initially, the weight of all the edges are set to be infinity.

For two nodes $vc_i = \{M_i, l_i, r_i, indx_i\}$ and $vc_j = \{M_j, l_j, r_j, indx_j\}$, the weight of the edges $e(i, j)$ and $e(j, i)$ are calculated as follows:

1) If $indx_i \neq indx_j$, $r_j \geq r_i$, $r_i \geq l_j$ and $l_j > l_i$ hold, $e(i, j)$ is set to be M_i;

2) if $indx_i \neq indx_j$, $r_j < r_i$, $r_i \leq l_j$ and $l_j < l_i$ hold, $e(j, i)$ is set to be M_j;

3) if $indx_i \neq indx_j$, $r_i \geq 2\pi$, $r_j \geq r_i - 2\pi$ and $r_i - 2\pi > l_j$ hold, $e(i, j)$ is set to be M_i;

4) if $indx_i \neq indx_j$, $r_j \geq 2\pi$, $r_i \geq r_j - 2\pi$ and $r_j - 2\pi > l_i$ hold, $e(j, i)$ is set to be M_j;

5) if $r_i - l_i \geq 2\pi$, $e(i, i)$ is set to be M_i;

Our optimization problem is converted into finding a cycle with the minimum total weights in G. We can use the algorithm in [26] to find the minimum weight cycle, which runs in $O(VE)$ time.

Theorem 20: *The minimum total sensor movement needed to form a cycle barrier is exactly the total weights of the minimum weight cycle on the weighted barrier graph G.*

Proof: Suppose there is an optimal sensor deployment of minimum total sensor movement forming a cycle barrier, whose sensor movement is smaller than that of the minimum weight cycle on G.

It is easy to know that there is a sensor-cluster in the optimal sensor deployment, which is not a virtual-cluster. By using the similar method in Theorem 19, we can also produce a sensor deployment which has a smaller sensor movement than the optimal sensor deployment, which is a contradiction.

Thus, the Theorem is proved. ■

Theorem 21: *The C-MSBC problem can be solved in* $O(n^6)$ *time.*

Proof: By Theorem 20, the C-MSBC problem can be transformed to finding the minimum weight cycle on the weighted barrier graph G. We can use the algorithm in [26] to find the minimum weight cycle, which runs in $O(VE)$ time. There are $O(n^3)$ vertices and $O(n^3)$ edges in G. Thus, the C-MSBC problem can be solved by $O(n^6)$ time. ■

VII. EVALUATION

In this section we evaluate the performance of the proposed algorithms by simulation in two different cases.

A. THE LINE BARRIER CASE

We first evaluate the solution for the L-MSBC problem called SMC solution. The sinks are deployed randomly in a belt region with length L and width W. The barrier is a line segment in the belt region starting from $[0, 0]$ to $[L, 0]$. L is set to be 1057 and W is 30. The number of sinks is 5. Each sink can send sensors to cover the barrier. The sensing range of sensors is 22. We mainly focus on the minimum total sensor movement for barrier coverage. This metric is evaluated on the length of the line barrier, the width of the deployed area, the number of sinks and the sensor's sensing range. We compare the SMC algorithm with the greedy algorithm, which run 100 times. The data points are a average of 100 experiments.

Fig 6 shows the effect of the length of line barrier on the minimum total sensor movement. The length of the line barrier varies from 177 to 1200 with step 176. As the length of the line barrier increases, the minimum total sensor movement by two algorithms both increases. The reason is that more sensors are needed to cover the line barrier . We can see that the minimum total sensor movement by SMC is always smaller than the result by the Greedy algorithm, which implies that the SMC algorithm outperforms the Greedy algorithm.

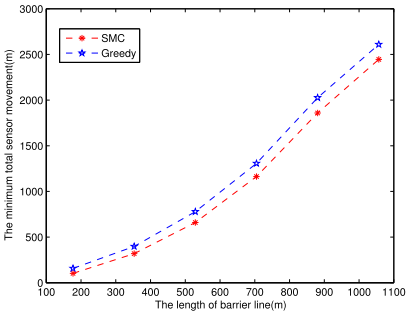


FIGURE 6. The length of the line barrier changes.

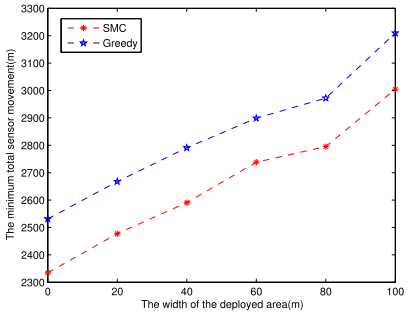


FIGURE 7. The width of the deployed area changes.

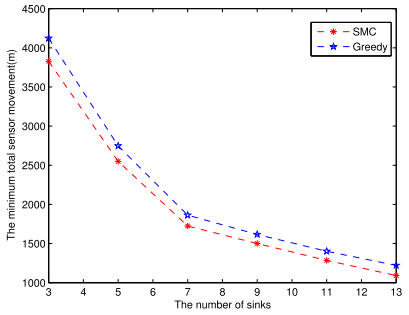


FIGURE 8. The number of sinks changes.

Fig 7 shows the effect of the width of the deployed area on the minimum total sensor movement. The width of the deployed area varies from 0 to 100 with the step 20. With the increasing of the width of the deployed area, the minimum total sensor movement by two algorithms both increases nearly linearly. We can see that the result obtained by SMC algorithm is about 90 percent of that by Greedy algorithm.

Fig 8 shows the effect of the number of sinks on the minimum total sensor movement. The number of sinks varies from 3 to 13 with the step 2. With the increasing of the number of sinks, the minimum total sensor movement by two algorithms both decreases sharply until the number of sinks approaches 7. After passing this value, the result decreases slowly. This demonstrates a tradeoff between the number of sinks and the minimum total sensor movement required to achieve barrier coverage. The result by SMC is always smaller than the result by the Greedy algorithm.

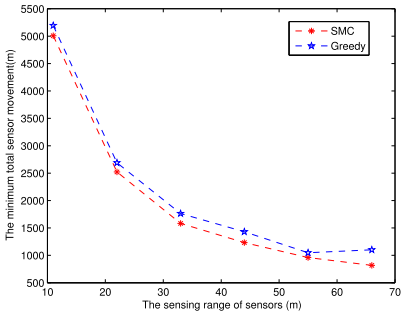


FIGURE 9. The sensing range of sensors changes.

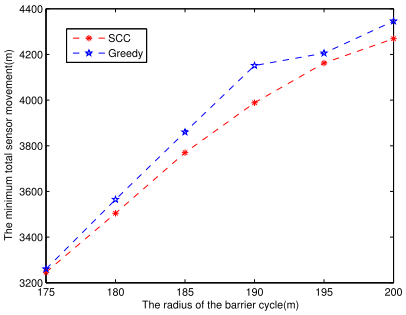


FIGURE 10. The radius of the barrier cycle changes.

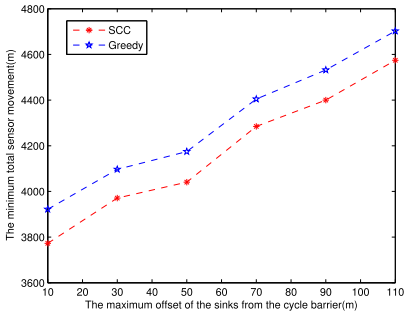


FIGURE 11. The maximum offset of the sinks from the cycle barrier changes.

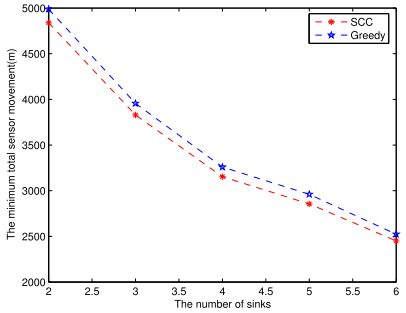


FIGURE 12. The number of sinks changes.

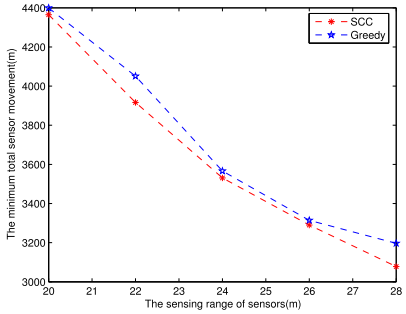


FIGURE 13. The sensor's sensing range changes.

Fig 9 shows the effect of the sensing range of sensors on the minimum total sensor movement. The sensing range of sensors varies from 11 to 66 with 11 as the step size. As the sensing range of sensors increases, the minimum total sensor movement decreases. That's because more sensors are needed to achieve barrier coverage when the sensing range increases. The result first decreases sharply, and then decreases slowly. When the sensing range is larger than 55, the effect on the result is not significant.

B. THE CYCLE BARRIER CASE

We evaluate the solution for the C-MSBC problem called SCC solution. The barrier is a cycle with the radius R. The sinks are deployed randomly along the cycle barrier with the maximum offset W. Each sink can send sensors to cover the cycle barrier. R is set to be 190 and W is set to be 22. The sensing range of sensors, denoted as r, is set to be 22. The number of sinks is 3. We mainly focus on the minimum total sensor movement for barrier coverage. This metric is evaluated on the radius of the barrier cycle, the maximum offset of the sinks from the cycle barrier, the number of sinks and the sensor's sensing range. As similar as the line barrier case, we also choose the greedy algorithm for comparison. The greedy algorithm and the SCC algorithm are run 100 times. The data points are a average of 100 experiments.

Fig 10 shows the effect of the radius of barrier cycle on the minimum total sensor movement. The radius of barrier cycle varies from 175 to 200 with step 5. The result by the SCC algorithm is smaller than that by the Greedy algorithm, thus SCC algorithm always outperforms the Greedy algorithm. As the radius of barrier cycle increases, the minimum total sensor movement also increases. We find that when the radius of the barrier cycle approaches 190, the gap of the two curves is the largest. Fig 11 shows the effect of the maximum offset of the sinks from the cycle barrier on the minimum total sensor movement. The maximum offset of the sinks from the cycle barrier varies from 10 to 110 with the step 20. The result by the two algorithms both increase as the maximum offset of the sinks from the cycle barrier increases. That's because the sensors need to move a larger distance to cover the barrier cycle.

Fig 12 shows the effect of the number of sinks on the minimum total sensor movement. The number of sinks varies from 2 to 6 with the step 1. The minimum total sensor movement by two algorithms both decreases as the number

of sinks increases. The minimum total sensor movement decreases by 50 percent when the number of sinks increases from 2 to 6. Fig 13 shows the effect of the sensing range of sensors on the minimum total sensor movement. The sensing range of sensors varies from 20 to 28 with 2 as the step size. The minimum sensor movements by the two algorithms both decrease as the sensing range of sensors increases. The result decreases by 30 percent when the sensing range of sensors increases from 20 to 28. The gap between these two curves is the largest when the sensing range of sensors is 22.

VIII. CONCLUSION

In this paper, we study the minimum total sensor movement problem for barrier coverage under sink-based deployment and propose some algorithms both for the line barrier and the cycle barrier. To solve the line barrier case, we enumerate all the projective-subsegments and compute all possible optimal sensor deployments for each projective-subsegment as virtual-clusters. Then we construct a weighted barrier graph and find the path with the minimum weight, which is the minimum total sensor deployment for achieving barrier coverage. We also solve the cycle barrier case by extending the techniques used in the line barrier case. In the future, we will study an approximation algorithm for solving the minimum total sensor movement problem for barrier coverage under the assumption that sensors are randomly distributed in the surveillance area, which has been proved to be NP-hard.

REFERENCES

[1] K. Lembke, L. Kietliński, M. Golański, and R. Schoeneich, "RoboMote: Mobile autonomous hardware platform for wireless Ad-hoc sensor networks," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jun. 2011, pp. 940–944.

[2] K. Michael and M. Michael, "The packbots are coming: Boosting security at the 2014 FIFA world cup," *IEEE Consum. Electron. Mag.*, vol. 3, no. 3, pp. 59–61, 2014.

[3] J. M. Soares, I. Navarro, and A. Martinoli, *The Khepera IV Mobile Robot: Performance Evaluation, Sensory Data and Software Toolbox*. Springer, 2016.

[4] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong barrier coverage of wireless sensor networks," in *Proc. 9th Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2008, pp. 411–420.

[5] Z. Chen, X. Gao, F. Wu, and G. Chen, "A PTAS to minimize mobile sensor movement for target coverage problem," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[6] G. Yang and D. Qiao, "Multi-round sensor deployment for guaranteed barrier coverage," in *Proc. INFOCOM*, Mar. 2010, pp. 1–9.

[7] Y. Wang and G. Cao, "Barrier coverage in camera sensor networks," in *Proc. 12th Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2011, p. 12.

[8] H. Ma, M. Yang, D. Li, Y. Hong, and W. Chen, "Minimum camera barrier coverage in wireless camera sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 217–225.

[9] J. Li, J. Chen, and T. H. Lai, "Energy-efficient intrusion detection with a barrier of probabilistic sensors," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 118–126.

[10] A. Chen, S. Kumar, and T. H. Lai, "Local barrier coverage in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 4, pp. 491–504, Apr. 2010.

[11] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage of line-based deployed wireless sensor networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 127–135.

[12] J. Czyzowicz, E. Kranakis, D. Krizanc, I. Lambadaris, L. Narayanan, J. Opatrny, L. Stacho, J. Urrutia, and M. Yazdani, "On minimizing the maximum sensor movement for barrier coverage of a line segment," in *Ad-Hoc, Mobile Wireless Networks*. Springer, 2009, pp. 194–212.

[13] D. Z. Chen, Y. Gu, J. Li, and H. Wang, "Algorithms on minimizing the maximum sensor movement for barrier coverage of a linear domain," *Discrete Comput. Geometry*, vol. 50, no. 2, pp. 374–408, Jul. 2013.

[14] J. Czyzowicz, E. Kranakis, D. Krizanc, I. Lambadaris, L. Narayanan, J. Opatrny, L. Stacho, J. Urrutia, and M. Yazdani, "On minimizing the sum of sensor movements for barrier coverage of a line segment," in *Ad-Hoc, Mobile Wireless Networks*. Springer, 2010, pp. 29–42.

[15] M. Mehrandish, L. Narayanan, and J. Opatrny, "Minimizing the number of sensors moved on line barriers," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2011, pp. 653–658.

[16] S. Dobrev, S. Durocher, M. Eftekhari, K. Georgiou, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, S. Shende, and J. Urrutia, "Complexity of barrier coverage with relocatable sensors in the plane," in *Algorithms Complexity*. Springer, 2013, pp. 170–182.

[17] S. Li and H. Shen, "Minimizing the maximum sensor movement for barrier coverage in the plane," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 244–252.

[18] A. Saipulla, B. Liu, G. Xing, X. Fu, and J. Wang, "Barrier coverage with sensors of limited mobility," in *Proc. 11th Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2010, pp. 201–210.

[19] B. Bhattacharya, M. Burmester, Y. Hu, E. Kranakis, Q. Shi, and A. Wiese, "Optimal movement of mobile sensors for barrier coverage of a planar region," *Theor. Comput. Sci.*, vol. 410, no. 52, pp. 5515–5528, 2009.

[20] X. Tan and G. Wu, "New algorithms for barrier coverage with mobile sensors," in *Frontiers in Algorithmics*. Springer, 2010, pp. 327–338.

[21] P. Huang, W. Zhu, and L. Guo, "Optimizing movement for maximizing lifetime of mobile sensors for covering targets on a line," *Sensors*, vol. 19, no. 2, p. 273, Jan. 2019. doi: 10.3390/s19020273.

[22] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage with line-based deployed mobile sensors," *Ad Hoc Netw.*, vol. 11, no. 4, pp. 1381–1391, 2013.

[23] Z. Wang, J. Liao, Q. Cao, H. Qi, and Z. Wang, "Achieving k-barrier coverage in hybrid directional sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 7, pp. 1443–1455, 2013.

[24] S. He, J. Chen, X. Li, X. Shen, and Y. Sun, "Mobility and intruder prior information improving the barrier coverage of sparse sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1268–1282, Jun. 2015.

[25] F. Fan, Q. Ji, G. Wu, M. Wang, X. Ye, and Q. Mei, "Dynamic barrier coverage in a wireless sensor network for smart grids," *Sensors*, vol. 19, no. 1, p. 41, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/1/41

[26] J. B. Orlin and A. Sedeno-Noda, "An O(nm) time algorithm for finding the min length directed cycle in a graph," in *Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2017, pp. 1866–1879.

SHUANGJUAN LI received the B.S. and M.S. degrees from Wuhan University, in 2005 and 2007, respectively, and the Ph.D. degree with Sun Yat-sen University, in 2016. She was an Engineer with the Guangzhou Communication Research Institute, from 2007 to 2013. She is currently a Lecturer with South China Agricultural University. Her research interests include wireless sensor networks, ad hoc networks, and optimization algorithm.

HONG SHEN received the B.Eng. degree from the Beijing University of Science and Technology, the M.Eng. degree from the University of Science and Technology of China, and the Ph.Lic. and Ph.D. degrees from Abo Akademi University, Finland, all in computer science. He was a Professor and the Chair of the Computer Networks Laboratory, Japan Advanced Institute of Science and Technology, from 2001 to 2006; he has been a Professor (Chair) of computer science with Griffith University, Australia, where he taught nine years, since 1992. He is currently a specially-appointed Professor with Sun Yat-sen University, China, and a tenured Professor (Chair) of computer science with The University of Adelaide, Australia. He has published more than 300 articles, including more than 100 articles in international journals, such as a variety of the IEEE and ACM TRANSACTIONS. His main research interests include parallel and distributed computing, algorithms, data mining, privacy preserving computing, high performance networks, and multimedia systems. He was a recipient of many honours and awards.

QIONG HUANG received the B.S. and M.S. degrees from Fudan University, in 2003 and 2006, respectively, and the Ph.D. degree from the City University of Hong Kong, in 2010. He is currently a Professor with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. He has published more than 100 research articles in international conferences and journals in the area of cryptography and information security, and served as a program committee member in many international conferences. His research interests include cryptography and information security, in particular, and cryptographic protocols design and analysis.

LONGKUN GUO received the B.S. and Ph.D. degrees in computer science from the University of Science and Technology of China, in 2005 and 2011, respectively. He is currently a Full Professor with the School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences). In 2011, he joined Fuzhou University. From 2015 to 2016, he was a Research Associate with The University of Adelaide. After that, he served as an Associate Professor and Head of the Department of Computer Science, College of Mathematics and Computer Science, Fuzhou University. He has authored over 40 academic articles in reputable journals or conferences, such as Algorithmica, the IEEE TMC, IJCAI, IEEE TPDS, and SPAA. His major research interests include efficient algorithm design, and computational complexity analysis for optimization problems in high-performance computing systems and networks.

Reducing Sensor Failure and Ensuring Scheduling Fairness for Online Charging in Heterogeneous Rechargeable Sensor Networks

Jiaxian Wu[a], Shuangjuan Li[a*], Qiong Huang[a,b]
[a]College of Mathematics and Informatics, South China Agricultural University, China
[b]Guangzhou Key Laboratory of Intelligent Agriculture, South China Agricultural University, China
Email: wujiaxian1998@163.com, lishj2013@hotmail.com, qhuang@scau.edu.cn
[*]*Corresponding author*

***Abstract*—The breakthrough of wireless power transfer technology provides an effective solution to the problem of energy depletion in Wireless Rechargeable Sensor Networks (WRSNs). Most existing work focuses on charging between a mobile charger and a requested sensor, such as NJNP and SAMER, under the assumption that sensors have the same battery capacity and energy consumption rate. In reality, it is more general that a WRSN consists of different types of sensors where they have different battery capacity and energy consumption rate, which is referred as Heterogeneous Wireless Rechargeable Sensor Network (HWRSN). We propose a novel online charging algorithm called VTMT to solve the charging problem in HWRSN. First, we propose the concept of Virtual Time, which is positively correlated with the waiting time of the requested sensor. Then selects the next charging sensor primarily based on the Virtual Time (VT) of the sensor and the Moving Time (MT) of the mobile charger to the node. Simulation results show that VTMT outperforms other charging schemes, which effectively reduce the failure rate of nodes and ensure the scheduling fairness.**

***Index Terms*—Heterogeneous Wireless Rechargeable Sensor Networks, Energy Replenishment, Online Charging Scheme.**

I. Introduction

As one of the most essential technology in Internet of Things, Wireless Sensor Networks (WSNs) are extensively used in precision farming [1], intelligent transportation [2], temperature regulation [3] and forest fire detection [4], etc. Nevertheless, due to the limited battery capacity of sensor nodes in the WSNs, the performance and lifetime of sensor nodes are severely constrained. How to prolong the lifetime of sensors is an important problem. Existing solutions can be roughly divided into three categories: energy conservation [5], energy harvesting [6] and wireless charging [7]. The wireless charging scheme uses a mobile charger to wirelessly charge the sensor node to extend its lifetime. There are three kinds of wireless power transfer technologies used for wireless charging: inductive coupling, electromagnetic radiation and magnetic resonance coupling [7]. Magnetic resonance coupling is widely used in wireless charging because of its high efficiency, long transmission distance, omni direction and insensitive to weather conditions. Adopting this scheme requires the deployment of mobile charging nodes and service station nodes in the network, which is referred as wireless rechargeable sensor networks (WRSNs). Such network with multiple types of sensors is called as heterogeneous wireless rechargeable sensor networks (HWRSNs). The mobile charging node is referred as Mobile Charger (MC) and the service station node is referred as Base Station (BS). Different from other schemes, wireless charging scheme can provide continuous and stable energy for nodes in WRSNs.

The current charging schemes can be roughly divided into two categories: offline [8]–[11] and online [12]–[16]. The offline charging scheme is that the MC periodically charges the sensor nodes in accordance with a predefined route. For the existing offline schemes, it is usually assumed that the energy consumption rate of the nodes is constant, which is inconsistent with the dynamic energy consumption rate in the practical applications. For online charging schemes, nodes can periodically send the residual energy message to the BS, so that the node consumption rate can be estimated. Due to the dynamic energy consumption rate of sensors, the online charging scheme cannot plan the charging paths of MC in advance and should schedule the charging orders of sensors in real time, which is more according with the practical applications.

There are some challenges in designing the online charging scheme: First, whether the charging scheme can reduce the node failure rate in the network, which is defined as the percentage of the sensors with energy depletion. It is a key factor in evaluating a online charging scheme. Second, whether the charging scheme can reduce the moving cost of the MC, which is defined as the total moving distance of the MC. Third, whether the charging scheme can guarantee scheduling fairness, which means that the requested sensors cannot wait for charging for a long time. The classic NJNP scheduling strategy [13] always chooses the nearest node to the MC as the next charging node, and new requests can be preempted during the MC's movement. Therefore, this scheme leads to a high node failure rate. Meanwhile, this scheme has a bad performance in HWRSNs.

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

We propose a novel online charging algorithm in HWRSNs, called a charging algorithm based on virtual time and moving time (VTMT). First, we define three concepts, virtual time, moving time and starvation node rate. Virtual time (VT) is positively related to the waiting time and the energy consumption rate of the requested node. Moving time (MT) is defined as the time of the MC moved to the node. We also define the concept of starvation node rate, which used to judge the scheduling fairness. Second, we investigate a simple and effective method of calculating the energy consumption rate of node based on historical information. The main idea of VTMT algorithm is to divide the VT by the MT as the Service Value (SV), and then select the node with the highest Service Value (SV), which is the ratio of VT to the MT, as the next charging node. The requested sensor with a longer virtual time are more likely to be charged preferentially. The nearer candidate node are chosen to reduce the moving cost of the MC. Experiments demonstrate that our algorithm results in the low node failure rate, short moving cost of MC and good scheduling fairness in HWRSNs.

The rest of the paper is arranged as follows: We introduce related works in Section II. The system model and notions will be given in Section III. In Section VI, we present the the VTMT algorithm. Simulation experiments are performed in Section V. Section VI concludes this paper.

II. Related Work

With the development of wireless rechargeable sensor networks, two types of charging schemes are proposed, offline charging schemes and online charging schemes.

For offline schemes, Fu et al. [8] transformed the charging delay minimization problem into a solvable linear programming problem, and proposed a heuristic algorithm to further reduce the computational complexity. Li et al. [9] put forward a joint routing and charging scheme to maximize the lifetime of HWRSNs. Xie et al. [10] used the piecewise linear approximation technique to find an approximation algorithm to maximize the proportion of the MC's vacation time over the cycle time. Peng et al. [11] transformed the energy replenishment problem into a classic traveling salesman problem and proposed two greedy algorithms to extend the lifetime of the sensor network. However, these offline schemes did not consider the real-time dynamic energy consumption rate of sensors, which are not applicable to many real-time applications.

The online schemes can handle the dynamic energy consumption rate of sensors. For the existing online charging schemes, He et al. proposed a first-come-first-serve (FCFS) scheduling strategy [12], which is simple but performs poor performance. It only considered the arrival time of requests, but did not consider the distance between the node and the MC, which led to the high moving cost of the MC and high node failure rate. In order to reduce the moving cost of the MC, He et al. proposed an efficient scheduling strategy called NJNP [13], which is based on the principle of nearest job next with preemption. This scheduling strategy can reduce the node failure rate and MC moving costs, but lack the scheduling fairness. In the work [14], Tomar et al. proposed an algorithm that multiplies the tolerable time of node by the distance of MC to node as the cost, and then selects the node with the lowest cost as the next node. The complexity of the algorithm was improved by the heap structure. Compared with NJNP, this algorithm can reduce the node failure rate and the average charging latency. In order to ensure the fairness of the scheduling and reduce the node failure rate, Feng et al. proposed a Starvation Avoidance Mobile Energy Replenishment scheme (SAMER) [15], which can avoid energy starvation through calculating and considering the maximum tolerable latency of each charging requirement. This scheme abandons the failed nodes. On the basis of SAMER, Zhu et al. proposed an Invalid Node Minimized Algorithm (INMA) [16]. INMA preferentially selects the node that causes the least number of other nodes to fail after charging it. Experiments show that compared with SAMER, the algorithm can reduce the failure rate of nodes under certain conditions.

III. System Model and Notions

A. System Model

We define heterogeneous wireless rechargeable sensor networks as a triple (N, BS, MC), as shown in Fig. 1. N is the set of sensor nodes in HWRSNs, where N=$\{N_1, N_2, ..., N_n\}$. The battery capacity of N_i is E_i, and the energy consumption rate of N_i is P_i. Sensors are randomly distributed in a square with side length M, and the Euclidean distance between N_i and N_j is denoted by D_{ij}. Sensor nodes can communicate with BS, where the time of communication and the energy consumed by communication can be ignored. BS is the only base station which has enough energy and the ability to calculate and store, and can communicate with MC. It can recharge the MC or replace its battery in negligible time. MC is the only wireless mobile charger in HWRSNs, whose battery capacity is E_{MC} and moving speed is υ. Let P_{MC} denote the energy consumed by the MC per unit distance of movement, and η is the charging rate between MC and the node. The threshold of MC is denoted as χ. If its residual energy is lower than χ, MC will return to the BS to supplement the energy. Except for the energy consumed by charging and moving, other energy consumption of MC is negligible. It can communicate with the BS, and the time and energy consumed by the communication can be ignored.

The system works as follows: In HWRSNs, the node N_i, i$\in\{1, 2, ..., n\}$, sends a residual energy message (ID, RE, T, NC) to the BS every Δ time, and the BS will calculate the node's dynamic energy consumption rate PD_i and determine whether the node needs to be charged based on this message. Among them, ID is the serial number of the node, RE is the residual energy of the node, where RE$\geq$0, T is the timestamp of the node sending message, NC indicates whether the node needs to be charged, and NC$\in\{$TRUE, FALSE$\}$. The BS maintains a Dynamic Energy Consumption Rate Pool (DECRP), which records the latest dynamic energy consumption rate PD_i of the node. The BS maintains a

Residual Energy Message Pool (REMP), which records the lastest residual energy message of each node. The BS will maintain a Charging Request Pool (CRP) which records the first charge request of each node, that is, the first NC is TRUE in the residual energy message. When the node sends the residual energy message, it will detect its own residual energy. If the residual energy is lower than the threshold ϕ_i, the node will set NC to TRUE, indicating that the node needs to be charged, otherwise the NC is set to FALSE by default.

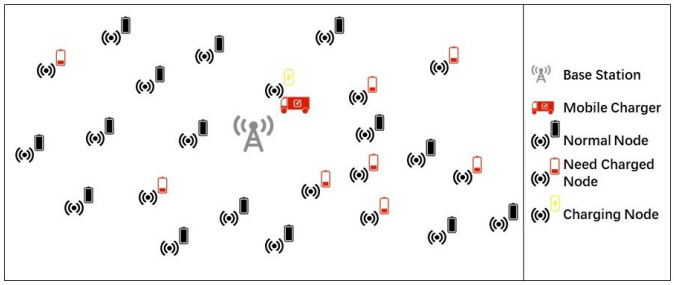


Fig. 1. Network Model

During MC charging, the charged node will suspend sending the residual energy message. When the MC completes the charging, MC will send a completed service message (ID) to the BS to inform that the charging service is completed, and the ID is the serial number of the completed charging node. At this time, the BS will remove the records about the node in DECRP, REMP and CRP to avoid the interference of outdated messages on subsequent scheduling. After the MC completes charging of the node, it will send a request (RE_{MC}, L) to the BS to query the next service location. RE_{MC} denotes the residual energy of the MC. L denotes the location of the MC, and $\mathrm{L}\in\{\mathrm{L}_0, \mathrm{L}_1, ..., \mathrm{L}_n\}$. L_0 is the location of BS and the location of N_i is L_i. After BS calculates MC's next service location L using our online charging algorithm, it will send a response (L) to the MC. L denotes the next service location of the MC. And then the MC will go to the next location for service.

The MC does not allow other sensor nodes to preempt during the charging process to avoid interruption of the charging process. Table I shows the symbols used in this paper.

B. Notions

Before showing the online charging scheme, we first give some definitions.

Definition 1. (Node Starvation) If the failure time of a sensor is greater than τ time, this sensor is said to be in the node starvation state.

If a sensor node is unable to be selected as the next charging node for a long time, resulting in the node being in a node starvation state. Node starvation rate is an important measure indicating whether the charging scheme is fair. The lower the starvation rate of the nodes, the fairer the charging scheme is. The formula for judging whether the node is in the node starvation state is as follows:

TABLE I
LIST OF NOTATIONS.

Symbol	Description
M	Side length of network
P_{MC}	Energy consumed by travelling one unit distance of MC
υ	Moving velocity of MC
η	Charging rate between MC and the node
RE_{MC}	Residual energy of MC
E_{MC}	Battery capacity of MC
χ	Threshold of MC return BS supplementary energy
RE_i	Residual energy of N_i
E_i	Battery capacity of N_i
P_i	Energy consumption rate of N_i
PD_i	Dynamic energy consumption rate of N_i
ϕ_i	Threshold of N_i send charging request
α	Update rate of dynamic energy consumption rate
Δ	Sensor node sends the residual energy message interval
L_i	The location of the BS or sensor node in the network
D_{ij}	Euclidean distance between L_i and L_j
TC	Current timestamp

$$TC - TF \geq \tau \tag{1}$$

Where TF denotes the timestamp of node running out of energy, and τ is a given constant . When τ=0, this formula can be used to judge whether the node fails.

Definition 2. (Virtual Time) The virtual Time of sensor N_i, denoted as VT_i, is defined to be a measurement used to select the next charging node, which is calculated as follows:

$$VT_i = \frac{(TC - T_i)PD_i}{\overline{PD}} \tag{2}$$

Where T_i denotes the timestamp of the node's first charging request, and $\overline{PD}$ is the average dynamic energy consumption rate of all nodes that need to be charged.

When calculating the virtual time, we consider the waiting time of the nodes to ensure fairness. The node with the longer waiting time will be more likely to be selected as the next charging node. Besides, we also consider the energy consumption rates of nodes, since the energy consumption rates of different nodes differ greatly, as shown in Eq. 2. The higher the energy consumption rate, the longer the virtual time.

It can be considered that the higher the energy consumption rate, the faster the virtual time increases with the waiting time.

Definition 3. (Moving Time) The moving time of sensor N_i, denoted as MT_i, is the time cost by MC moving to this sensor, which is calculated as follows:

$$MT_i = \frac{D_{ji}}{\upsilon} \tag{3}$$

Where D_{ji} denotes the distance of the MC from the current location L_j to the next node N_i's location L_i.

The moving distance of the MC is an important factor in scheduling, because reducing the moving distance of the MC not only helps to reduce the energy consumption of the

MC, but also indirectly reduces the node failure rate and the starvation rate. Therefore, in our algorithm, the node closer to MC will be more likely to be selected as the next charging node. In addition, as shown in Eq. 3, we convert the distance of MC to node into the moving time of MC.

IV. System Working Process and VTMT Algorithm

A. System Working Process

In this subsection, BS will use the following method to calculate the dynamic energy consumption rate, denoted as PD, of the node once it receives the residual energy message not for the first time. The dynamic energy consumption rate PD of the node is calculated based on the historical information.

Before calculating PD, we first define the current energy consumption rate, denoted as PR_i of the N_i as follows:

$$PR_i = \frac{RE_{io} - RE_{in}}{\Delta} \tag{4}$$

Where RE_{io} denotes the residual energy in the last residual energy message of the node (o represents old), RE_{in} denotes the residual energy of the current residual energy message (n represents new). We do not calculate PR, when the BS receives the first residual energy message.

We define the dynamic energy consumption rate PD_i as follows:

$$PD_i = \begin{cases} (1-\alpha)PD_i + \alpha PR_i, & PD_i \text{ exists} \\ PR_i, & PD_i \text{ not exists} \end{cases} \tag{5}$$

Where α is parameter. The larger the α is, the faster the dynamic energy consumption rate is more updated. When α=1, PD_i=PR_i, that is, the real-time energy consumption rate to the node is taken as the dynamic energy consumption rate.

When the BS receives the MC's request (RE_{MC}, L) for the next destination, the BS will calculate the next destination L_{next} of the MC according to the VTMT algorithm. If L_{next} is the location of the BS, then the MC will return to the BS. If it is the location of the node, then the MC will go to the next node to charge it.

B. VTMT Algorithm

In this subsection, we propose the VTMT algorithm, which is used by BS to select the next charging node of the MC. The VTMT algorithm works as follows:

1) step 1: First, determine whether the CRP is empty. If it is empty, return L_0. Otherwise, go to step 2.

2) step 2: For all nodes in the CRP, calculate the moving time of MC to node MT and virtual time VT, and normalize the MT and VT respectively to be MT^* and VT^*. Then divide VT^* by MT^* as the service value (SV) of the node and find out the node N_i with the largest service value among all requested nodes as the next charging node. Go to step 3.

SV_i is calculated as follows:

$$SV_i = \frac{VT_i^*}{MT_i^*} \tag{6}$$

3) step 3: Determine whether the MC has enough energy to return to the BS after charging the node N_i, that is, whether it meets Eq. 7. If it is satisfied, then return L_i. Otherwise, return L_0.

The formula for judging whether the MC can serve this node is as follows:

$$RE_{MC} - D_{ji} \times P_{MC} - TC_i \times \eta \geq \chi \tag{7}$$

Where D_{ji} denotes the distance between current location L_j to L_i, TC_i is charging time for MC to charge node N_i.

TC_i is defined as follows:

$$TC_i = \begin{cases} \frac{E_i - ERE_i}{\eta - PD_i}, \text{ERE}_i > 0 \\ \frac{E_i}{\eta - PD_i}, \text{ERE}_i \leq 0 \end{cases} \tag{8}$$

Where ERE_i=RE_i-(TC-T_i)×PD_i-MT_i×PD_i, which denotes the estimated residual energy. RE_i denotes the residual energy in the last residual energy message of N_i, T_i denotes the timestamp of the last residual energy message of N_i, and MT_i denotes the moving time of MC from current location L_j to L_i. Since the nodes also need to consume energy during charging, the actual charging rate is η-PD_i. If the energy of the node is exhausted when the MC reaches the node, that is, $ERE_i \leq 0$, then $\frac{E_i}{\eta - PD_i}$ is used to calculate the charging time of the node.

The pseudo code of the VTMT algorithm is given in Algorithm 1.

Algorithm 1 VTMT Algorithm

Input: $L_{MC}, DECRP, CRP$
Output: L

```
if CRP is empty then
    return L_0
end if
N = getAllNeedChargingNodes(CRP)
PD̄ = calculateAveragePD(N, DECRP)
VT = φ, MT = φ, SV = φ
for N_i in N do
    VT = (TC − T_i) * PD_i / PD̄
    MT = D_ij / v
end for
VT* = normalize(VT), MT* = normalize(MT)
for N_i in N do
    SV = VT_i* / MT_i*
end for
N_next = getMaxServiceValueNode(SV)
if N_next satisfies Eq. 7 then
    return N_next.getLocation()
end if
return L_0
```

V. Performance Evaluation

In this section, we evaluate the performance of our proposed VTMT algorithm under different network conditions. We analyze the experimental results by comparing with the state-of-the-art charging schemes NJNP and SAMER. The metrics

TABLE II
DEFAULT PARAMETERS.

Parameter	Value
M	100m
Number of Node	150
υ	1m/s
P_{MC}	4mJ/m
η	100mJ/s
E_{MC}	5,000,000mJ
χ	1,000mJ
E_i	Randomly from 1000 to 3,000mJ
ϕ_i	$0.4E_i$
Δ	10s
α	0.25
P_i	Randomly from 0.1 to 1.0mJ/s
τ	1,000s
Simulation time	86,400

used for comparison are node starvation rate, node failure rate and moving cost.

A. Experimental Environment Settings

The experiment is performed using Java. 150 nodes are randomly deployed in a 100m$\times$100m area, and the simulation time is 24 hours. The battery capacity of each node is randomly drawn from 1000 to 3,000mJ and energy consumption rate of each node is randomly drawn from 0.1 to 1.0mJ/s. The value in each figure is the mean of 1000 experiments. The experimental parameters are given in Table II.

B. Different Number of Nodes

As shown in Fig. 2, the number of sensors varies from 25 to 300. As shown in Fig. 2(a) and Fig. 2(b), it is found that as the number of nodes increases, the node failure rate and the starvation rate also gradually increase. This is because as the number of nodes increases, the MC cannot serve each node in a timely manner. However, the node failure rate and the starvation rate are obviously lower than NJNP and SAMER, which implies that VTMT is more suitable for HWRSNs. From Fig. 2(b), we can see that VTMT can reduce the starvation rate of nodes, which shows that VTMT is fairer than the other two algorithms. In Fig. 2(c), as the number of nodes increases, the moving cost increases first and then decreases. That is because as the number of nodes increases, the MC needs to serve more nodes and move longer. When the number of nodes grows to a threshold, the MC cannot provide charging services for all nodes in a timely manner. The average distance between nodes decreases, and then some nodes closer to the MC will be selected, resulting in smaller moving cost.

C. Different Charging Rate

As shown in Fig. 3, the charging rate varies from 100 to 300mJ/s. It can be seen from Fig. 3(a) and Fig. 3(b) that as the charging rate increases, the node failure rate and the starvation rate gradually decrease. As the charging rate increases, MC can charge the nodes more quickly, thus reducing the charging time and serving more nodes. And we can see that under different charging rates, both the failure rate and the starvation rate of VTMT are significantly lower than NJNP and SAMER. As shown in Fig. 3(c), as the charging rate increases, the moving cost of all charging schemes also gradually increase. Since the MC can charge more nodes as the charging rate increases, it needs to move longer.

D. Different Velocity of MC

As shown in Fig. 4, the moving speed of MC varies from 0.6 to 1.5m/s. It can be viewed from Fig. 4(a) and Fig. 4(b) that the node failure rate and the node starvation rate decrease with the increase of MC moving velocity. Since the increasing moving speed can reduce MC's moving time, more time is spent on charging nodes. Compared with the other two charging schemes, VTMT results in the lower node failure rate and the lower starvation rate. As shown in Fig. 4(c), as the moving speed of the MC increases, the moving cost of the MC also increases gradually. This is because as the moving speed of the MC increases, the moving time of the MC decreases, and the MC has more time to charge more nodes. Therefore, it needs to move longer.

VI. CONCLUSION

In this paper, we proposed an online charging algorithm called VTMT, which was suitable for HWRSNs. We also calculated the dynamic energy consumption rate based on historical information, which can give a very good approximation to the energy consumption rate of the nodes. Experiments showed that in HWRSNs, compared with the state-of-the-art NJNP and SAMER, our proposed VTMT algorithm could reduce the failure rate of nodes and ensure the fairness of scheduling. In the future, we will study an one-to-many online charging algorithm for HWRSNs, that is a mobile charger can charge more than one sensor simultaneously.

VII. ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (Grant No. 61702198), Science and Technology Program of Guangzhou (No. 201902010081), Guangdong Major Project of Basic and Applied Basic Research (No. 2019B030302008), National Natural Science Foundation of China (No. 61872152)

REFERENCES

[1] J. Xia, Z. Tang, X. Shi, L. Fan and H. Li.: An environment monitoring system for precise agriculture based on wireless sensor networks. In: 2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks, pp. 28–35. IEEE, Beijing (2011)

[2] Kafi MA, Challal Y, Djenouri D, Doudou M, Bouabdallah A, Badache N.: A study of wireless sensor networks for urban traffic monitoring: Applications and architectures. In: Procedia Computer Science 19, pp. 617–626. Elsevier, Canada (2013)

[3] Risteska Stojkoska, B., Popovska Avramova, A.: Application of wireless sensor networks for indoor temperature regulation. International Journal of Distributed Sensor Networks, 1–10 (2014)

[4] Yu L., Wang N., Meng X.: Real-time forest fire detection with wireless sensor networks. International Conference on Wireless Communications, Networking and Mobile Computing. 1214–1217 (2005)

[5] Bin W., Wenxin L., Liu L.: A survey of energy conservation, Routing and coverage in wireless sensor networks. In: Active Media Technology - 7th International Conference. 59–70. Springer, Berlin (2011)

[6] Harb A.: Energy harvesting: State-of-the-art. Renewable Energy **36**(10), 2641–2654 (2011)

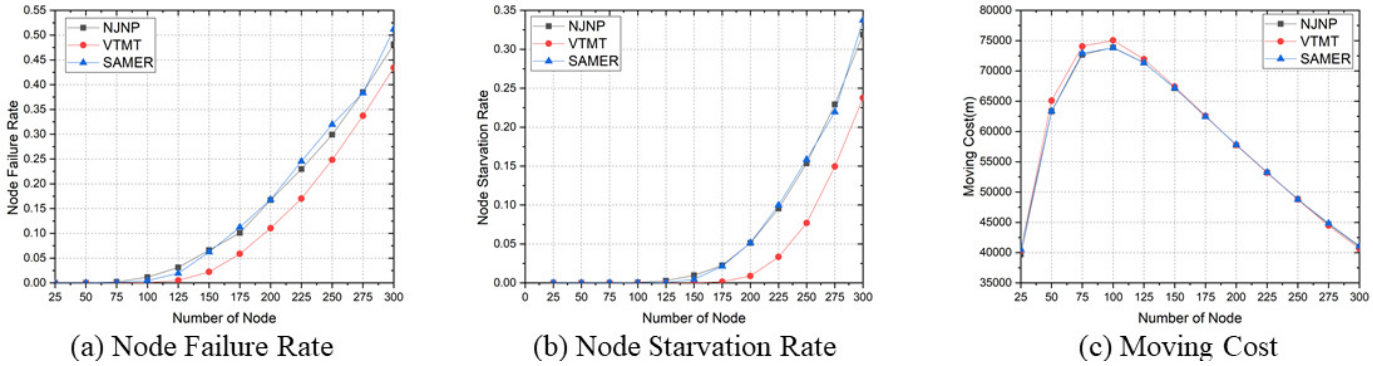


Fig. 2. Experimental results under different number of nodes.

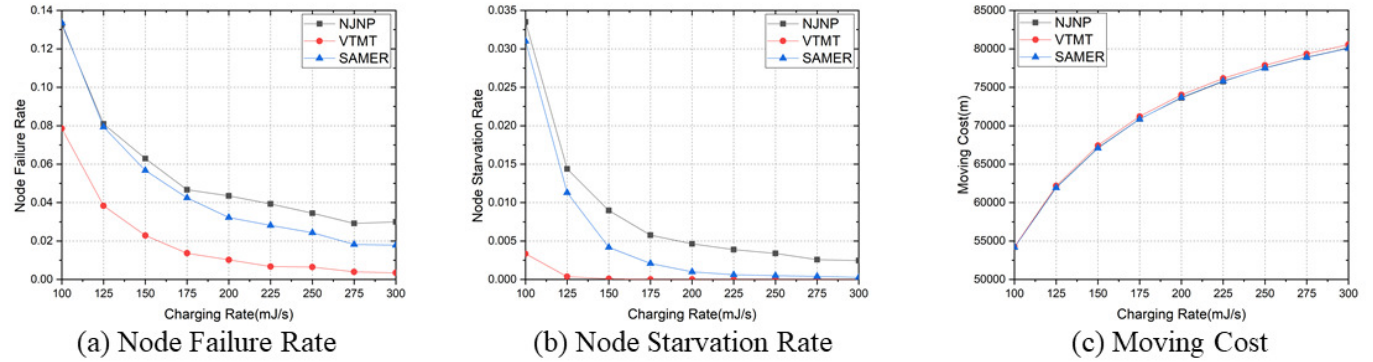


Fig. 3. Experimental results under different charging rate.

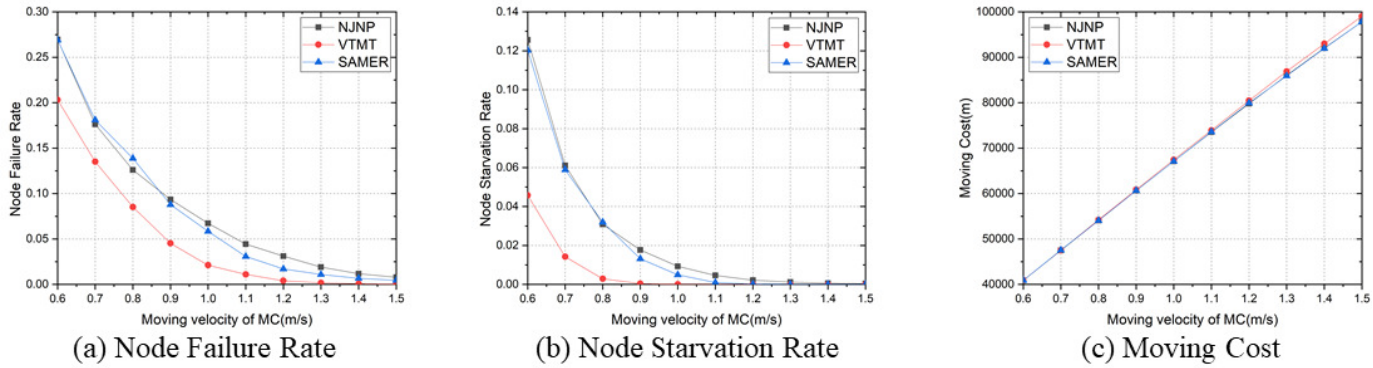


Fig. 4. Experimental results under different velocity of MC.

[7] L. Xie, Y. Shi, Y. T. Hou and A. Lou: Wireless power transfer and applications to sensor networks. IEEE Wireless Communications **20**(4), 140–145 (2013)

[8] L. Fu, P. Cheng, Y. Gu, J. Chen and T. He: Optimal charging in wireless rechargeable sensor networks. IEEE Transactions on Vehicular Technology **65**(1), 278–291 (2016)

[9] Z. Li, Y. Peng, W. Zhang and D. Qiao: J-RoC: A joint routing and charging scheme to prolong sensor network lifetime. In: 2011 19th IEEE International Conference on Network Protocols, pp. 373–382 (2011)

[10] L. Xie, Y. Shi, Y. T. Hou and H. D. Sherali: Making sensor networks immortal: An energy-renewal approach with wireless power transfer. IEEE/ACM Transactions on Networking **20**(6), 1748–1761 (2012)

[11] Y. Peng, Z. Li, W. Zhang and D. Qiao: Prolonging sensor network lifetime through wireless charging. In: 31st IEEE Real-Time Systems Symposium, pp. 129–139. IEEE, San Diego (2010)

[12] L. He, Y. Zhuang, J. Pan and J. Xu: Evaluating on-demand data collection with mobile elements in wireless sensor networks. In: 2010 IEEE 72nd Vehicular Technology Conference - Fall, pp. 36–39 (2010)

[13] L. He, L. Kong, Y. Gu, J. Pan and T. Zhu.: Evaluating the on-demand mobile charging in wireless sensor networks. IEEE Transactions on Mobile Computing **14**(9), 1861–1875 (2015)

[14] A. Tomar, R. Anwit and P. K. Jana: An efficient scheme for on-demand energy replenishment in wireless rechargeable sensor networks. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 125–130. IEEE, Udupi (2017)

[15] Y. Feng, N. Liu, F. Wang, Q. Qian and X. Li: Starvation avoidance mobile energy replenishment for wireless rechargeable sensor networks. In: 2016 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE, Kuala Lumpur (2016)

[16] J. Zhu, Y. Feng, M. Liu, G. Chen, Y. Huang.: Adaptive online mobile charging for node failure avoidance in wireless rechargeable sensor networks. Computer Communications, 28–37 (2018)

Exact algorithms for barrier coverage with line-based deployed rotatable directional sensors

Zijing Ma[a], Shuangjuan Li[a*], Dong Huang[a]
[a]College of Mathematics and Informatics, South China Agricultural University, China
Email: mazijingscau@hotmail.com, lishj2013@hotmail.com, huangdong06@163.com
[*]*Corresponding author*

***Abstract*—Barrier coverage is an important coverage model for intrusion detection, which requires a chain of sensors across the deployment area with the adjacent sensors' sensing areas overlapping. Directional sensors are often dispersed from an airplane following a predetermined line. However, barrier coverage cannot be guaranteed after initial sensor deployment due to the sensors' random offsets and random orientations. Fortunately, directional sensors can rotate to mend the barrier gaps using this line-based sensor deployment model. Existing work proposed a greedy heuristic approach to mend the gaps by rotating sensors, but it cannot answer whether there exists a barrier. We fill in this gap by presenting an exact algorithm which can determine whether there exists a barrier. We first introduce the notion of feasible orientation range and then try to calculate each sensor's feasible orientation starting from the leftmost sensor. We also propose a fast algorithm of choosing the sensors' orientations from their feasible orientation ranges to form a barrier if there exists a barrier, or form a set of sub-barriers if there does not exist a barrier. Simulation results show that our algorithm outperforms the distributed algorithm in the existing work.**

***Index Terms*—wireless sensor networks, directional sensor, strong barrier coverage, rotatable sensor**

I. Introduction

Wireless Sensor Networks (WSNs) have been widely used in many fields such as intrusion detection, border surveillance, critical resource protection and battlefield surveillance. Barrier coverage is an important coverage model of intrusion detection, which aims to form a barrier consisting of sensors across the region of interest (ROI) such that any intruder passing through the ROI can be detected by at least one sensor [1]. Sensors are often dispersed from an airplane following a predetermined line in the ROI, where the offset of each sensor's actual landing location follows a normal distribution. It is important to study the barrier coverage problem of such line-based sensor deployment model([2], [4]–[6], [9]), which is more realistic than the poisson distribution model [12]. Due to the random offsets of the sensors' locations, it is difficult to guarantee barrier coverage after initial line-based sensor deployment and the barrier gaps are unavoidable.

Most of the research on barrier coverage are based on isotropic sensor whose sensing range is modeled as a circle [4], which is an ideal model. However, sensors with directional sensing range, often modeled as a sector [3], are widely used in many practical applications, such as cameras, audio sensors, infrared sensors and radar sensors. It is more practical to study how to achieve barrier coverage using directional sensors than the isotropic sensors. Due to the sensors' random offsets of locations and random orientations, the orientation of these sensors is also a key factor to form barrier coverage besides the positions of the directional sensors. There may exist some barrier gaps after initial line-based sensor deployment, which can be mended by changing the orientations of the sensors. However, it is challenging to determine the orientations of the directional sensors to mend these gaps, since the sensors can rotate 360 degrees.

Existing work in [5] proposed a distributed algorithm of rotating the sensors, called Distributed Gap Mending Algorithm, to mend the barrier gaps. However, this algorithm cannot give an exact answer to the problem of determining whether there exists a barrier. Distributed Gap Mending Algorithm used a greedy method, which determined the orientation of each sensor only using the information of its closest neighbor sensors. It always chose the orientation of each sensor such that its sensing area is the nearest to its right closest neighbor sensor. However, this chosen orientation may not be the optimal choice for deciding the orientations of other right neighbor sensors.

In this paper, we fill in this gap by proposing an exact algorithm to solve the problem of determining whether there exists a barrier. We first introduced a notion of feasible orientation range. That is, if an orientation of one sensor is not in its feasible orientation range, this sensor cannot form a barrier with other sensors with any possible orientations. Then we calculate each sensor's feasible orientation range starting from the leftmost sensor. If there exists one sensor whose feasible orientation range is null, it implies that there does not exist a barrier; otherwise, there exists a barrier and we propose an algorithm to find the orientations of all the sensors to form a barrier. Even if there does not exist a barrier, we can also find the orientations of all the sensors to form a set of sub-barriers such that the total number of gaps is minimized.

The rest of the paper is organized as follows. Section II reviews some related works. In section III we will establish the network model. Section IV proposes an algorithm of determining whether there exists a barrier. In section V we propose an algorithm of finding the sensors' orientations to form a barrier or a set of sub-barriers. In section VI we evaluate the performance of the algorithms by simulations. In section VII we conclude this paper.

978-1-7281-3106-1/20/$31.00 ©2020 IEEE

II. RELATED WORK

Barrier coverage is a hot topic in WSNs. It aims to construct a chain of sensors across the deployed area to detect any intruder crossing through it. The notion of barrier coverage was first proposed in the work [1], in which two types of barrier coverage were studied: weak barrier coverage and strong barrier coverage. Weak barrier coverage aims to detect any intruder crossing the ROI along the vertical paths, while strong barrier coverage aims to detect any intruder whose crossing paths are arbitrary. In this paper we study the strong barrier coverage, short for barrier coverage. The work in [6]–[10] studied how to achieve barrier coverage with sensors. However, most of these works assume that the sensing model of sensors are omnidirectional.

The work in [12] studied the barrier coverage problem in directional sensor network under the poisson distribution model. It defined the virtual node to reduce the solution space from continuous domain to discrete domain and then constructed a barrier graph using these virtual nodes as vertices. By finding a path in this graph, it could determine whether there exist sensors' orientations that can provide barrier coverage. However, the algorithm could not give us an exact answer but an approximate one, because the solution space is continuous other than discrete.

The work in [2] studied how to mend the barrier gaps by rotating the directional sensors for barrier coverage under the line-based deployment. It proposed algorithms for the weak barrier coverage and strong barrier coverage. The simple rotation algorithm was proposed to only rotate the sensor on either side of the gap to mend this gap. When this simple algorithm could not mend the gap, the chain reaction-based rotation algorithm was proposed to rotate the sensors one by one like a chain reaction to fix the gap. The work in [5] proposed distributed algorithms by rotating sensors to mend the gap based on line-based deployment. It determined the orientation of one sensor based on the closest left and right neighbor sensors. However, both of the algorithms in [2], [5] cannot answer whether there exists a barrier. We study the same scenario as the work in [2], [5], and try to fill in this gap by giving an exact answer to this problem.

III. NETWORK MODEL

The ROI is a rectangular belt region of length L and width H, where $L \gg H$. N directional sensors are deployed evenly on a predetermined horizontal line in ROI, where the X coordinate of sensor s_i's target location is calculated by $(2i-1)/2N$. However, due to environmental constraints, the actual locations of sensors have random offsets.

As shown in Fig. 1, a directional sensor s_i is modeled as a sector, denoted as $< (x_i, y_i), \theta, \varphi, \mathrm{R}_s >$. Here, (x_i, y_i) are the cartesian coordinates of sensor s_i, θ is the view angle, φ is the orientation angle and R_s is the sensing range. The sector of sensor s_i' view area is denoted as S_i. For any point p on the arc of S_i, we rotate $\overrightarrow{s_i p}$ around s_i in a clockwise direction until we meet the first endpoint of this arc denoted as m_i and the other endpoint denoted as n_i. The vector $\overrightarrow{s_i m_i}$ is called the starting edge of sector S_i while $\overrightarrow{s_i n_i}$ is called the end edge of S_i. Sensor s_i can rotate to change its orientation angle and can rotate from 0 to 360 degree. Assume that each directional sensor knows its coordinate (x_i, y_i) using GPS or other localization algorithms. Fig. 2 shows an initial line-based deployment of directional sensors, which has some barrier gaps, denoted by $G_1, G_2, ...G_9$.

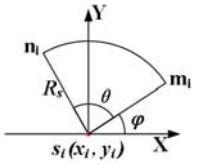


Fig. 1. Directional sensing model

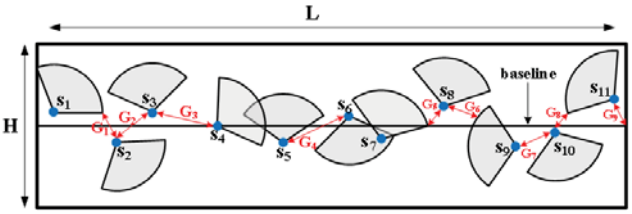


Fig. 2. Initial line-based sensor deployment with gaps $G_1, G_2, ...G_9$

We study how to rotate the directional sensors to achieve barrier coverage. We first define the problem of how to determine whether there exists a barrier as follows:

Definition 1. The decision problem of barrier coverage (DBC): Given n directional sensors deployed along a predetermined horizontal line with random offsets, the problem is to determine whether there exists an orientation angle of each sensor such that the sensing ranges of the adjacent sensors overlap with each other to form a barrier crossing from the left boundary of the region to the right boundary.

We also define the problem of how to choose the orientation angles of these sensors to form a barrier or a set of sub-barriers as follows:

Definition 2. The orientation problem of barrier coverage(OBC): Given n directional sensors deployed along a predetermined horizontal line with random offsets, the problem is to find the orientation angle of each sensor such that the sensing ranges of these sensors overlap to form a barrier crossing from the left boundary of the region to the right boundary or a set of sub-barriers such that the total numbers of gaps is minimized.

IV. MOTIVATION

In this section we analyze the Distributed Gap Mending Algorithm [5] and show that this algorithm may not form a barrier even if there exists a barrier. This algorithm chose the orientation of each sensor such that its sensing area is

the nearest to its closest right neighbor sensor, which is local optimal.

Given the orientation of sensor s_{i-1}, this algorithm always chose the orientation of sensor s_i such that the corresponding sector is the closest to the vector $\overrightarrow{s_i s_{i+1}}$. For example, as shown in Fig. 3(a), S_i rotates to overlap with S_{i-1}. If S_i rotates anti-clockwise to touch S_{i-1}, a_i is the point where S_i intersects with S_{i-1}. If S_i rotates clockwise to touch S_{i-1}, b_i is the point where S_i intersects with S_{i-1}. S_i finally rotates clockwise to touch $\overrightarrow{s_i b_i}$, since $\overrightarrow{s_i b_i}$ is closer to $\overrightarrow{s_i s_{i+1}}$ than $\overrightarrow{s_i a_i}$, as shown in Fig. 3 (b). Similarly, we rotate S_{i+1} to touch $\overrightarrow{s_{i+1} b_{i+1}}$, since $\overrightarrow{s_{i+1} b_{i+1}}$ is closer to $\overrightarrow{s_{i+1} s_{i+2}}$ than $\overrightarrow{s_{i+1} a_{i+1}}$, as shown in Fig. 3 (c). However, S_{i+2} cannot intersect with S_{i+1}, since S_{i+1} cannot intersect with the circle which S_{i+2} is located at.

In fact, as shown in Fig. 3 (d), S_{i+2} can rotate to touch S_{i+1}, if S_i rotates to touch $\overrightarrow{s_i a_i}$ other than $\overrightarrow{s_i b_i}$. Thus, Distributed Gap Mending Algorithm may not find the proper orientations of sensors to form a barrier if there exists one.

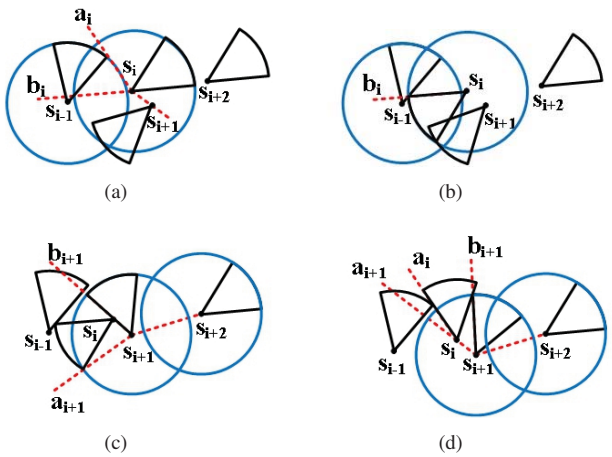


Fig. 3. A failure case of Distributed Gap Mending Algorithm [5]

V. DBC Algorithm

In this section, we will propose an algorithm for the decision problem of barrier coverage, short for DBC algorithm. To determine whether there exists a barrier, we try to calculate each sensor's feasible orientation range.

A. calculate the feasible orientation range

We first introduce some notions.

Definition 3. Virtual circle: Virtual circle, denoted as C_i, is the circle where the sector S_i is located at.

As shown in Fig. 4(a), sensor s_i's sensing area is represented by the sector bounded by black edges and its virtual circle is the circle in blue.

Definition 4. Sub-barrier: A sub-barrier is formed by a set of sensors if the sensing ranges of the adjacent sensors overlap with each other.

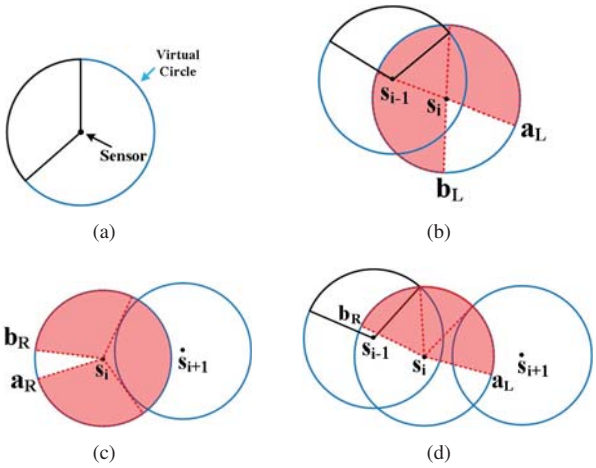


Fig. 4. **(a)** virtual circle; **(b)** feasible orientation range; **(c)** right orientation range; **(d)** updated feasible orientation range

If a sub-barrier also overlaps with the left and right boundary of the ROI, this sub-barrier is called a barrier.

Definition 5. Feasible orientation: For sensor s_i, one orientation angle is called the feasible orientation if the sensor can form a sub-barrier together with its left neighbor sensors.

It is easy to know that there may be many feasible orientations. Thus, we define the feasible orientation range as follows:

Definition 6. Feasible orientation range: For sensor s_i, its feasible orientation range, denoted as O_i, includes all possible feasible orientations.

We also define the feasible region.

Definition 7. Feasible region: For sensor s_i, its feasible region, denoted as T_i, is the union of the sector S_i whose orientation is within its feasible orientation range.

Let $\alpha\left(\overrightarrow{f}\right)$ denote the included angle of vector $\overrightarrow{f}$ and the x-axis. The feasible orientation range O_i and feasible region T_i are calculated shown in Fig. 4 (b). Suppose the feasible region T_{i-1} of sensor s_{i-1} is the sector with black edges. The sector S_i first rotates such that it does not intersect with T_{i-1}. Then rotate S_i around s_i anticlockwise until it first meets T_{i-1}, denoting its starting edge as $\overrightarrow{s_i a_L}$. Continue rotating S_i around s_i anticlockwise until it first leaves T_{i-1}, denoting its end edge as $\overrightarrow{s_i b_L}$. Thus, the feasible orientation range O_i is the range from $\alpha\left(\overrightarrow{s_i a_L}\right)$ to $\alpha\left(\overrightarrow{s_i b_L}\right) - \theta$ in the anticlockwise direction and the feasible region T_i is the region enclosed by $\overrightarrow{s_i a_L}$, $\overrightarrow{s_i b_L}$ and the arc $\overset{\frown}{a_L b_L}$(i.e. the shadowed sector in Fig. 4(b)).

Given the feasible orientation range of sensor s_{i-1}, we calculate the feasible orientation range of s_i as follows:

1) If s_i is the first sensor, the feasible orientation range of s_i is calculated according to the intersection of circle C_i and the left boundary of region. The two intersection points are

denoted as p_i and p_j respectively (if C_i and the left boundary are tangent, p_i and p_j are the same point). We rotate sector S_i to intersect with the left boundary only at point p_i or p_j, denoting its orientation as A_i or A_j respectively. Thus, the feasible orientation range of s_i is the range from A_i to A_j .

2) If s_i is not the first sensor, then four cases are considered according to the locations of s_i and s_{i-1}.

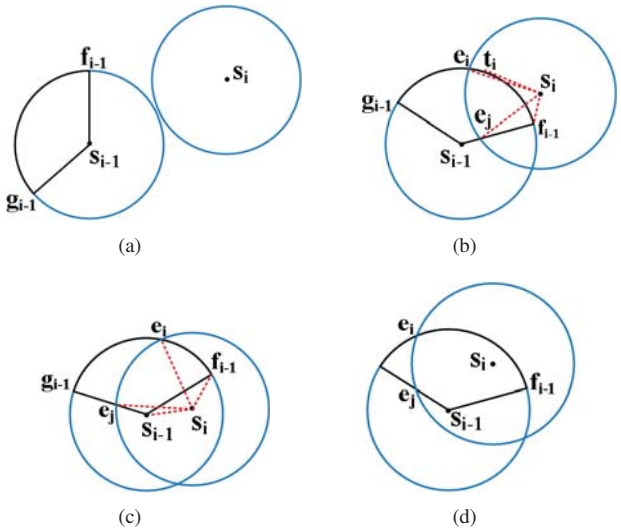


Fig. 5. Examples of calculating feasible orientation range

Case 1: C_i does not intersect with C_{i-1}. It implies that S_i cannot overlap with S_{i-1}, as shown in Fig. 5(a). Thus, there is a gap between these two sectors and the feasible orientation range of s_i is set to be [0,2π].

Case 2: C_i intersects with C_{i-1} and s_i is located outside C_{i-1}, as shown in Fig. 5(b). We define a vector set D. First draw two tangent lines from s_i to C_{i-1} with tangent points denoted as t_i and t_j. If t_i or t_j is in C_i and also in the sector T_{i-1}, then $\overrightarrow{s_it_i}$ or $\overrightarrow{s_it_j}$ will be added to D. The vectors $\overrightarrow{s_ie_i}$ and $\overrightarrow{s_ie_j}$ will be also added to D, where e_i and e_j are the intersection points of sector T_{i-1} and C_i. Suppose f_{i-1} and g_{i-1} are the end points of the arc of the sector T_{i-1}. If f_{i-1} or g_{i-1} is inside C_i, then $\overrightarrow{s_if_{i-1}}$ or $\overrightarrow{s_ig_{i-1}}$ will be added to D. We choose the vector in D with the smallest angle, denoted as v_i. We also choose one with the largest angle, denoted as v_i'. Thus, the feasible orientation range of s_i is the range from $\alpha\left(\overrightarrow{v_i}\right)-\theta$ to $\alpha\left(\overrightarrow{v_i'}\right)$. As shown in Fig. 5(b), the feasible orientation range of s_i is the range from $\alpha\left(\overrightarrow{s_it_i}-\theta\right)$ to $\alpha\left(\overrightarrow{s_if_{i-1}}\right)$.

Case 3: C_i intersects with C_{i-1} and s_i is inside C_{i-1} but outside T_{i-1}, as shown in Fig. 5(c). We also define a vector set D. If T_{i-1} and C_i do not intersect, S_i cannot overlap with S_{i-1}, which implies that there is a gap between them and the feasible orientation range of s_i is set to be [0,2π]; otherwise, $\overrightarrow{s_ie_i}$ or $\overrightarrow{s_ie_j}$ will be added to D, where e_i and e_j are the intersection points of sensor T_{i-1} and C_i. If f_{i-1} or g_{i-1} is inside C_i, then $\overrightarrow{s_if_{i-1}}$ or $\overrightarrow{s_ig_{j-1}}$ will be added to D. Similar as Case 2, we choose the vector in D with the smallest and largest angle, denoted as v_i and v_i'. Thus, the feasible orientation range of s_i is the range from $\alpha\left(\overrightarrow{v_i}\right)-\theta$ to $\alpha\left(\overrightarrow{v_i'}\right)$. As shown in Fig. 5(c), the feasible orientation range is the range from $\alpha\left(\overrightarrow{s_if_{i-1}}-\theta\right)$ to $\alpha\left(\overrightarrow{s_is_{i-1}}\right)$.

Case 4: C_i intersects with C_{i-1} and s_i is inside C_{i-1} as well as T_{i-1}, as shown in Fig. 5(d). Since s_i is located inside T_{i-1}, S_i intersects with T_{i-1} in any orientation, thus the feasible orientation range of s_i is [0,2π].

It is easy to know that the feasible orientation range of s_i is also related with the location of its closest right neighbor sensor. Thus, we introduce the notion of the right orientation range and then update the feasible orientation range using the right orientation range.

B. calculate the right orientation range

We first define the right orientation range.

Definition 8. Right orientation range**:** For sensor s_i**,** its right orientation range, denoted as R_i, includes all the orientations which satisfy that S_i can overlap with its closest right neighbor sensor or the right boundary of ROI.

The sector S_i first rotates such that it does not intersect with C_{i+1}. Then rotate S_i around s_i anticlockwise until it first meets C_{i+1}, denoting its starting edge as $\overrightarrow{s_ia_R}$. Continue rotating S_i around s_i anticlockwise until it first leaves C_{i+1}, denoting its end edge as $\overrightarrow{s_ib_R}$. Thus, the right orientation range O_i is the range from $\alpha\left(\overrightarrow{s_ia_R}\right)$ to $\alpha\left(\overrightarrow{s_ib_R}\right)-\theta$ in the anticlockwise direction, as shown in Fig. 4(c). Then, update the feasible orientation range O_i as the range from $\alpha\left(\overrightarrow{s_ia_L}\right)$ to $\alpha\left(\overrightarrow{s_ib_R}\right)-\theta$ and the feasible region T_i is the region enclosed by $\overrightarrow{s_ia_L}$, $\overrightarrow{s_ib_R}$ and the arc $\widehat{a_Lb_R}$(i.e. the shadowed sector in Fig. 4(d)).

In this subsection we will calculate the right orientation range of s_i.

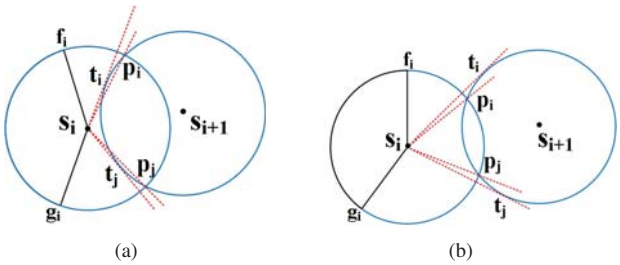


Fig. 6. Examples of calculating right orientation range

If the feasible orientation range of s_{i+1} has not been calculated, three cases will be considered.

Case 1: C_i does not intersects with C_{i+1}. It implies that S_i cannot overlap with S_{i+1}. Thus, there is a gap between them.

Case 2: C_i intersects with C_{i+1} but s_i is outside C_{i+1}, as shown in Fig. 6. We draw two tangent lines from s_i to C_{i+1} with the tangent points denoted as t_i and t_j. Let p_i and p_j denote the intersections of C_i and C_{i+1}. If two tangent points t_i and t_j are inside C_i, we rotate S_i to intersect with C_{i+1} only at point t_i or t_j, denoting the orientation of sensor s_i as

A_i or A_j respectively; otherwise, we calculate A_i and A_j by rotating S_i to intersect with C_{i+1} only at p_i or p_j. Then the right orientation range is the range from A_i to A_j. In Fig. 6(a), the right orientation range is $\alpha\left(\overrightarrow{s_it_i}-\theta\right)$ to $\alpha\left(\overrightarrow{s_it_j}\right)$. In Fig. 6(b), the right orientation range is $\alpha\left(\overrightarrow{s_ip_i}-\theta\right)$ to $\alpha\left(\overrightarrow{s_ip_j}\right)$.

Case 3: C_i intersects with C_{i+1} and s_i is inside C_{i+1}. It implies that whatever orientation S_i rotates, S_{i+1} can rotate to intersect with it, so the right orientation range of s_i is [0,2π].

After the right orientation range is calculated, update the feasible orientation range of sensor s_i by the intersection of it and right orientation range. Besides, we will update the feasible orientation ranges of all the sensors on its left side belonging to the same sub-barrier. The process of calculation is similar as the process of calculating the feasible orientation range in the above subsection.

C. Algorithm Description

In this subsection, we propose the algorithm of determining whether there exists a barrier.

The basic idea of this algorithm is to calculate the feasible and right orientation range of each sensor from the left to the right. If none of the intersection of these two ranges is null, it implies that there exists a global barrier; Otherwise, there does not exist one and we will calculate the feasible orientation ranges of the sensors to form a set of sub-barriers.

The procedures of this algorithm are described as follows:

1) For each sensor s_i, calculate its feasible and right orientation range using the methods in the above two subsection.

2) If the intersection of these two ranges is null, then it means that there does not exist a global barrier and the current sub-barrier ends with sensor s_i. A new sub-barrier starts with sensor s_{i+1}, go to 1); otherwise, go to 3).

3) For each sensor s_j $(j < i)$ back to forth, which belongs to the current sub-barrier, recalculating its right orientation range. If the intersection of these two ranges is null, then it means that there does not exist a global barrier and the current sub-barrier ends with sensor s_i. A new sub-barrier starts with sensor s_{i+1}, and recalculate its feasible and right orientation range, go to 2); If none of the intersection of all these sensors is null, update their feasible orientation ranges by the intersections and go to 1).

Note that if s_i is the first or the last sensor, its feasible or right orientation range should be calculated based on the locations of left or right boundary of ROI. If a new sub-barrier starts with sensor s_i, then its feasible orientation range is initially set to be [0, 2π].

The detail of this algorithm is shown in Algorithm 1. Let cur denote the first sensor's index of the current sub-barrier. Let isSuccess denote the variable indicating whether there exists a global barrier.

Algorithm 1 Algorithm of DBC

Input: Sensor Set S
Output: Feasible Orientation Range Set O=$\{O_i|1 \le i \le n\}$, boolean isSuccess

```
1: cur = 0, isSuccess = true
2: for each sensor s_i ∈ S
3:    calculate feasible and right orientation range of S_i as O_i and R_i
4:    if O_i ∩ R_i is null then
5:       isSuccess = false
6:       cur = i + 1
7:       continue
8:    else
9:       O'_i = O_i
10:      O_i = O_i ∩ R_i
11:   end if
12:   if i > 1 then
13:      position = i − 1
14:      while position ≥ cur do
15:         calculate right orientation range of S_position as R'_position
16:         O'_position = O_position ∩ R'_position
17:         if O'_position is null then
18:            O_i = O'_i
19:            cur = i + 1
20:            isSuccess = false
21:            break
22:         end if
23:         position = position − 1
24:      end while
25:   end if
26:   if position < cur then
27:      position = i − 1
28:      while position ≥ cur do
29:         O_position = O'_position
30:         position = position − 1
31:      end while
32:   end if
33: end for
34: return isSuccess and O
```

VI. Algorithm of OBC

In this section, we propose an algorithm of finding the orientations of all the sensors to form a barrier or form a set of sub-barriers, called algorithm of OBC.

The basic idea of this algorithm is to choose an orientation of each sensor from its feasible orientation range such that its sensing area overlaps with that of its closest left neighbor sensor. If there is no such orientation, it implies that there is a gap between this sensor and its left closest neighbor sensor and a new sub-barrier starts from this sensor. Rotate this sensor to the boundary of this range; otherwise, rotate this sensor to intersect with its closest left neighbor sensor by choosing one orientation from its feasible orientation range. Note that the feasible orientation range of one sensor may include more than one ranges.

Algorithm 2 shows the detail of OBC algorithm.

Fig.7 presents a part of final deployment of sensors using our proposed algorithm and the Distributed Gap Mending

Algorithm 2 OBC Algorithm

Input: Sensor Set S

Output: The orientations of Sensors S

```
Calculate feasible orientation range set O by Algorithm 1
for each sensor s_i ∈ S
    if i = 1
        rotate S_i to one boundary of s_i's feasible orientation range
    else
        isRotated = false
        for each range of feasible orientation range o ∈ O_i
            if there is an orientation in o such that S_i can intersect with S_{i+1} then
                rotate S_i to this orientation
                isRotated = true
            end if
        end for
        if isRotated = false
            rotate S_i to one boundary of s_i's feasible orientation range
        end if
    end if
end for
return S
```

Algorithm [5]. Sensors are deployed with view angle θ = 60. Fig. 7(a) shows a barrier formed using our algorithm. Fig. 7(b) shows the orientations of sensors computed in the work [5] under the same initial sensor deployment, which results in a barrier gap.

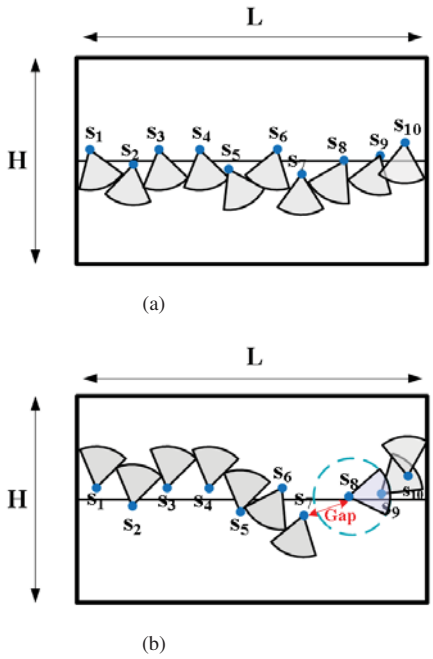


Fig. 7. **(a)** The final sensor rotation scheme by our algorithm; **(b)** The final sensor rotation scheme by the Distributed Gap Mending Algorithm [5]

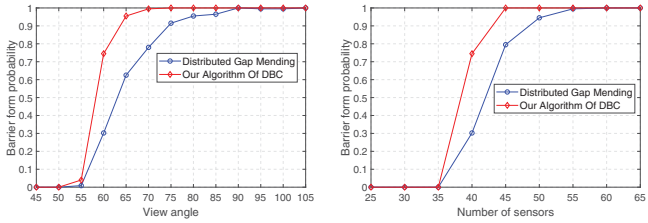


Fig. 8. **(a)**Probability of forming a strong barrier vs view angle of sensors **(b)**Probability of forming a strong barrier vs number of sensors

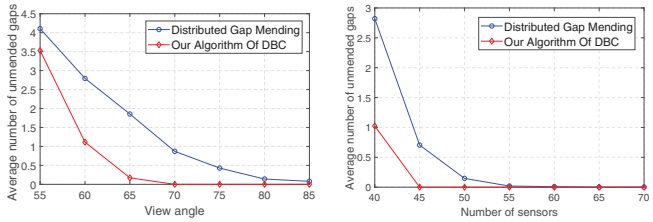
VII. Simulation Results

In this section, we evaluate our proposed algorithm using Java by compared to the Distributed Gap Mending Algorithm [5]. We assume the ROI is a rectangle belt region with length L = 500m and H = 100m. Sensors are deployed randomly along the middle-line of this region. The initial orientation angle of the sensors is randomly chosen from $[0, 2\pi]$. The sensing range of sensor is 15m. In each experiment, we evaluate the probability of forming a barrier and unmended gaps of forming a strong barrier. The result is an average result of 100 experiments.

First, we evaluate the probability of forming a barrier when the view angle changes. The number of sensors is 40. As shown in Fig. 8(a), the probability of forming barrier increases as the view angle of sensors increases. Our proposed algorithm has a higher successful ratio than the distributed algorithm. Moreover, our algorithm can form a barrier when the view angle is not smaller than 70 while the distributed algorithm can form a barrier when the view angle must be larger than 90. Since sensors with larger view angle consume more energy to sense, the barrier formed by our algorithm with smaller view angle can have a long lifetime than that by the distributed algorithm.

Then, we investigate the probability of forming barrier when the number of sensors changes. The view angle of sensors is 60. The number of sensors varies from 25 to 65 with a step 5. Fig. 8(b) shows that the probability of forming barrier increases as the number of sensors increases. Our proposed algorithm always has a higher successful ratio than the distributed algorithm. Moreover, our algorithm can form a barrier only using 45 sensors while the distributed algorithm must use 55 sensors. It implies that under the same conditions, our algorithm needs fewer directional sensors, which can decrease the cost of directional sensors.

Furthermore, we evaluate the unmended barrier gaps when the view angle and number of sensors changes respectively. In Fig. 9(a), the number of sensors is 40. It demonstrates that our algorithm of DBC always results in fewer unmended gaps than the distributed algorithm as the view angle of sensors increases. Moreover, the result obtained by our algorithm decreases more quickly than that by distributed algorithm. Fig.9(b), the view angle of sensors is 60. The result shows



(a) (b)

Fig. 9. **(a)**Unmended gaps of forming a strong barrier vs view angle of sensors **(b)**Unmended gaps of forming a strong barrier vs number of sensors

that our proposed algorithm has a better performance than the distributed algorithm as the number of sensors increases. Our algorithm results in only one gap while the distributed algorithm results in 2.8 gaps when the number of sensors is 40. Thus, our algorithm can leave fewer gaps than the distributed algorithm when the number of sensors is not sufficient.

VIII. Conclusion

We present an exact algorithm of determining whether there exists a strong barrier by rotating the directional sensors with the line-based directional sensors. Then we also propose an algorithm of determining the sensors' orientations for forming a strong barrier or a set of sub-barriers if the number of sensors is not sufficient. Simulation results indicate that our algorithm outperforms the distributed algorithm.

IX. Acknowledgment

This work is supported by National Natural Science Foundation of China (Grant No. 61702198 and 61976097).

References

[1] Santosh Kumar, Ten H Lai, and Anish Arora, "Barrier coverage with wireless sensors", in Proceedings of the 11th annual international conference on Mobile computing and networking (2005), pp. 284--298.

[2] Jiaoyan Chen, Bang Wang, Wenyu Liu, Laurence T Yang, and Xianjun Deng, "Rotating directional sensors to mend barrier gaps in a line-based deployed directional sensor network", IEEE Systems Journal 11, 2 (2014), pp. 1027--1038.

[3] Zhibo Wang, Jilong Liao, Qing Cao, Hairong Qi, and Zhi Wang, "Achieving k-barrier coverage in hybrid directional sensor networks", IEEE Transactions on Mobile Computing 13, 7 (2013), pp. 1443--1455.

[4] Anwar Saipulla, Cedric Westphal, Benyuan Liu, and Jie Wang, "Barrier coverage of line-based deployed wireless sensor networks", in IEEE INFOCOM 2009 (2009), pp. 127--135.

[5] Yueshi Wu and Mihaela Cardei, "Distributed algorithms for barrier coverage via sensor rotation in wireless sensor networks", Journal of Combinatorial Optimization 36, 1 (2018), pp. 230--251.

[6] Anwar Saipulla, Benyuan Liu, and Jie Wang, "Barrier coverage with airdropped wireless sensors", in MILCOM 2008-2008 IEEE Military Communications Conference (2008), pp. 1--7.

[7] Jie Shen, Zhibo Wang, and Zhi Wang, "Fault tolerant line-based barrier coverage formation in mobile wireless sensor networks", International Journal of Distributed Sensor Networks 11, 10 (2015), pp. 930585.

[8] Zhao Zhang, Weili Wu, Jing Yuan, and Ding-Zhu Du, "Breach-Free Sleep-Wakeup Scheduling for Barrier Coverage With Heterogeneous Wireless Sensors", IEEE/ACM Transactions on Networking 26, 5 (2018), pp. 2404--2413.

[9] Jiaoyan Chen, Bang Wang, Wenyu Liu, Xianjun Deng, and Laurence T Yang, "Mend barrier gaps via sensor rotation for a line-based deployed directional sensor network", in 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference … (2013), pp. 2074--2079.

[10] Shuangjuan Li and Hong Shen, "Minimizing the maximum sensor movement for barrier coverage in the plane", in 2015 IEEE Conference on Computer Communications (INFOCOM) (2015), pp. 244--252.

[11] Lu Zhao, Guangwei Bai, Hang Shen, and Zhenmin Tang, "Strong barrier coverage of directional sensor networks with mobile sensors", International Journal of Distributed Sensor Networks 14, 2 (2018), pp. 1550147718761582.

[12] Dan Tao, Shaojie Tang, Haitao Zhang, Xufei Mao, and Huadong Ma, "Strong barrier coverage in directional sensor networks", Computer Communications 35, 8 (2012), pp. 895--905.

普通高等教育农业农村部“十四五”规划教材

普通高等教育“十四五”规划教材

Python

》语言《

程序设计教程

陈湘骥　梁　云◎主编

PYTHON
LANGUAGE
PROGRAMMING TUTORIAL

中国农业大学出版社
China Agricultural University Press

内容简介

本书介绍了Python编程语言的基础知识和程序设计方法。全书共分11章，循序渐进地引导读者逐步学习Python的基本概念和应用。第1、2章介绍与程序设计相关的概念、基本数据类型和表达式的使用；第3、4章进一步讲解输入与输出、选择结构、循环结构和异常处理结构的语法与应用；第5、6章深入讲解和学习自定义函数、结构化数据类型及其应用；第7章介绍了文件操作；第8～11章通过案例引导读者学会运用标准库和扩展库解决实际应用中遇到的问题，展示了Python语言在GUI编程、数据分析、数字图像处理、网络编程等领域的应用。本书力求概念清楚、结构严谨，叙述通俗易懂，采用“提出问题—分析问题—解决问题”的方式阐述程序设计的思路与方法。

本书适合高等学校计算机及相关理工科专业的学生使用，也可以作为各专业计算机语言类公共课教材和程序设计初学者的参考用书。

图书在版编目(CIP)数据

Python语言程序设计教程 / 陈湘骥，梁云主编. --北京：中国农业大学出版社，2024.9
ISBN 978-7-5655-3240-5

Ⅰ. TP311.561

中国国家版本馆CIP数据核字第202423UZ77号

农业农村部教材办公室审定编号：NY-1-0157

书　　名　Python语言程序设计教程
　　Python Yuyan Chengxu Sheji Jiaocheng
作　　者　陈湘骥　梁　云　主编

策划编辑　张秀环　　责任编辑　潘博闻
封面设计　李尘工作室
出版发行　中国农业大学出版社
社　　址　北京市海淀区圆明园西路2号　　邮政编码　100193
电　　话　发行部 010-62733489，1190　　读者服务部 010-62732336
　　编辑部 010-62732617，2618　　出　版　部 010-62733440
网　　址　http://www.caupress.cn　　E-mail　cbsszs@cau.edu.cn
经　　销　新华书店
印　　刷　河北朗祥印刷有限公司
版　　次　2024年11月第1版　2024年11月第1次印刷
规　　格　185 mm×260 mm　16开本　13.25印张　321千字
定　　价　45.00元

编审人员

主　编　陈湘骥　梁　云

副主编　李玉峰　邱少健

编　者（按姓氏笔画排序）

王国华　李双娟　李玉峰　邱少健

陈湘骥　林旭东　梁　云

主　审　肖　磊

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导庞绍宇荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛全国总决赛C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602113362

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年6月23日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导吴文熙荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛全国总决赛C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602114823

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年6月23日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导罗林荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602057338

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导薛欣琪荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602058432

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导陈立宇荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602056743

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导何嘉豪荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602058361

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导庞绍宇荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组一等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602057408

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导吴文熙荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组一等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1602057343

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导陈嘉玲荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区Java软件开发大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1605014333

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导陈晓仪荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区Java软件开发大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1605014597

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导方利岚荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区Java软件开发大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1605014866

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导蒋定金荣获第十六届蓝桥杯全国软件和信息技术专业人才大赛广东赛区Java软件开发大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：1605014936

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2025年5月26日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导吴文熙荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛全国总决赛C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021580221

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年6月2日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导张伟健荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021549170

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导庞绍宇荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021549360

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导吴文熙荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组一等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021549361

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导马正杰荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021549362

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导罗林荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组二等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021548518

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导陈嘉玲荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021548536

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导刘泽佳荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021548743

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日

蓝桥杯大赛

获奖证书

华南农业大学李双娟：

指导陈均林荣获第十五届蓝桥杯全国软件和信息技术专业人才大赛广东赛区C/C++程序设计大学B组三等奖，被评为优秀指导教师。

特发此证，以资鼓励。

证书编号：021548858

证件号码：430482198310260047

工业和信息化部
人才交流中心

蓝桥杯大赛组委会

2024年4月29日