博士学位论文

无线传感器网络中多路径路由的研究

RESEARCH ON MULTI-PATH ROUTING IN WIRELESS SENSOR NETWORKS

张可佳

2012 年 6 月

国内图书分类号：TP393　　学校代码：10213
国际图书分类号：681.5　　密级：公开

工学博士学位论文

无线传感器网络中多路径路由的研究

博 士 研 究 生：张可佳
导　　师：高宏教授
申 请 学 位：工学博士
学　　科：计算机软件与理论
所 在 单 位：计算机科学与技术学院
答 辩 日 期：2012 年 6 月
授予学位单位：哈尔滨工业大学

Classified Index: TP393
U.D.C.: 681.5

Dissertation for the Doctoral Degree in Engineering

RESEARCH ON MULTI-PATH ROUTING IN WIRELESS SENSOR NETWORKS

Candidate: Zhang Kejia

Supervisor: Prof. Gao Hong

Academic Degree Applied for: Doctor of Engineering

Specialty: Computer Software and Theory

Affiliation: School of Computer Science and Technology

Date of Defence: June, 2012

Degree-Conferring-Institution: Harbin Institute of Technology

摘　要

一个（无线）传感器网络是由部署在监测区域内大量廉价微型的传感器节点组成，通过无线通信方式形成的多跳自组织网络系统。由于传感器节点采用无线方式进行通信并且通信距离十分有限，在传感器网络中的数据传输通常需要由多个中间节点进行转发，这就使得如何设计高效的路由策略成为传感器网络研究的关键问题。给定源节点s和目的节点t，一组连接s和t的路径是（节点）不相交路径如果它们之中的任意两条不共享任何中间节点。与采用单一路径进行路由相比，采用多条路径，特别是多条（节点）不相交路径，进行路由可以很大程度上提升传输的吞吐量、传输的可靠性，网络的负载平衡以及信息的安全性。另一方面，如果不同路径上的两个节点在彼此的通信范围内（互为邻居节点），两条路径上的数据传输就会发生无线干扰。路径间大量的无线干扰会很大程度上降低传输的吞吐量、增加节点的能量消耗以及降低信息的安全性。为此，非干扰路径的概念被提出。多条路径是非干扰路径如果它们之中的任意两条既没有公共节点也没有互为邻居的节点。利用非干扰路径进行路由可以进一步提升网络的性能。本文针对传感器网络的特点，研究了在网络中如何建立不相交路径和非干扰路径等问题。主要研究成果包括：

第一，本文研究了k不相交路径问题，即给定两个节点s和t以及正整数k，寻找连接s和t的k条不相交路径。这个问题的难点在于如何保证答案正确性，即如果网络中存在k条不相交路径，算法一定能够输出k条不相交路径。虽然已经有一些应用于无线网络中的分布式方法用来寻找连接s和t的多条不相交路径，但是这些方法都不能保证答案正确性。一些图论中的集中式算法虽然可以保证答案正确性，但是这些算法都需要收集整个网络的拓扑信息从而不适合于传感器网络。本文针对这个问题，提出了一个高效的分布式算法EDA，只需要在节点间交换短小的数据并收集少量的信息，就可以找到网络中的k条不相交路径。EDA可以保证答案正确性。实验结果表明，与同样可以保证答案正确性的集中式方法相比，EDA的通信效率要高得多。与当今最具效率的分布式方法（不能保证答案正确性）相比，EDA在通信效率方面也是完全可以比较的。并且在找到的路径数量方面，EDA具有明显的优势。

第二，本文研究了最优k不相交路径问题，即给定两个节点s和t以及正整数k，寻找连接s和t的k条不相交路径，使得这些路径的长度之和最小。这里假设网络中的每个链接都有一个长度来表示其能量代价或者传输延迟。这个问题的难点在于如何保证答案正确性和结果最优性。所谓结果最优性是指算法

输出的路径集合在所有的可能解中具有最小的长度之和。现有的分布式寻找连接s和t的多条不相交路径的方法既不能保证答案正确性也不能保证结果最优性。虽然有一些集中式方法可以保证答案正确性和结果最优性，但是它们都需要收集全网的拓扑信息，从而不适合应用在传感器网络中。本文提出了一个高效的完全分布式算法OFDP，通过节点间交换少量的数据，就可以找到k条长度之和最小的不相交路径。OFDP既能保证答案正确性又能保证结果最优性。OFDP不需要从网络中收集任何信息，节点的信息交换也不依赖于任何结构。如果不考虑路径的长度，通过对OFDP进行简化，本文还提出了另一个用于解决k不相交路径问题的分布式算法FDP。实验结果验证了这两个算法的高效。

第三，本文研究了长度受限的不相交路径问题，即给定两个节点s和t以及长度限制L，寻找最多的连接s和t的不相交路径，使得每条路径的长度都不大于L。首先考虑每个链接的长度都是1的情况。当$L \geq 5$时，这个问题不但是NP完全问题而且是APX完全问题。因此不存在这个问题的多项式时间近似模式，以常数近似比来近似这个问题也是非常难的。迄今为止，只有一个关于这个问题的启发式算法被提出，而且该算法采取的是集中式的处理方式。本文提出了这个问题的一个$\sqrt{n}$近似算法GDA以及GDA的分布式实现。其中，n是网络中节点的数量。另外，本文还给出了这个问题的一个分布式的启发算法OPDA。OPDA可以很好地应对每个链接都有任意长度的情况。实验结果表明，无论是在找到的路径数量方面还是在通信效率方面，OPDA都有很高的效率。

第四，本文研究了（广义）非干扰路径问题，即给定网络中的k对节点$\{s_1, t_1\}, \ldots, \{s_k, t_k\}$，用非干扰路径来连接最多的节点对。本文证明了即使当$k = 2$时，这个问题也是一个NP难问题。本文还证明了对于任意的k和$\epsilon > 0$，以$m^{1/2-\epsilon}$来近似这个问题是NP难的。其中，m是网络中链接的数量。因此，$\sqrt{m}$是可能取得的最好的近似比。本文给出了这个问题的一个贪心算法GSPW，并证明了GSPW的近似比为$\sqrt{m}$。本文还给出了这个问题的一个在线算法OSBPW，及其分布式实现，并证明了OSBPW拥有很好的理论下界。

针对所研究的问题，本文提出的方法充分考虑了传感器网络的特点，弥补了现有研究工作的不足。对于有些问题，本文更是首次给出了可分析的近似算法或者首次证明了问题的近似比下界，在理论上有所创新和发现。

关键词： 传感器网络；多路径路由；不相交路径；非干扰路径

Abstract

A (wireless) sensor network comprises a large number of cheap small-size sensor nodes. These sensor nodes form a multi-hop self-organizing network. Since sensor nodes communicate with each other through wireless radio devices and the transmitting range of sensor nodes is very limited, data transmissions in sensor networks usually need to be relayed by many hops, which makes efficient message routing a crucial problem in this area. Given a source node s and a terminal node t, a set of paths connecting s and t are (node) disjoint paths if any two of them do not have any common nodes besides s and t. Compared with routing through a single path, routing through multiple paths, especially multiple (node) disjoint paths, can greatly improve network performance in the aspects of transmission throughput, transmission reliability, balanced load and information security. On the other side, if two nodes in two different paths are in the transmitting range of each other (so they are neighbors), the data transmissions of the two paths will interfere with each other. Too much wireless interference among routing paths will greatly reduce data throughput, increase energy consumption of senor nodes and degrade information security. Thus, the notion of non-interfering paths is proposed. A set of paths are non-interfering paths if any two of them have neither common nodes nor adjacent nodes. Routing through non-interfering paths will further improve network performance. Specific for sensor networks, this paper researches on the problems of how to construct multiple disjoint paths and non-interfering paths. The main research results are:

First, this paper studies k disjoint paths problem, i.e., given two nodes s and t; a positive integer k, find k disjoint paths connecting s and t. The difficulty of this problem is to guarantee answer correctness, i.e., an algorithm will output k disjoint paths if there exist k disjoint paths in the given network. Some distributed algorithms have been proposed for finding multiple paths connecting two given nodes in wireless networks, but all of them can not guarantee answer correctness. Although some centralized algorithms in graph theory area can guarantee answer correctness, all of them need to collect the topology information of the whole network, which makes them not suitable for sensor networks. This paper proposes an efficient distributed algorithm EDA for this problem. By exchanging short messages among the nodes and collecting a small amount of information, EDA will find k disjoint paths with answer correctness guarantee. Experimental results show that

compared to the existing centralized algorithms which can also guarantee answer correctness, EDA is much more communication-efficient. When compared to the most efficient existing distributed algorithm (which can not guarantee answer correctness), the communication cost of EDA is totally comparable, and EDA has apparent advantage at the aspect of found path number.

Second, this paper studies optimal k disjoint paths problem, i.e., given two nodes s and t; a positive integer k, find k disjoint paths connecting s and t with minimum sum length. Here, each link in the given network has a length to denote its energy cost or transmitting delay. The difficulty of this problem is to guarantee answer correctness and optimal result. "optimal result" means that the paths output by an algorithm have minimum sum length among all the possible results. All the existing distributed algorithms for finding multiple disjoint paths connecting s and t can neither guarantee answer correctness nor optimal result. Although some centralized algorithms in graph theory area can guarantee answer correctness and optimal result, they have to collect the topology information of the whole network. The cost of such operation is unbearable for sensor networks. This paper proposes an efficient and totally distributed algorithm OFDP for this problem. By exchanging short messages among the nodes, OFDP can find k disjoint paths with minimum sum length, with guarantee of answer correctness and optimal result. OFDP does not need to collect any information from the network, and the data exchanging does not rely on any pre-formed structures. If not considering length of paths, by simplifying OFDP, this paper gives another distributed algorithm FDP for k disjoint paths problem. The experimental results confirm the efficiency of the two algorithms.

Third, this paper studies length-bounded k disjoint paths problem, i.e., given two nodes s and t; a length constraint L, find the maximum number of disjoint paths connecting s and t, where the length of each path is no more than L. At first, a simple case is considered, where the length of each link is 1. When $L \geq 5$, this problem is not only NP complete but also APX complete. Therefore, there is no polynomial time approximation scheme for this problem, and it is hard to approximate this problem with constant ratio. So far, there is only one heuristic algorithm proposed for this problem, and the algorithm processes in a centralized way. This paper proposes an $\sqrt{n}$-approximation algorithm GDA for this problem, and gives the distributed implementation of GDA. Here, n is the number of the nodes in the given network. In addition, this paper gives a distributed heuristic algorithm OPDA for this problem. OPDA handles the more complicated situation very well, where each link has an arbitrary length. The experimental results show that OPDA

is very efficient in both aspects of found path number and communication cost.

Fourth, this paper studies (general) non-interfering paths problem, i.e., given k node pairs $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, connect the maximum number of these node pairs via non-interfering paths. This paper proves that this problem is NP hard even when $k = 2$. This paper also proves that for general k and any $\epsilon > 0$, it is NP hard to approximate this problem within $m^{1/2-\epsilon}$, where m is the number of the links in the given network. Thus, $\sqrt{m}$ is the best approximation ratio we can achieve. This paper gives a greedy algorithm GSPW, and proves that GSPW is an $\sqrt{m}$-approximation algorithm. This paper also gives an on-line algorithm OSBPW and its distributed implementation. OSBPW has a good lower bound in theory.

For the studied problems, the approaches proposed by this paper fully consider the characteristics of sensor networks, and compensate for the lack of the existing research works. For some of these problems, it is the first time that someone give algorithms with analyzable approximation ratios or someone give the lower bounds to approximate them, so this paper has innovations and discoveries in theory.

Keywords: Wireless Sensor Networks, Multi-Path Routing, Disjoint Paths, Non-Interfering Paths

目 录

Contents

第 1 章 绪论

1.1 课题的背景和意义

微电子技术的发展使得处理器、存储器和感知元件可以越做越小而功能却越来越强。计算技术的发展可以使得我们利用有限的资源来完成更复杂的计算。无线通信技术的进步使得我们可以在保证传输的情况下将无线通信装置集成到很小的芯片上。于是，在微小体积内能够集成信息采集、数据处理和无线通信等多种功能的传感器应运而生。无线传感器网络（Wireless Sensor Network，WSN）是由部署在监测区域内大量廉价微型的传感器节点组成，通过无线通信方式形成的多跳自组织网络系统，用来协作地感知、采集和处理监测区域内感知对象的信息，并发送给观察者[1]。传感器网络[1]能够让我们更好的认识物理世界，使得我们随时随地都可以获取大量详实的物理世界信息。传感器网络在国防军事、环境监测、交通管理、医疗卫生、农业生产、工业生产、深海勘测和矿井挖掘等领域均具有广泛的应用前景[2–16]. 传感器网络是信息感知和采集的一场革命，在新一代网络中具有关键作用。通过传感器网络对数据世界和物理世界的整合，人们能够更好的认识自然、认识环境并改造它们。正是由于传感器网络的广泛的应用前景与巨大的应用价值，它被认为是改变世界的十大新技术之一，引起了世界各国的工业界、学术界及军事部门的广泛关注，成为计算机科学领域的新的研究热点。

传感器网络多跳自组织的特点决定了数据在传输过程中需要多次地被转发，所以路由策略的制定就显得尤其重要。另外，传感器网络以下这些特点决定了不能将其它网络中的路由策略直接应用于传感器网络，而是应该独立为其设计路由策略。1）传感器节点多是电池供电而且不可能经常更换电池，所以节省能量就成为了路由设计的主要目标；2）传感器节点的计算和存储功能十分有限，路由的中间节点不能缓存大量数据，也不能执行太复杂的程序，所以路由策略应该尽量简单和避免大量缓存数据。3）由于传感器节点的通信能力有限，节点或者链接失效时有发生，这就使得传感器网络中的数据传输不可避免的伴随数据丢失的现象，所以路由策略还应该尽量保证传输的可靠性。4）传感器节点的无线通信带宽有限，所以传感器网络的路由策略还应该考虑如何利用有限的资源来提高数据传输的吞吐量。5）传感器网络的寿命受制于网络

[1]本文中的“传感器网络”即指“无线传感器网络”

中的每个节点，所以路由策略应该充分考虑网络中的负载平衡，避免让某些节点承担过多的工作而提前失效。

给定一对路由的源节点和目的节点，相比于只用一条路径传输数据，在这两个节点间采用多条路径，特别是多条不相交路径，来传输数据可以大大增加传输的可靠性、吞吐量以及网络的负载平衡。如果考虑到安全因素，采用多条路径传输数据还可以增强网络信息的安全性。所以，传感器网络中的多路径路由策略就成为了一个重大的挑战性课题，具有很大的理论和实际意义。另外，当网络中有多对路由需求（即多对源节点和目的节点）需要同时被满足时，如何优化的布置这些路由使得网络达到最优的传输状态（包括可靠性、吞吐量、负载平衡性等等）也成为了急需解决的关键性问题。

1.2 传感器网络

1.2.1 传感器网络的结构和特点

传感器网络可以在独立的环境下运行，也可以通过基站（网关）连接到Internet上，使用户可以远程访问。如图 1-1 所示，一个典型的无线传感器网络的结构包括分布式传感器节点（sensor nodes）、基站（sink）、互联网（internet）和客户端（client）等。

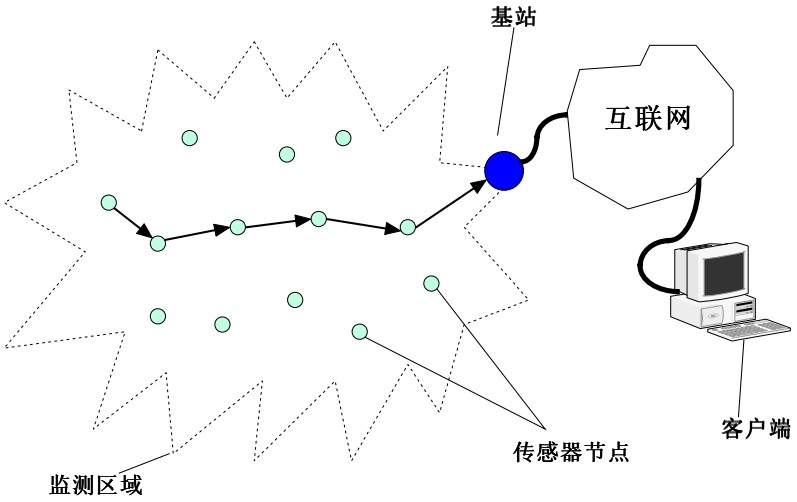


图 1-1 一个典型的传感器网络

Fig.1-1 A typical sensor network

传感器节点是传感器网络的基本组成部分。绝大多数传感器节点的体积都非常小，甚至小如灰尘[17]。但是它们却在很小的体积内集成了信息采集、数据处理、无线通信等多种功能。如图 1-2 所示，一个传感器节点包含了四个基本

单元：感知单元、处理单元、通信单元以及电源部分[18, 19]。感知单元集成了感知元件，可以感知温度、湿度、光强、压力、化学成分、磁场强度、放射性强度等等[20–22]，并将其转换为数字信号。有些传感器还可以采集周围的声音[23, 24]和图像[25, 26]。处理单元是由CPU、存储器、嵌入式操作系统[27–29]等组成，负责协调节点各部分的工作，如对感知部件获取的信息进行必要的处理、保存，控制感知部件和电源的工作模式等。通信单元集成了无线接收器和MAC模块(现今多数节点将MAC协议硬件化[30, 31])，负责与其它传感器或基站进行通信。电源负责给其它三个单元提供能量。通常情况下，体积微小的传感器节点是由电池供电。这些传感器节点又大多被放置在人员难以到达的环境，如战场、输油管道、深海、地下，所以几乎没有机会为传感器更换电池[32, 33]。因此，如何节省能量从而延长传感器节点的寿命就成为了传感器网络设计者最主要的设计目标。此外，对于传感器节点来说，可选择的其他功能单元包括：定位系统[34, 35]、运动系统[36–38]以及发电装置[39, 40]等。

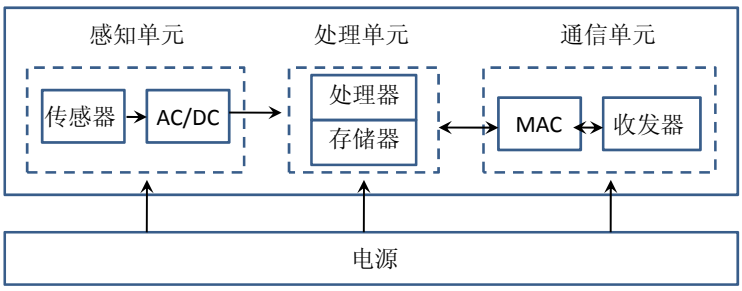


图 1-2 传感器节点的结构

Fig.1-2 The structure of sensor node

基站节点是传感器网络中一个功能强大的节点，例如一个小型的计算机[41]，通常与互联网相连。基站节点负责：1）收集传感器网络中的数据并发送给客户端；2）将客户端布置的任务（如下发的查询）发送给网络中的节点。根据基站节点的特点，尤其是没有能量限制的特性，我们还可以考虑将一些复杂的通信和计算工作交给基站节点完成。

传感器节点被大量地部署在监测区域内（通常是随机布置，如飞机播撒）。这些节点通过自组织方式构成无线网络，以协作的方式感知和处理监测区域内的信息。根据文献[42]，传感器节点各项操作的能量消耗如图 1-3 所示。对于传感器节点，通信操作是最耗费能量的，将1比特数据传输100米所消耗的能量大约相当于执行3000条计算指令消耗的能量。与接收和发送数据消耗的能量相比，计算和感知数据消耗的能量几乎可以忽略。于是，为传感器网络节省能量主要就是考虑如何节省通信时消耗的能量。无线通信的能量消耗与数据传输的

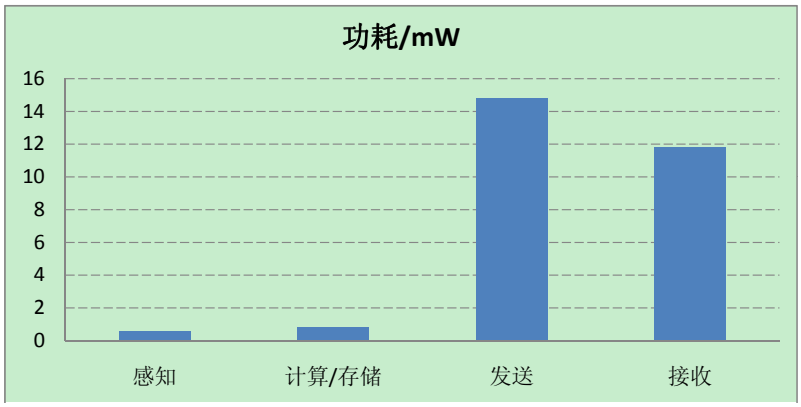


图 1-3 传感器节点的能量消耗

Fig.1-3 The power consumption of sensor node

距离有关。研究表明[2]，传输的能量消耗与通信距离的关系为

$$E = kd^n$$

其中，k为与数据长度有关的常量，d为传输距离，$2 < n < 4$。于是，将数据直接传输很长的距离就不如将数据沿途转发多次节省能量。假设如图 1-4 所示，节点s要发送数据给节点t。如果s直接将数据发送给节点t，消耗的能量为$E = kd^n$。如果s将数据沿着$s \to u \to v \to t$的路径发送（中间由u和v进行转发）,消耗的总能量为$E = k(d_1^n + d_2^n + d_3^n)$。显然，后一种策略更加节省能量。所以，我们应该在满足连通度的前提下尽量减少单跳通信距离。另一方面，传感器节点功能简单的特点也制约了其通信半径。因此，传感器节点的通信半径大多在100m以内。这就决定了传感器网络多跳自组织的特点，也就是网络中的数据传输通常需要由多个中间节点进行转发，每个节点在此过程中既可以是数据源、数据目的地，也可以是转发节点。

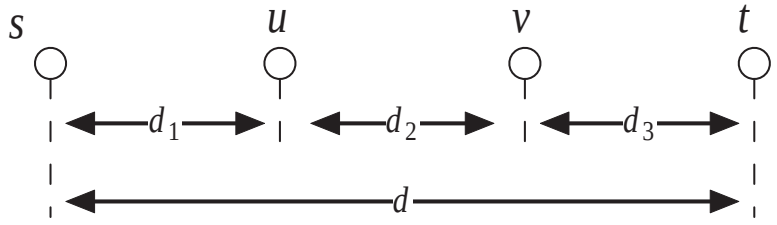


图 1-4 一个转发数据的列子

Fig.1-4 An example of relaying data

其它网络相比，传感器网络具有以下特点：

- 能量优先性。由于体积微小的传感器节点通常采用电池供电，而且被放置在人员难以到达的区域从而很难为其更换电池，所以如何为节点节省能量从而延长整个网络的寿命就成为了传感器网络设计的主要目标。这

是传感器网络区别于其它网络的一个显著特征。

- 计算和存储能力有限。由于传感器节点体积微小又要考虑到能量消耗，其上的处理器和存储器的功能十分有限，因此传感器节点执行不了复杂的计算操作和大规模的存储。
- 通信能力有限。由于通信芯片体积微小而且还要考虑能量消耗，所以传感器节点的通信能力十分有限，传输距离大多在100米以内，传输带宽通常只有几百kbps。
- 大规模网络。考虑到传感器节点体积微小造价低廉的特点，为了获得更准确的信息，使用者通常会在监测区域内布置大量的传感器节点。节点的数量可能达到成千上万甚至更多，而且分布在很大的地理区域内。
- 多跳自组织网络。通常，传感器节点的位置不能被预先精确设定，节点间是否能够通信也不能预先知道。例如，通过飞机播撒大量的传感器节点到战场上。所以，传感器节点需要有自组织的能力，根据路由协议和节点间的邻居关系自动形成可以转发数据的网络系统。另外，节点的通信距离有限使得网络中的数据传输通常需要被许多中间节点转发。

上述这些特点使得其它网络中的路由策略无法被直接应用到传感器网络中，必须考虑传感器网络的这些特点为其独立地制定路由策略。首先，与其它网络中路由只考虑延迟优化或者传输质量优化不同，传感器网络中路由必须考虑节省能量。第二，传感器网络不适合复杂的路由策略，也不适合需要大量缓存数据的路由策略。第三，传感器网络的路由策略必须考虑整个网络的负载平衡，避免造成个别节点的过早死亡从而影响整个网络的寿命。另外，传感器网络的路由策略还必须适应自组织网络的特点，数据传输的下一跳通常只能通过十分有限的信息来判断。

1.2.2 传感器网络的研究现状

目前，传感器网络的研究热点主要集中在操作系统、网络协议、节点同步和定位、网络拓扑控制、数据查询及处理以及网络安全等方面。

操作系统的研究目的是为传感器节点开发适合的操作系统（大多是嵌入式无内核的操作体统）。这其中最著名的是加州大学伯克利分校开发的TinyOS[27]系统，迄今已经被广泛的用于传感器网络的应用和研究中,成为业界标准之一。基于TinyOS开发的传感器网络模拟器TOSSIM[43]也被传感器网络的研究者广泛使用。除了的TinyOS以外，其它已经被开发的传感器网络的操作系统还有SOS[28]，MANTIS OS[29], liteOS[44]，RETOS[45]等等。

传感器网络协议的研究基本可以分为路由协议和MAC协议两个部分。路由协议负责根据传感器网络多跳自组织的特点为传输数据优化地建立传输路径，其基本内容将在 1.3 中进行介绍。MAC协议主要是为了制定合适的节点睡眠机制、避免信号碰撞、避免接收不必要的数据和避免不必要的侦听信道，从而达到为节点节省能量以及提高传输质量的目的。除了广泛用于低能耗无线网的IEEE 802.15.4[46]以外，还有S-MAC[47]、T-MAC[48]、X-MAC[49]、Z-MAC[50]、RI-MAC[51]、CMAC[52]和DW-MAC[53]等多种适合于传感器网络的MAC协议被提出。

节点的同步目的是使整个网络的节点达到时间上的一致从而能够更好的协同工作。例如，当我们测量车速时，需要计算汽车通过两个传感器的时间差。为了达到精确测量，我们需要两个节点的时钟保持高度一致。传感器网络中的同步算法要么是基于节点广播的时钟信息交换，要么基于层次结构的时钟信息交换，然后通过各种机制来消除传输过程中的时间不一致性。其中，比较著名的同步算法有RBS[54]、TPSN[55]、TINY-SYNC和MINI-SYNC[56]、GTSP[57]、Leakage-aware算法[58]、Routing-header-embedded算法[59]、Secure Time算法[60]等等。在大多数监测中，确定节点的准确位置是最基本的要求。例如，战场上的节点根据自己的位置报告敌人出现的准确位置。考虑到经济原因，应用者只能为网络中很少一部分节点，而不是所有节点，安装卫星定位系统，所以如何利用信标节点（已经确定位置的节点）为没有确定位置的节点准确定位就成了一个关键问题。现有技术一般会根据节点间的距离、角度、信号强度或传输延迟，采用三边测量法、三角测量法或极大似然估计法来为节点定位。主要的定位方法有TOA定位[61]、TDOA定位[62]、AOA定位[63]、RSSI定位[64]、质心定位[65]、APIT定位[66]、DV-Hop定位[67]等等。

传感器网络的拓扑控制目的是生成良好的拓扑结构，从而提高网络中数据传输的效率延长网络生命，为上层应用奠定基础。拓扑控制的主要思想是在满足网络覆盖率或连通度的前提下，通过控制节点的发送功率（从而控制节点的通信范围）和选择节点层次结构，删除多余的无线链接，生成高效的网络拓扑结构。现有的拓扑控制方法主要有基于功率控制的方法[68–70]、基于分簇的方法[71–73]和基于睡眠调度的方法[74, 75]。

传感器网络中的数据查询和处理技术包括数据管理、数据融合、查询优化等方面，其目的是高效的存储和管理网络中的数据、高效的执行用户的查询。与传统数据库中的强调优化I/O 次数和计算时间不同，传感器网络中的数据查询和处理主要目的是优化网络中的通信量从而为网络节省能量。目前，比较成型的传感器网络数据管理系统有TinyDB[76]和Cougar[77]系统。传感器网络中比较

流行的数据存储方法是以数据为中心的存储[78–81]，其基本思想把网络中相近似的数据存放到相同节点上，通过建立的索引，能够高效地执行用户的查询。数据融合技术[82–85]通过特定的结构，对网络中的多份数据进行综合、删除冗余数据、对相似数据进行聚集，从而能够提升网络中能量的利用效率。为了节省通信量，传感器网络中的查询处理[77, 86–92] 主要在网内进行，其主要思想是：将查询下发到网络中，相关节点根据查询上传数据，在上传的过程中进行查询优化，尽量减少传输的数据量。

在很多对抗性应用，尤其是军事应用中，传感器网络不可避免的面临着被侦听、被破坏、被篡改、被欺骗。针对这些攻击行为，如何保证网络内的数据，尤其是关键数据的安全性就成为了传感器网络研究的关键问题之一。常用的安全技术主要包括为保证安全传输的密钥建立与管理技术[93–96]、为发现攻击节点入侵的监测技术[97–99]、为了增加路由安全性的安全路由技术[100, 101]和为了应对网络欺骗的虚假数据过滤技术[102, 103]等。

1.3 传感器网络中的路由

1.3.1 传感器网络中路由的特点

网络中路由协议的主要任务是将源节点的数据沿着优化的路径发送给目的节点，其中寻找源节点和目的节点间的优化路径是路由研究的主要问题。传统的路由协议优化的目标主要是寻找源节点和目的节点间延迟最小的路径，同时最大化网络流量避免拥塞。能量消耗不是它们考虑的重点。传感器网络能量优先的特点要求所应用的路由协议必须以节省能量为主要目标，这就造成了传感器网络中的路由与其它网络中路由的显著差别，也是我们需要重点考虑传感器网络路由的主要原因。另外，传感器网络中的路由还必须适应其多跳自组织以及节点功能简单的特点。总结起来，传感器网络中的路由应该满足以下要求：

- 能量优先。根据传感器网络的特点，我们将优化能量作为路由的首要目标。与传统网络中路由的优化目标不同，传感器网络中的路由主要问题是寻找源节点和目的节点间能量消耗最小的路径，同时平衡整个网络的能量消耗。
- 基于局部拓扑信息。由于传感器网络是一个自组织网络而且拓扑结构随着节点的加入、死亡、移动而不断变化，所以获得整个网络的拓扑信息就变得不切实际。每个节点在路由过程中只能根据自己掌握的局部拓扑信息进行判断，选择下一跳节点。这就要求路由算法必须是基于局部拓

扑信息的分布式算法。

- 简单高效。传感器节点功能简单的特点使得我们不可能让其执行太复杂的计算或者缓存大量数据，这就要求传感器网络的路由协议在计算上要简单，在传输上要高效。
- 负载平衡。为了避免节点死亡引起的整个网络的瘫痪，网络中的数据传输因该避免过多使用某些节点从而造成它们的过早死亡，所以传感器网络中的路由还必须考虑网络的负载平衡。
- 可扩展性。传感器网络的规模不尽相同，节点数量可能达到成千上万甚至更多，这就要求路由具有良好的可扩展性。
- 适应拓扑变化。传感器网络中会频繁出现节点死亡、新节点加入、节点移动等现象，这就造成网络拓扑的频繁变化，所以路由协议必须能够动态调整以适应这些变化。

一般路由策略有两种机制，即预先设定机制和面向需求机制。所谓预先设定机制是指无论是否有路由需求，每个节点都保存一个路由表，记录了到网络中其它所有节点的路由（即传输路径的下一跳节点）。这种路由要求每个节点都掌握全局的拓扑信息。并且，当网络中发生任何拓扑变化时（例如链接失效、链接恢复、节点死亡、新节点加入等），必须将这种变化广播给网络中的所有节点。另外，无论是否有路由需求，每个节点必须保存到其它所有节点的路由。由于传感器网络资源有限和拓扑变化频繁的特点，使得这种路由机制不适合传感器网络。所谓面向需求机制是指网络不预先设定任何路由表，当网络中有路由需求时，网络动态的为该需求建立路径。这种按需路由的机制不需要收集全局的拓扑信息，适合传感器网络自组织的特点。并且，它可以很好的适应网络的拓扑变化。因此，传感器网络中的绝大多数路由都采取这种面向需求的机制。

基于以上原因，传感器网络中的路由与其它网络中的路由有很大的不同，需要研究者考虑其特点为其单独制定合适的路由策略。这就是为什么路由成为传感器网络研究的热点之一。

1.3.2 传感器网络和图的关系

在路由研究中，我们都会考虑网络的拓扑结构，也就是网络中节点和无线链接的关系。一般情况下，我们可以将传感器网络表示为其拓扑图$G = (V, E)$。其中，顶点（节点）集合为$V = \{v \mid v$ 是网络中的节点$\}$，边集合为$E = \{(u, v) \mid u$ 和v 之间存在无线链接$\}$。这样，传感器网络中的路由问题就可以被转化为图中

的路径问题。不过，我们需要注意的是，图论中的算法多是集中式算法，即在已知图的全部信息的情况下完成计算。而在传感器网络中，我们需要的是分布式方法，即每个节点都参与计算，并且这些计算只能根据节点所掌握的其周围的少量拓扑信息。

1.3.3 传感器网络中路由的研究现状

下面，我们分几大类来介绍传感器网络中有代表性的路由协议[104]。

1.3.3.1 按需路由

我们将传感器网络中最基本的路由策略都归为这一类，它们之中比较有代表性的有：

(1)洪泛路由[105]。这种最简单的路由策略来自与传统网络。当每个节点有数据要传输或者接收到数据时，就将该数据进行广播，直到数据过期或者到达目的地为止。这种方法保证了数据可以达到网络中的任何地方。虽然这是一种动态的路由协议，但是数据的冗余接收和转发会极大地消耗传感器网络的资源。

(2)Gossip路由[106]。这种路由可以看作是洪泛路由的改进。当节点有数据要传输或者接收到数据时，节点将该数据随机转发，直到数据到达目的地或者过期。这种方法虽然避免了数据的冗余接收，但是数据的无目的转发会造成传输的延迟过大以及节点能量的无谓消耗。

(3)AODV路由[107]。当数据源s想要发送数据给目的地t时，在传输数据之前有一个发现路径阶段，用来发现从s到t的路径。其基本思想是：由s节点开始，通过广播路径请求信息，建立一棵以s节点为根的生成树（包含所有节点），每个节点确认其在树中的父亲节点。当t节点加入到树中时，它沿着生成树发送一个路径确认消息给s节点。这样从s到t的路径上的每个节点都知道自己的上一跳和下一跳节点。

(4)DSR路由[108]。这种方法与AODV路由类似，同样是在路径发现阶段建立一棵以s节点为根的生成树。DSR与AODV的区别在于：在建立生成树时，每个节点将迄今为止发现的从s到其自身的路径包含在所广播的路径请求信息中。当t节点加入到树中时，它就获得了从s到t的路径的完整信息。这个路径信息将被作为包头放在要传递的数据中。

(5)能量优化路由[82, 109]。这类路由可以看作是AODV或者DSR的改进。与ADOV和DSR贪心式建立以数据源节点为根的生成树不同，这类路由建立的是一棵优化的生成树。通过生成树，每个节点都能找到从源节点到其自身的

或是能量消耗最小、或是可用能量最大、或是跳数最小的路径。

1.3.3.2 层次路由

这类路由是基于传感器网络的层次结构，例如簇结构，来传递数据。它们之中比较有代表性的有：

(1)LEACH路由[72]。该协议主要用于将整个网络的数据向基站节点进行聚集。首先将整个网络划分为多个簇，每个簇有一个簇头节点负责收集簇内的数据。当执行数据聚集操作时，每个节点通过一跳传输将数据传给所在簇的簇头。各个簇头节点再将簇内数据的聚集值通过一跳传输传给基站节点。这个协议的主要工作在于轮换性选择簇内节点做簇头以平衡能量消耗。它的缺点在于：由于簇头节点都必需经一跳传输和基站通信，所以该协议只适合应用于小型的传感器网络。而且，该协议只适合于数据聚集操作。

(2)TEEN路由[110]。该协议同样用于网络中的数据聚集。它采用与LEACH相同的分簇方式。与LEACH不同的是，TEEN没有限制簇头节点必须和基站一跳通信，而是根据簇头节点与基站的距离将它们组织为以基站节点为根的树形结构。当这些簇头节点沿着这棵树向基站传输数据时，完成数据的进一步聚集。

(3)HEED路由[71]。该协议也是为了网络中的数据聚集操作。与LEACH协议类似，数据先在簇内聚集，然后由簇头传送给基站。不同的是，簇头节点采取分布式的选取方式。每个节点根据其邻居和自身的剩余能量以及通信代价等信息计算其成为簇头的概率，通过多轮计算选出簇头。

1.3.3.3 面向查询的路由

这类路由主要用于回答用户的查询，也就是用户感兴趣的数据，如特定的温度、湿度等等。面向查询的路由中比较有代表性的有：

(1)定向扩散路由[111]。在这种路由中，查询由基站节点通过兴趣消息，采用洪泛的方式下发到整个网络中。在兴趣消息传播的过程中，每个节点建立其自身到基站节点的梯度（多数情况下为跳数）。在兴趣消息下发结束后，网络中所有包含查询相关数据的节点沿着梯度方向上报数据。在上报数据的过程中，从各相关节点到基站节点的路径被建立起来。这种路由适合需要节点连续上报数据的查询。另外，文献[112, 113]中的路由策略是对定向扩散路由的改进，主要是更优化的建立节点的梯度。

(2)谣传路由[114]。如果用户的查询只需要节点一次上报少量数据，定向扩散路由的代价（包括兴趣消息扩散和梯度建立）就太大了。谣传路由正是为了高效的应对这种查询。首先假设用户只可能对某些特殊事件的数据感兴趣，而且每个节点保存了一个包含所有特殊事件的列表。当某个节点监测到一个特殊事件时，无论用户是否会对此下发查询，该节点都将事件的数据通过一个代理

消息在网络中随机转发。每个转发节点保存代理消息的副本。代理消息有一定的生命周期，当被转发到一定次数时随机传递停止。如果基站节点获得用户的查询，它将查询消息也在网络中随机转发。这样代理消息和查询消息的传输路径就有很大的机会交叉于某个节点，于是查询就可以被这个节点保存的代理消息的副本所回答。

1.3.3.4 地理路由

这类路由协议用于每个节点都通过GPS定位装置或者定位技术确定了自己的地理位置的传感器网络。在这类网络中，路由的目的地不再是某个节点ID，而是某个地理坐标。距离这个地理坐标最近的节点就是数据传输的终点。在路由的过程中，节点可以根据自己和邻居的地理位置以及目的地的位置选择数据转发的下一跳。这类路由中比较有代表性的有：

(1)GPSR路由[115]。这是一个典型的地理路由协议。在这个协议中，数据包中包含了目的节点的地理位置信息，每个节点都知道自己和邻居的位置。路由过程中，每个节点都将数据包转发给距离目的地最近的邻居。有些情况下，某些中间节点会发现自己距离目的地比所有邻居都近，从而无法选择下一跳节点。这种情况被称为路由上的空洞。当探测到空洞时，GPSR采取右手法则沿着空洞周围传输来绕过空洞。

(2)GEAR路由[116]。这种路由用于将用户的查询消息向目标区域的传播，并收集目标区域的数据。这里，我们假设用户的查询都是针对某个区域的查询。首先，基站将查询沿着一条优化路径传送到目标区域内距离自己最近的节点。该节点作为查询区域内的“代理”，将查询以洪泛或者层次传递的方式下发到区域内的每个节点。这些节点将数据上报给代理节点。汇聚后的数据再由代理节点沿着反向路径传送给基站。其中，基站节点到目标区域的优化路径根据节点的位置和剩余能量计算得到。

(3)SPEED路由[117]。该协议的目的是建立一条从源节点到目的地的能够保证传输速度的高效路径。该协议同样采用地理贪心策略。假设每个节点都记录了相邻节点的位置和它们之间的传输速度。当数据包传递到某个节点时，根据应用，这个节点会计算一个传输速度的阀值。该节点将所有距离目的地比自己近的邻居作为下一跳的候选集合。它从候选集合中挑选到其链接的传输速度最大的节点最为下一跳。如果候选集合中所有链接的传输速度都小于阀值，则利用反馈机制重新选择路径。

1.3.3.5 可靠路由

由于传感器节点的不可靠性，造成传感器网络中会频繁出现链接失效、节点失效等导致路由失败的现象。为了保证数据能够到达目的地，各种可靠性路

由策略被被提出。这些策略的基本思想都是利用多条传输路径来保证通讯的可靠性。当某条路径失效后，可以用其它路径来替代它。还可以将数据的多个备份沿着不同路径传输来提高其到达目的地的概率。各种多路径路由策略将在1.4中给予详细介绍。

1.4 传感器网络中的多路径路由

1.4.1 多路径路由的动机

传感器网络中的多路径路由就是指在源节点和目的节点间建立多条用于数据传输的路径来满足用户的路由需求。这种多路径的策略可以在很大程度上提高数据传输的吞吐量、可靠性和网络的负载平衡，还可以使信息传输更加安全。

由于传感器网络中无线链接的带宽十分有限，要想在源节点和目的节点间高速传输数据，只用一条路径显然是不可能的，而多路径路由却可以完成这一任务。假设我们建立了连接源节点和目的节点的多条路径。通过将数据分别沿着这些路径进行传输，可以有效提高两个节点间数据传输的吞吐量。

由于传感器网络中节点功能有限而且容易收到攻击，这就造成了网络中频繁出现链接或节点失效的现象，从而造成路由失败。如果采用单一路径传输，一旦其上的节点或者链接发生错误，就会造成数据丢失，而多路径路由却能很好的应对这类情况。如果我们建立了连接源节点和目的节点的多条路径，当某条路径传输失败时，我们可以选择其它路径来完成传输。如果将数据备份成多份沿着不同的路径进行传输，还可以大大提高传输成功的概率。

在传感器网络中采用一条固定的路径大量传输数据，该路径上节点的能量很快就会被消耗殆尽，从而使网络寿命大大降低。多路径路由却可以很好的解决网络中的负载平衡问题。在建立了连接源节点和目的节点的多条路径之后，我们可以交替的利用这些路径来完成两个节点间的数据传输，使网络中节点的能量消耗更加平衡。

另外，在某些对抗性应用中，传感器网络中的数据传输很可能被敌方布置的间谍节点侦听。如果采用单一路径传输数据包，敌方很容易获得这些数据包。但是如果将敏感数据包拆分为多个部分，然后将它们沿着不同的路径传输，敌方获得整个数据包的可能性就大大降低了，整个网络的安全性会得到显著提升。

1.4.2 不同的多路径机制

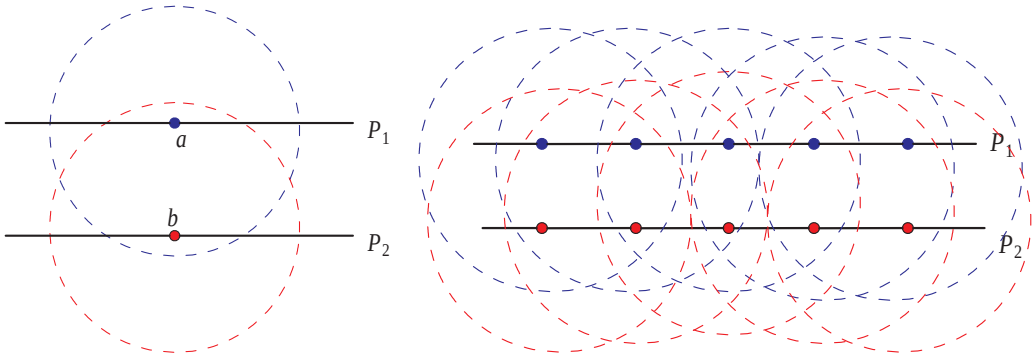
从以上多路径路由给传感器网络性能带来的改进可以看出，路径间独立性越大（耦合性越小），多路径路由带来的收益也就越大。根据路径间独立性不同，我们可以将传感器网络中的多路径机制分为伪多路径、链接不相交多路径、节点不相交多路径和非干扰多路径四种。

(1)伪多路径。这种机制允许多条路径拥有公共链接（边）或者公共节点。但是，为了保证多路径路由的优势，路径间的耦合程度不能过大，也就是任意两条路径间不能拥有过多的公共节点或者公共边。如何限定路径间的耦合程度是这种多路径机制的一个关键问题。如果不加任何限制，所找到的连接源节点和目的节点的多条路径有可能就是同一条路径。即使在限制路径耦合程度的情况下，也很可能出现所有路径都经过网络中的某些节点或者某些连接的现象。这种情况将造成多路径路由的优势丧失殆尽。例如，这些公共节点或者链接的带宽将会成为数据传输的瓶颈；这些公共节点的能量将会很快被消耗；如果这些公共节点或者链接失效，数据传输将会彻底失败。

(2)链接（边）不相交多路径。这种多路径机制是指任意两条路径间没有公共链接（边）。这种多路径机制保证了任意一条路径上的链接失效都不会对其它路径造成影响。但是，仍然会出现某些节点被网络中的大多数路径共享的现象。这种现象同样会影响多路径路由的性能。例如，这些公共节点的能量消耗将会比其它节点快的多；如果某些节点失效，数据传输将会失败；敌方只需布置少量节点就可以获得所有路径上的数据。

(3)节点不相交多路径。这种多路径机制是指除源节点和目的节点外，任意两条路径间没有公共节点。显然，节点不相交多路径一定是边不相交多路径。这种多路径机制可以极大的发挥多路径路由的优势，在提高数据传输的吞吐量、可靠性，提升网络的负载平衡和安全性方面都有很好的表现。

(4)非干扰多路径。这种多路径机制是指任意两条路径之间既没有公共节点也没有互为邻居的节点（两个节点不在彼此的通信范围内）。显然，非干扰多路径一定是节点不相交多路径，非干扰多路径机制是对路径独立性的进一步加强。对于有线网络来说，节点不相交多路径足够满足用户提升网络性能的需求。但是，对于传感器网络来说，节点不相交多路径机制仍然存在很多不足，这是因为没有考虑到网络中的无线干扰问题。由于多数传感器网络都使用单一信道，如果一个节点同时接收到两个或两个以上的无线信号，就会产生信号干扰从而使该节点不能正确地接收任何数据。假设我们有两条节点不相交的路径P_1和P_2，如图 1-5 a) 所示，P_1上的节点a和P_2上的节点b在彼此的通信



a) 两个节点间的无线干扰　　b) 多个节点间的无线干扰

a) Interference between two nodes　　b) Interference among multiple nodes

图 1-5 路径间的无线干扰(虚线圆圈表示节点通信半径)

Fig.1-5 Wireless interference between routes (dashed circles denote transmitting ranges of nodes)

范围内（即它们互为邻居节点）。当节点a接收路径P_1上的数据时，节点b就不能转发路径P_2上的数据，否则就会发生无线干扰。反之，当节点b接收路径P_2上的数据时，a也不能转发路径P_1上的数据。为了避免这种干扰，节点间只能通过MAC协议来抢占信道，没有抢到信道的节点只能退避一段时间等待当前传输结束。如图 1-5 b) 所示，如果P_1和P_2上有很多互为邻居的节点，它们之上的传输就会因为频繁的退避等待而大大降低传输速度，节点的能量也会因为竞争信道而被大量消耗。另一方面，还是在图 1-5 a) 的例子中，如果a是敌方布置的间谍节点，它不但可以获得路径P_1上的数据，还可以通过监听的方式获得节点b转发的路径P_2上的数据。因此，数据传输的安全性也会因为无线干扰而降低。为此，非干扰路径的概念被研究者提出[118, 119]。

如无特殊说明，下文中的“不相交路径”指“节点不相交路径”。通过以上描述，我们可以知道，对传感器网络的性能带来有效提升的主要是不相交多路径和非干扰多路径。因此，本文将这两种多路径机制作为研究的重点。

1.4.3 多路径路由的研究现状

本小节，我们将分几类问题来介绍传感器网络以及其它领域多路径路由的研究现状。由于边不相交路径问题可以转化为节点不相交路径问题，所以我们将前者划归到后者进行讨论。

1.4.3.1 两条不相交路径问题

很多研究工作考虑的是寻找连接源节点和目的节点的两条不相交路径问题。其中，一条路径作为传输的主要路径，另一条路径作为备用路径。当主要路径失效时，启动备用路径来保持数据传输的畅通。

文献[120]针对的问题是：给定网络中的一个固定的节点v_d（通常是基站节点），对于网络中其它每个节点（被称作普通节点）v，找到两条连接v和v_d的不相交路径。当然，这个问题的前提是给定网络的拓扑图是一个2连通图。对于普通节点v_1和v_2，假设我们已经找到两条连接v_1和v_d的不相交路径P_1和P_1'，两条连接v_2和v_d的不相交路径P_2和P_2'。又假设P_1和P_2相交于节点u，而且v_1和v_2分别沿着P_1和P_2向节点v_d传送数据。当有数据包被传递到节点u时，它需要判断这个数据包是来自v_1还是v_2，然后选择继续传递给P_1上的下一跳节点或者P_2上的下一跳节点。于是，每个路由中间节点必须根据数据源的不同而选择不同的下一跳节点，尽管这些路由的目的地都是v_d。这样，节点保存的路由表将会非常大。该文献作者希望每个中间节点只根据路由的目的地来选择下一跳节点。例如在上面例子中，如果路径P_1和P_2相交于节点u，则必须保证P_1和P_2从u到v_d的部分都是相同的。该文献提出的解决方法是通过对拓扑图的耳分解来构造每个节点到v_d的两条不相交路径。这样构造出来的路径将在路由上保持一致性，即路由的下一跳只会根据目的节点来区分。

文献[121]在图论领域研究了相关问题：在有向图$G = (V, E)$中，假设每个边$e \in E$都有一个长度$l(e)$。给定网络中的一个固定的源节点s，对于网络中的每个其它节点v，找到两条从s到v的总长度最短的不相交路径。这里要找到的是最优不相交路径，即长度之和最短的两条不相交路径。该文献给出了一个时间复杂性为$O(m \log_{(1+m/n)} n)$的集中式算法，其中$n = |V|, m = |E|$。文献[122]要在网络领域解决相同的问题，并给出了一个通信复杂性为$O(m + dn)$的分布式算法，其中d为网络拓扑图G的直径。算法的基本思想是：构造一颗以s节点为根的最优生成树，在树中每个节点到s的路径都是最短路径。之后，通过特殊变换重新定义每条边的长度。然后，通过修改后的Bellman-Ford算法来寻找新的图中s到每个节点的最短路径，与原有路径一起组成s到每个节点的两条长度之和最短的路径。

文献[123]致力于寻找连接源节点和目的节点的两条路径。其中一条作为主要路径，另一条作为备用路径。当主要路径发生问题时，可以启用备用路径来完成数据传输。通过两条路径公共节点的数量和路径长度的对比，量化了两条路径的独立程度。为了保证数据传输，备用路径和主要路径的差别不能太大。其作者希望所找到的两条路径独立性越高越好，长度差别越小越好。基于平衡树构造策略，作者提出了这个问题的两个算法。

文献[124]研究了在包含非对称节点的图中，寻找两条连接s和t的不相交路径问题，并给出了这个问题的一个高效算法。所谓非对称节点是指该节点的某些邻居不能通过该节点建立路径。例如节点b和c同为节点a的邻居，但是路

径$b \to a \to c$却并不存在。算法用到了最大流问题中增广路径的概念。所找到的两条不相交路径具有最小的长度之和。

文献[125]研究的问题是判断网络中是否存在连接每对节点的两条不相交路径，即网络的拓扑图是否是一个2连通图。为此，其作者提出了一个高效的分布式方法。在算法中，每个节点分布式的判定其临近区域的拓扑子图是否是一个2连通图。通过信息交换每个节点能够获知n跳范围内的拓扑信息，然后根据这些信息判断其n跳邻居中是否有割点。在假定网络的拓扑图是一个单位圆盘图（即每个节点的通信范围都是大小相同的圆）的情况下，可以保证答案是正确的。

前面有些工作考虑的是：在给定图中寻找连接给定节点对的两条不相交路径，使得它们的长度之和最小。这是一个P问题。文献[126]考虑的是：在给定图中寻找连接给定节点对的两条不相交路径，使得其中较长的一条的长度最小。该文献证明了无论给定的图是有向图还是无向图，这个问题都是一个NP难的问题。其作者还给出了针对这个问题的一个启发式算法。

1.4.3.2 k不相交路径问题

这类问题研究的是寻找给定节点s和t之间的k条不相交路径。其中，k是用户指定的正整数。

文献[127]考虑了两种情况的多路径路由。一种是k条路径都是不相交路径，一种是这k条路径是部分不相交路径，也就是伪多路径。其重点工作是比较这两种多路径机制在路由中，尤其是节点失效情况下的表现。关于寻找k条不相交路径，其所用的方法是找到当前最短的路径，从网络中删除其中间节点，再找下一条最短的路径，直至找到k条不相交路径。

文献[128]研究的是在节点已知地理位置的情况下，如何寻找k条不相交路径。其基本思想是：s节点以地理贪心的方式同时发送多个建立路径消息。接收到建立路径消息并且没有被其它路径占据的节点回应路径确认消息。如果该节点已经被其它路径占据，则回应路径拒绝消息，以便上一跳节点重新选择路径。另一方面，该文献还制定了一个策略来绕过路由中的空洞。文献[129]考虑的同样是在节点已知地理位置的情况下寻找k条不相交路径问题。其方法的基本思想是一条一条地寻找不相交路径，一旦一条路径被建立起来，路径将不再变化，其上节点不再参与其它路径。寻找路径时采取的是经过调整的地理贪心策略，在考虑节点距离的同时，考虑节点与目的节点和源节点的角度。

文献[130]假设节点可以对每次采集的数据按照过去的经验值给出一个估计。如果采集的数据与估计值差别过大，则表示附近有特殊事件发生。在没有特殊事件发生时，节点采用一条路径向基站传递数据。当有特殊事件发生时，

由于数据比较重要，节点采用多条不相交路径向基站节点传递数据。不相交路径的数量是根据节用户的可靠性需求（数据达到概率）、信道质量（差错率）以及路径长度来计算得出的。至于如何寻找多条不相交路径，该文献采取的是和文献[127]相同的方法。

文献[131]考虑的同样是根据用户的可靠性需求、信道质量以及路径长度来确定不相交路径的数目。与文献[130]不同的是，其不相交路径的数量是动态的。数据源节点根据自己和邻居节点到目的节点的距离以及周围信道的质量来将用户的可靠性需求分配给周围的邻居节点。下一跳节点将自己作为新的数据源节点，根据所分配的可靠性需求来继续寻找路径。路径的数量在不同距离上根据所分配的可靠性需求动态调整。

文献[132]修改了定向扩散路由[111]来建立源节点和目的节点间的k条不相交路径。正如定向扩散路由一样，每个节点在兴趣消息扩散阶段建立其到目的节点梯度。在上报数据阶段，沿着梯度的反向，建立多条不相交路径。建立路径的准则依旧是先到先占。

文献[133]在AODV协议的基础上进行了改动，提出了AODVM+协议，用来寻找源节点s和目的节点t之间的多条不相交路径。与AODV协议不同的是：在s节点将路径请求信息洪泛到整个网络的阶段，要建立的不是一颗以s节点为根的生成树。每个节点要记录的并不是唯一的父亲节点，而是多个可能作为其上一跳（距s节点更近）的节点。洪泛完成后，从t到s方向回溯多条路径。回溯的过程中，每个节点在多个可能的上一跳节点中选择一个没有被其它路径占据的节点作为其上一跳节点。

文献[134]要解决的问题是给定网络中的一个固定节点s，寻找每个节点到s的多条不相交路径，并且这些路径中包含了该节点到s的最短路径。该文献提出了关于这个问题的一个分布式算法。算法基于建立的以s节点为根的最短生成树，通过在树中交换少量信息并控制信息的传播方向，为每个节点建立多条到s的不相交路径。

文献[135]针对的问题是并行计算背景下，在验证一个图是否是k连通图。其中的一个子问题是寻找给定节点s和t之间的k条不相交路径。其采用的方法是逐条的寻找连接s和t的路径。该文献定义了增广路径的概念，类似最大流算法中的增广路径。在找到n条路径之后，通过寻找增广路径，并将增广路径添加到已找到的路径上，可以产生$n+1$条不相交路径。文献[136]在此基础了做了改进，提出了优化地寻找增广路径的方法。在找到n条路径之后，通过对图进行更有效的变换，去除了一些理论上不必要的子图，使得增广路径的寻找更加简单。

1.4.3.3 最优k不相交路径问题

假设网络中的每个连接（边）都有一个长度来表示其代价，如能量代价或者传输延迟。最优k不相交路径问题是指：给定网络中的节点s和t，寻找连接s和t的k条不相交路径，使得这些路径在所有可能解中长度之和最小。

文献[137]给出了关于这个问题的一个集中式算法。算法可以保证答案的正确性，即如果网络中存在k条不相交路径，算法一定能够找出k条不相交路径。此外，算法还可以保证所找到的k条路径的长度之和最小。算法的基本思想是：通过对图的变换操作，将不相交路径问题转化为网络中的最大流问题。然后运用Ford-Fulkerson算法[138]来逐条寻找不相交路径。

文献[139]针对的是同样的问题，所提出的方法是在文献[137] 基础上的改进。通过循环地寻找s和t之间的最短路径，来建立k条不相交路径。每次找到最短路径之后，对图做一定的变换（包括分离节点、改变边的权值等等），使得所找到的k条路径一定是长度之和最小的k条路径。其所用的寻找最短路径的方法为改进的Dijkstra[140]算法，可以处理权值为负的边。

文献[141]在网络领域考虑寻找连接s和t的能量代价之和最小的k条不相交路径。该文献提出这些路径的能量代价之和并不等于k条路径上边的权值之和，所以这个问题略微不同于长度之和最小的k不相交路径问题。这是因为在第一跳传输时，s节点可以通过广播的方式向所有的下一跳节点同时传输数据。所以，路径的能量代价之和应该等于所有路径中第一条边权值的最大值加上其它所有边的权值。该文献所提出的方法除了优化寻找第一跳节点外，其它部分运用了文献[137, 139]中的方法。

1.4.3.4 多代价不相交路径问题

所谓多代价不相交路径问题是指：我们对找到的k条路径按找到的顺序进行编号。对于网络中的每条边e，当其位于第1条、第2条、...、第k条路径中时，它有k个不同的代价。这样的网络被称作多代价网络。在多代价网络中，给定节点对s和t，寻找代价之和最小的k条连接s和t的不相交路径问题被称作多代价不相交路径问题。

文献[142]证明了即使当$k = 2$的时候，无论需要的路径是节点不相交路径还是边不相交路径，多代价不相交路径问题都是一个NP完全问题。当$k = 2$并且所要的路径是边不相交路径时，文献[143]给出了这个问题的一个精确算法。当然，算法的在最坏情况下的时间复杂性可能达到指数级别。当k取任意正整数时，文献[144]给出了这个问题的一个启发式算法，名字叫做k-penalty。算法利用文献[139]中的思想。当网络是一个单代价网络时（即边位于每条路径的代价相同），k-penalty与文献[139] 中的算法表现相当。文献[145]在k-penalty的基础

上进行了改进，提出了KPI算法。KPI算法增量的寻找路径。每确定一条路径之后，就修改相关边的权值，然后优化地寻找下一条路径。

1.4.3.5 长度受限的不相交路径问题

长度受限的不相交路径问题是指：给定图中两个节点s和t，以及一个路径长度限制L，找到最多的连接s和t的长度小于等于L的不相交路径。

文献[146]证明了当$L \geq 5$时，这个问题的判定问题是一个NP完全问题。文献[147]进一步证明了当$L \geq 5$时，这个问题的判定问题是一个APX完全问题，也就是没有关于这个问题的多项式时间近似模式。这意味着我们不可能以任意的近似比来近似这个问题。所以为这个问题设计可分析的近似算法是非常困难的。迄今为止，关于这个问题，只有一个启发式算法在文献[148]被提出。算法逐条的寻找连接s和t的长度小于等于L的路径并删除其上节点。如果找不到长度小于等于L的路径，算法尝试破坏某条已发现的路径P_i来产生一条新的长度小于等于L的路径P_i'，但必须保证P_i'的长度大于P_i的长度。这样直到产生不了任何新的路径为止。这个算法可以保证当$L \leq 4$时，算法输出的是最优解。

以上的问题是在图中的每条边长度都是1的情况下定义的。如果网络中的每条边都有一个可以取任意非负数的长度，其拓扑图就成为了一个加权图。如果给定的图是一个有向加权图，并且需要的路径是边不相交路径，那么这个问题就成为了有向加权图中长度受限的边不相交路径问题。文献[149]证明了对于任意的$\epsilon > 0$，以$m^{1/2-\epsilon}$来近似有向加权图中长度受限的边不相交路径问题是NP难的。此外，该文献还给出了这个问题的一个$\sqrt{m}$近似算法。算法的基本思想是逐条寻找长度小于等于L但是其上边数最少的$s \sim t$路径，并将路径的边从网络中删除。

1.4.3.6 广义边不相交路径问题

这个问题是对不相交路径问题的扩展，可以描述为：给定k个节点对$\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_k, t_k\}$，用边不相交路径连接最多的节点对。如果我们需要的是节点不相交路径，可以通过对拓扑图做简单的变换将其转换为寻找边不相交路径。如果这k个节点对都相同，这个问题就变成了k不相交路径问题。

广义边不相交路径问题是图论中的一个经典问题。文献[150]中对这个问题的背景做了很好的介绍。无论给定的图是有向图还是无向图，这个问题都是一个NP难问题[151]。即使我们限定给定的图是一个可平面图，这个问题依旧是NP难问题[152]。甚至我们将给定图限定为一个网格图（节点成棋盘状排列），这个问题仍然是一个NP难问题[153]。文献[154]证明了在无向图中，如果给定的参数k是一个常数，这个问题有多项式时间算法。在有向图中，即使$k = 2$，这个问题也是一个NP难问题[155]。

对于有向图中的广义边不相交路径问题，文献[149]证明了除非$P = NP$，否则对于任意的$\epsilon > 0$，这个问题不能被$m^{1/2-\epsilon}$近似。这个证明基于在有向图中，即使$k = 2$，广义边不相交路径问题也是一个NP难问题。而这个结论在无向图中并不成立，因此这个结论不适用于无向图。关于无向图中的广义边不相交路径问题，文献[156]证明了除非$NP \subseteq ZPTIME(n^{poly\log n})$，否则对于任意的$\epsilon > 0$，这个问题不能被$\log^{1/2-\epsilon} n$近似。

关于无向图中的广义边不相交路径问题，文献[157]给出了一个近似比为$O(\sqrt{m})$的近似算法。其中，$m = |E|$为图中边的数量。还是对于这个问题，文献[150]给出了一个近似比为$O(d')$的在线算法。其中，$d' = \max\{D_G, \sqrt{m}\}$，$D_G$为给定图$G$的直径。文献[158, 159]给出了这个问题的近似比为$O(\sqrt{n})$的近似算法，这是现今已知的最好的近似算法。其中，$n = |V|$为给定图中节点数量。

当给定的图是网格图时，文献[150]给出了这个问题的近似比为常数的算法。该文献进一步将这个结果扩展到密集嵌入图中，这种图可以看作是网格图在各种平面上的扩展。当给定的图是一个欧几里德可平面图，并且所有的给定节点对都位于图中的外部面时，文献[160]给出广义边不相交路径问题的线性时间的精确解。当给定的图是一个可平面图，并且允许最多有2条路径共享同一条边时，文献[152]给出了这个问题的近似比为$O(\log n)$的算法。当给定的图是一个可平面图，并且最多允许4条路径共享同一条边时，文献[161]给出了这个问题的常数近似比算法。

1.4.3.7 非干扰路径问题

所谓的非干扰路径是指任意两条路径之间既没有公共节点也没有互为邻居的节点（两个节点不在彼此的通信范围内）。路径间的无线干扰被定义为不同路径上的节点在彼此的通信范围内（互为邻居节点），这样它们的信号会对彼此产生干扰。无线干扰给传感器网络带来的害处在文献[162]中有详尽的阐述。我们将非干扰路径问题分为$s \sim t$非干扰路径问题和广义非干扰路径问题。

所谓$s \sim t$非干扰路径问题是指：给定节点s和t，找出连接s和t的k条非干扰路径。这里，不考虑路径在s和t周围发生的干扰，即任意两条路径的生成子图都是一个包含s和t的无弦的环。关于这个问题现有的研究只考虑了$k = 2$的情况。这个问题即使在$k = 2$的时候也是一个NP完全问题[163]。

文献[119]假设网络中节点的通信范围都是大小相等的圆（即网络是一个单位圆盘图），每个节点都知道自己的地理坐标，节点分布均匀并且密度已知。在此基础上该文献给出了一个算法寻找连接s和t的两条非干扰路径。算法的基本思想是沿着s和t的连线设置一个宽度为节点通信半径的禁入区域。在禁入区域两侧沿着区域边界寻找两条连接s和t的路径。这样可以保证两条路径上的节

点都不在彼此的通信范围内。算法有一个可分析的概率可以确保找到两条非干扰路径。文献[164] 量化地分析了在这种网络中，无线干扰对路由的吞吐量、传输延迟等方面的影响。

文献[165, 166]分析了路径间的无线干扰所造成的危害，以量化的方法定义了路径之间的耦合程度。在此基础上，该文献分析得出当网络中的节点配备有向天线时（即节点的信号可以以某个角度进行扩散），减小路径间的耦合程度要容易的多。在节点配备有向天线的网络中，文献[167]给出了一个可以找到两条非干扰路径的方法。其基本思想于文献[119]的方法相似。

文献[168]提出，一般情况下，节点信号的干扰范围会超出其信号的传输范围（一般为传输范围的2倍）。在这种情况下，该文献提出了新的模型来量化路径间的耦合程度。在此基础上，提出了一种简单的方法来寻找连接源节点和目的节点的3条耦合程度尽量小的路径。算法的基本思想是采取贪心的策略一条一条地建立路径。还是基于这种节点信号的干扰范围是其传输范围2倍的假设，文献[169, 170]给出了算法用于在节点已知其地理坐标的网络中，寻找2条非干扰路径。算法的基本思想与文献[119]中建立禁入区域的思想相同。

所谓广义非干扰路径问题是指：给定网络（图）中的节点对$\{s_1,t_1\},\{s_2,t_2\},\ldots,\{s_k,t_k\}$，用非干扰路径连接最多的节点对。当这$k$个节点对都相同时，这个问题就成为了$s \sim t$非干扰路径问题。显然，这个问题是NP难的。这个问题在图论领域中的研究比较多。在图论领域，非干扰路径又被称作生成不相交路径，意为路径的集合就是其上节点的生成子图，即路径间没有边。文献[118, 171]证明了：（i）当k是一个常数并且给定的图是一个有向可平面图时，这个问题有多项式时间的解。（ii）当k是一个常数并且给定图是无向可平面图时，这个问题有线性时间的解。

1.4.3.8 现有工作的不足

现有的在传感器网络中建立多路径的方法有各种各样的不足。有些是因为算法效率和计算准确性之间的矛盾，有些则是因为是新的问题很少被研究。

对于k不相交路径问题，现有研究工作的不足在于：现有的适合于传感器网络的分布式处理方法都不能保证找到网络中最多的不相交路径；现有的能够找到网络中最多不相交路径的方法都是集中式方法，需要收集整个网络的拓扑信息，从而不适合应用在传感器网络中。

对于最优k不相交路径问题，现有研究工作的不足在于：现有的适合于传感器网络的分布式方法都既不能保证找到网络中最多的不相交路径也不能保证输出最优结果；现有的能够找到最多路径和输出最优结果的方法都是集中式的方法，需要收集整个网络的拓扑信息，从而不适合传感器网络。

对于长度受限的k不相交路径问题，现有研究工作的不足在于：没有可分析的近似算法被提出；也没有适合于传感器网络的分布式处理方法被提出。

对于广义非干扰路径问题，现有研究工作的不足在于：没有关于这个问题近似难度的结果；没有可分析的近似算法被提出；也没有适合于传感器网络的分布式处理方法被提出。

1.5 本文的主要研究内容

通过以上介绍，我们可以看到传感器网络中的多路径路由，特别是利用多条不相交路径或者非干扰路径进行路由具有很重要的理论和实际意义。尽管已有研究工作考虑过这些问题，但是这些研究工作或者是在图论领域被提出，或者是针对其它网络被提出，而没有考虑传感器网络的特点。即使是传感器网络领域的相关工作，也存在这各种各样的不足。于是，本文重点研究了在传感器网络中利用不相交多路径和非干扰多路径进行路由的问题，其主要内容包含以下几个方面：

（1）k不相交路径问题。也就是给定网络中的节点s和t，寻找k条连接s和t的不相交路径的问题。对比于单路径路由，利用多条不相交路径进行路由可以在吞吐量、可靠性、负载平衡和安全性方面大幅提升网络性能。其中最基本的问题就是k不相交路径问题。关于这个问题在网络领域和图论领域已有很多相关研究。在网络领域提出的方法虽然采取了适合于传感器网络的分布式处理方式，从而不用收集全局拓扑信息，但是这些方法都不能保证答案正确性。所谓答案正确性是指当网络中存在k条不相交路径的情况下，算法一定能够输出k条不相交路径，即算法能够找到最多的不相交路径。在图论领域提出的方法虽然能够保证答案正确性，但是由于采取了集中式的处理方式，要求我们必须收集整个网络的拓扑信息，从而不适合传感器网络。

关于这个问题，本文提出了一个分布式算法EDA逐条寻找不相交路径。EDA采取分布式处理方式，通过网络中的部分节点少量交换信息，使得现有路径上的节点获取一些连接信息，根据这些连接信息可以生成更多的不相交路径。EDA可以保证答案正确性。据我们所知，它是现今仅有的不需要收集全局拓扑信息就可以保证答案正确性的算法。实验结果验证了EDA在传感器网络中的效率。无论是在找到的路径数量方面还是在通信代价方面，EDA都有很优秀的表现。

（2）最优k不相交路径问题。这个问题考虑网络中每个连接都有相应的权值来表示其延迟或者能量消耗，即网络的拓扑图是一个加权图。问题定义为给

定网络中的节点s和t，寻找k条连接s和t的不相交路径，使得这些路径在所有的可能结果中长度之和最小。对比于简单的k不相交路径问题，这个问题考虑到了对路径的优化，从而使得产生的结果更加适合于路由。这个问题的研究难点在于要求算法既能保证答案正确性又能保证结果最优性。所谓“结果最优性”是指输出的k条不相交路径在所有可能结果中长度之和最小。现有的在网络中寻找不相交路径的方法都不能满足这两点要求。虽然，图论领域的相关方法可以产生保证答案正确性和结果最优性，但是其集中式的处理方式不适合传感器网络。

关于这个问题，本文提出了一个完全的分布式算法OFDP，逐条地产生不相交路径。OFDP采用完全分布式的处理方式。当找到n条路径之后，通过在网络中的节点间交换少量数据，获得连接s和t的优化增广路径。通过将这条增广路径添加到现有路径上，可以产生最优的$n+1$条不相交路径。OFDP不需要从网络中收集任何信息，节点间的信息交换也不依赖任何结构。OFDP既能保证答案正确性又能保证结果最优性。据我们所知，OFDP是现今唯一可以满足这两个要求又不用收集拓扑信息的方法。实验结果表明了OFDP在传感器网络中的高效，找到的路径数量多并且通信代价小。

在OFDP的基础上，我们又提出了一个关于k不相交路径问题（不用优化路径长度）的分布式算法FDP，FDP能够保证答案正确性并拥有很高的通信效率。

（3）长度受限的不相交路径问题。问题定义为给定网络中的节点s和t，尽量多地寻找连接s和t 的不相交路径，使得每条路径的长度都不大于用户指定的阀值L。k不相交路径问题或者最优k不相交路径问题产生的不相交路径中可能有些长度很大的路径。路径过大的长度意味着过大的传输延迟或者能量消耗，从而使这些路径不适合传输数据。另外，由于网络中每个链接都有一定的丢包率，如果路径长度过大，沿着该路径传输的数据到达目的地的概率就会很小，从而使得找到的路径意义不大。因此，用户在很多情况下会限制所找到的路径长度。这个问题不但是NP难而且是APX完全的，也就是不存在关于这个问题的多项式时间近似模式。现今只有一个关于这个问题的启发式方法在图论领域中被提出。由于其集中式的处理方式，使得这个方法不适合传感器网络。还没有关于这个问题的可分析的近似算法。

首先，本文提出了关于这个问题的一个分布式近似算法GDA，GDA的近似比为$\sqrt{n}$。其中n为网络中节点的数量。此外，本文还提出了一个分布式的启发算法OPDA。从s节点和t节点开始，分别建立两棵以这两个节点为根的树。通过延迟控制等方式，保证两棵树的平衡性。在两棵树交叉的节点上，记录两棵树

的交叉信息。然后收集这些交叉信息，并采取优化方法产生多条长度受限的不相交路径。实验结果验证了OPDA在找到的路径数量和通信效率方面的优异表现。

（4）广义非干扰路径问题。也就是给定网络中的节点对$\{s_1,t_1\},\{s_2,t_2\},\ldots,\{s_k,t_k\}$，用非干扰路径连接最多的节点对。由于不相交路径没有考虑到传感器网络中的无线干扰问题，相比于不相交路径，利用非干扰路径路由在吞吐量、负载平衡和安全性方面就有了进一步的提升。因此，这个问题在传感器网络领域具有很重要的实际意义。迄今为止，关于这个问题我们只知道它是一个NP难问题，当k是一个常数并且网络的拓扑图是可平面图时这个问题有多项式时间的解。

关于这个问题，本文证明了：i) 即使当$k=2$时，这个问题也是NP难的；ii) 对于任意的k和$\epsilon>0$，除非P=NP，否则这个问题不能被$m^{1/2-\epsilon}$近似。其中，$m=|E|$为网络拓扑图中边（也就是网络中的链接）的数量。

本文给出了一个基于贪心策略的算法GSPW。GSPW的近似比为$\sqrt{m}$，也就是理论上最优的近似算法。另外，本文还给出了这个问题的一个在线算法OSBPW。OSBPW具有很好的理论下界。

1.6 本文章节安排

本文在第2章研究了k不相交路径问题；在第3章研究了最优k不相交路径问题；在第4章研究了长度受限的不相交路径问题；在第5章研究了广义非干扰路径问题。最后，本文对所做工作进行了总结并指出了进一步要研究的问题。

第 2 章 无线传感器网络中的k不相交多路径路由

2.1 引言

一个典型的传感器网络是由大量部署在监测区域内的小型、电池供电的传感器节点组成的。由于传感器节点采用无线方式进行通信并且通信距离十分有限，在传感器网络中的数据传输通常需要由多个中间节点进行转发，这就使得如何设计高效的路由策略成为传感器网络研究的关键问题。近年来，采用多条路径，特别是多条（节点）不相交路径进行路由的策略得到了越来越广泛的研究[123–125, 127–134]。对于网络中的两个节点s和t，一条连接s和t的路径被称作一条$s \sim t$路径。多条$s \sim t$路径被称作不相交路径如果它们之中的任意两条不共享任何的中间节点。本章研究的问题可以表示如下：

k不相交路径问题：给定网络中的两个节点s和t以及一个用户指定的正整数k，寻找k条不相交的$s \sim t$路径。

相对于单路径路由，采用多条不相交路径进行路由可以有效地改善路由的表现。在两个节点间利用多条不相交路径并行传输多个数据包，可以有效地增加它们之间通信的吞吐量；由于传感器网络中节点和链接经常失效或发生错误，我们可以将数据备份成多份并沿着连接两个节点的多条不相交路径进行传输。这样，当某些路径上的传输失败时，数据仍有很大的机会到达目的地，传输的可靠性得到了大大的增强；如果两个节点使用一条路径大量传输数据，这条路径上节点的能量将会被很快地用光。如果两个节点交替使用多条不相交路径传输数据，网络中的能量消耗将会更加平衡，网络的寿命将会被延长；另一方面，在对抗性应用中，如果采用一条路径传输数据，数据很容易被敌方布置的间谍节点侦听。如果我们将敏感数据拆分为多个部分，然后将这些部分分别沿着多条不相交路径进行传输，敌方获得整个数据包的可能性就大大降低了，网络中信息的安全性会得到显著提升。

针对k不相交路径问题，本章提出了一个分布式算法EDA，增量地寻找连接s和t的k条不相交路径。EDA可以保证答案正确性。所谓“答案正确性”是指如果网络中存在k条不相交$s \sim t$路径，算法一定会输出k条不相交$s \sim t$路径。也就是说EDA将会输出$\min\{k, k^*\}$条不相交$s \sim t$路径。这里假设网络中最多存在k^*条不相交$s \sim t$路径。EDA采用分布式的处理方法，只需要在网络的节点间交换少量短小的数据包，就可以找到k条不相交路径。迄今为止，所有可以保证答案正

确性的方法都是需要收集整个网络拓扑信息的集中式方法，所有的分布式处理方法都不能保证答案的正确性。据我们所知，EDA是现今仅有的不用收集整个网络拓扑信息就可以保证答案正确性的方法。模拟实验结果验证了EDA在找到路径数量和通信代价方面的高效。

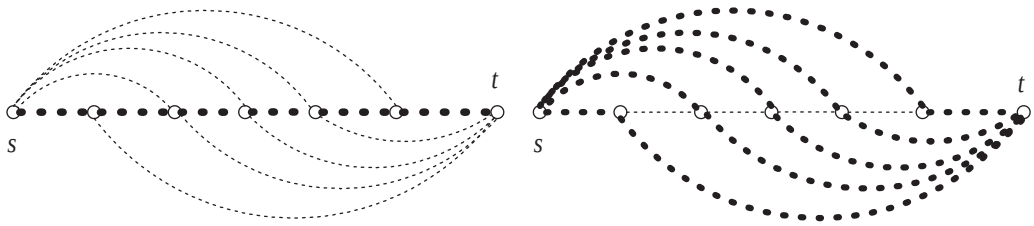
本章的内容安排如下：2.2介绍了一些相关工作；2.3给出了一些必要的预备知识；2.4描述了本文提出的分布式方法EDA；2.5证明了EDA可以保证答案正确性；2.6给出了实验结果来验证EDA的效率；最后，2.7总给全章。

2.2 相关工作

在本节，我们对相关工作及其优缺点进行简单的总结，更具体的内容已在1.4.3 中进行了介绍。

文献[120–125] 考虑了寻找两条连接目的节点和源节点的不相交路径问题。如果用这些方法解决k不相交路径问题，它们只能处理$k = 2$的情况。即使当$k = 2$时，它们中的大多数方法都不能保证答案正确性。

文献[127–134]给出了分布式算法用来寻找连接目的节点和源节点的k条不相交路径。这些算法都有着相同的思想：循环地寻找与已经找到的路径不相交的$s \sim t$路径；一旦找到一条路径，这条路径就被固定下来，其上节点不再参与其它路径的建立。这类方法固然简单且容易实现，但是存在一个严重的问题，即不能保证答案的正确性。假设用户要在一个如图 2-1 所示的传感器网络中寻找5条不相交$s \sim t$路径。利用这些文献中的方法，我们建立的第一条路径如图 2-1 a)所示。由于这条路径被固定下来，我们将找不到其它的与之不相交的路径，于是我们不得不报告网络中只有一条不相交路径。而实际上，如图 2-1 b) 所示，网络中确实存在5条不相交路径。由此可见如果不能保证答案正确性，我们将会丢失很多路径。



a) 第一条被找到的路径　　b) 网络中实际存在的5条不相交路径

a) The first found path　　b) 5 disjoint paths existing in the network

图 2-1 答案不正确的例子(虚线表示路径)

Fig. 2-1 An example of answer incorrectness (doted lines denote paths)

文献[135–137, 139, 141]在图论领域给出了集中式的方法用来寻找k条不相交$s \sim t$路径。这些方法都能保证答案正确性，也就是这些方法都能找到最多的连接s和t的不相交路径。但是，这些方法都是图论中的集中式方法，需要掌握全局的拓扑信息。如果我们将这些方法应用在传感器网络中，我们将不得不收集整个网络的拓扑信息，而这项操作的通信代价对传感器网络来说过于高昂。另外，我们还必须根据网络拓扑的变化来随时更新所掌握的拓扑信息。由节点死亡、新节点加入或链接失效所造成的网络拓扑变化在传感器网络中会频繁地发生，频繁更新全局拓扑会带来很高的能量消耗。

综合看来，现有的方法要么不能保证答案正确性，要么需要收集整个网络的拓扑信息从而通信代价过高。而本文提出的算法EDA在保证答案正确性的前提下采用高效的分布式的处理方式来寻找不相交路径，从而适合于传感器网络。

2.3 预备知识

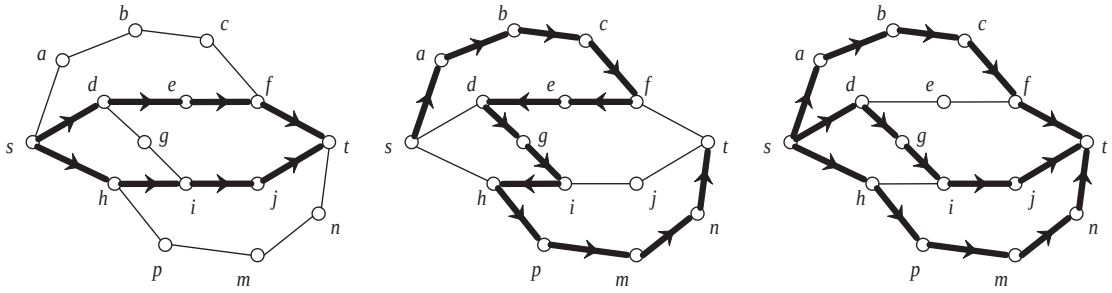
我们假设给定的传感器网络是静态的；网络中所有节点共用一个信道；所有的链接都是双向的，既如果节点u可以接收到节点v的信号，那么节点v一定可以接收到节点u的信号；网络是连通的，既给定任意两个节点，它们一定可以通过网络中的某条路径进行通信。我们可以将给定的网络用其拓扑图$G = (V, E)$表示。其中，顶点（节点）集合为$V = \{v \mid v \text{ 是网络中的节点}\}$，边集合为$E = \{(u, v) \mid u \text{ 和 } v \text{ 之间存在无线链接}\}$[1]。根据我们的假设，图$G$是一个无向连通图。对于其它的图$G'$，我们用$V(G')$表示其节点集合，用$E(G')$表示其边集合。

图G中的一条路径P是G的一个子图，可以表示为一系列不同的节点，即$P = (v_1, v_2, \ldots, v_n)$。其中，对于$i = 1, \ldots, n$有$v_i \in V$, 对于$i = 1, \ldots, n-1$有$(v_i, v_{i+1}) \in E$。这里，$v_1$和$v_n$是$P$的两端节点，$P$上的其它节点被称作中间节点。一条两端节点是$v_1$和$v_n$的路径被称作一条$v_1 \sim v_n$路径。我们用$v_i P v_j$（$1 \le i < j \le n$）来表示$P$的子路径$(v_i, \ldots, v_j)$，并称之为$P$的片段。需要注意的是每个$P$的片段至少包含一条$P$的边。

定义 2.1 k条$s \sim t$路径是不相交路径如果它们之中的任意两条不共享任何中间节点。

[1]在本文以后的部分，我们将不区分“节点”和“顶点”的概念，也不区分“链接”和“边”的概念。

定义 2.2 H是G的一个子图，图G中的一条路径B是相对于H的桥路径，如果：
1)B的所有中间节点属于$V \setminus V(H)$；
2)B的所有边属于$E \setminus E(H)$。



a) 已经找到的两条路径　　b) 增广路径　　c) 所产生的三条路径
a) The 2 found paths　　b) An augmenting path　　c) The generated 3 paths

图 2-2 增广路径的例子

Fig. 2-2 An example of augmenting path

假设我们已经在图G中找到了n条不相交路径$P_1, \ldots, P_n$。在本章以后的部分，“桥路径”表示图G中相对于H的桥路径，其中$H = P_1 \cup \cdots \cup P_n$。考虑下面的例子。在一个如图 2-2 所示的网络中，如图 2-2 a) 中粗线所示，我们已经找到了两条不相交$s \sim t$路径$P_1 = (s, d, e, f, t)$和$P_2 = (s, h, i, j, t)$。于是路径(s, a, b, c, f)、(d, g, i)和(h, p, m, n, t)就是网络中的桥路径。

定义 2.3 (**增广路径**) 一条增广路径P_a是一条$s \sim t$路径从s到t交替地经过：i) 桥路径；ii) 沿着从t到s 的方向某条已发现路径的片段。

这里，H仍然代表已经发现的n条不相交$s \sim t$路径构成的子图。通过将一条增广路径P_a“加”到H上，我们可以得到$n + 1$条不相交$s \sim t$路径。这里，“加”表示取P_a和H的对称差，即添加那些只属于P_a的边并删除那些P_a与H共享的边。

还是在图 2-2 所示的例子中，在发现如图 2-2 a) 所示的两条路径后，我们可以找到如图 2-2 b) 所示的增广路径$P_a = (s, a, b, c, f, e, d, g, i, h, p, m, n, t)$。从$s$到$t$，$P_a$交替地经过桥路径$(s, a, b, c, f)$；从$t$到$s$方向$P_1$的片段$(f, e, d)$；桥路径$(d, g, i)$；从$t$到$s$方向$P_2$的片段$(i, h)$；桥路径$(h, p, m, n, t)$。通过将$P_a$加到$H$上，我们可以得到如图 2-2 c) 所示的三条路径。

在以上对增广路径的描述中，路径的方向特别重要。为此，在本章以后的部分，我们将特别强调路径的方向。例如，我们将路径$P_1 = (v_1, v_2, v_3)$和$P_2 = (v_3, v_2, v_1)$看作是两条路径，尽管在原来的概念中它们表示的是同一条路径。其

中，P_1是一条从v_1到v_3的路径，P_2是一条从v_3到v_1的路径。除了路径表达式中节点的顺序以外，“从”和“到”也特别强调了路径的方向。对于每条$s \sim t$路径来说，我们定义它默认的方向是从s到t的方向。

定理 2.1 在发现n条不相交$s \sim t$路径$P_1, \ldots, P_n$之后，网络中存在更多的不相交$s \sim t$路径当且仅当存在一条增广路径。

这个定理的证明可以参见文献[135, 136]。定理 2.1 表明我们可以通过反复寻找增广路径的方法找到网络中最多的不相交$s \sim t$路径。

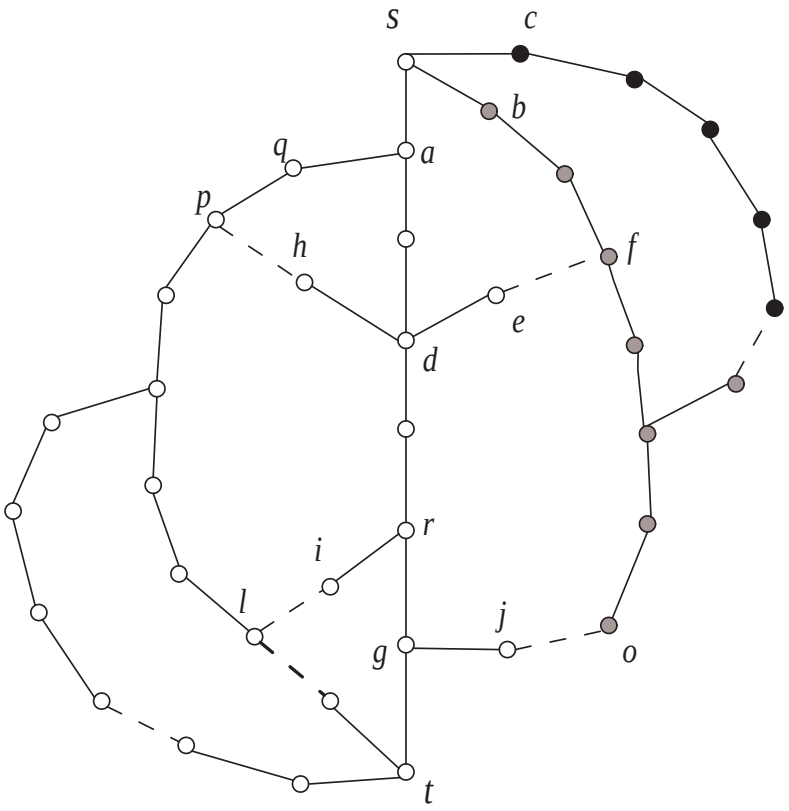


图 2-3 构建的生成树（实线表示树边，虚线表示非树边）

Fig.2-3 The constructed spanning tree (solid lines denote tree edges, dashed lines denote non-tree edges)

为了高效地寻找不相交路径，我们需要建立一棵以s节点为根的生成树T。其中，$V(T) = V$，$E(T) \subseteq E$。对于树中的任意两个节点u和v，uTv表示T中从u到v的路径。

对于任意节点v，$T(v)$表示以v为根的T的子树，这棵子树包含了v的所有子孙。让$c_1, \ldots, c_m$表示T中s 节点的孩子。对于每个属于$T(c_i)$的节点v，它的源就是c_i，用$origin(v)$表示。假设我们在给定的传感器网络中建立了一棵如图 2-3 所

示的生成树。在这棵树中，所有白色节点源都是节点a（既在子树$T(a)$中），所有灰色节点的源都是节点b，所有黑色节点的源都是节点c。

在已发现路径$P_1,\ldots,P_n$上的节点被称作路径节点，其它节点被称作非路径节点。对于已发现路径P_i上的节点v，如果在片段sP_iv上共有x条边，那么v就是P_i上第x跳节点，它的跳数是x。v的上一跳节点就是P_i上第$x-1$跳节点，v的下一跳节点就是P_i上第$x+1$跳节点。P_i的第一跳节点就是s节点在P_i上的邻居。

2.4 高效的分布式算法EDA

给定网络中的节点s和t，以及用户指定的正整数k，EDA采用分布式处理方式逐条寻找不相交$s \sim t$路径。EDA可以保证输出$\min\{k,k^*\}$条不相交$s \sim t$路径。其中，网络中最多存在k^*条不相交$s \sim t$路径。EDA有4个主要步骤，分别为：建立树；发现桥路径；收集桥路径信息；寻找增广路径。

Input: 网络中的两个节点s和t；正整数k
Output: $\min\{k,k^*\}$条不相交$s \sim t$路径
1 调用**建立树**算法;
2 将树中路径sTt标记为第一条发现的路径;
3 **repeat**
4 调用**发现桥路径**算法;
5 调用**收集桥路径信息**算法;
6 调用**寻找增广路径**算法;
7 **if** 不能找到增广路径 **then**
8 停止并输出已发现的路径;
9 **until** 找到k条不相交的$s \sim t$路径;

算法 2-1 EDA
Algo. 2-1 EDA

在第一步（建立树）中，我们建立一棵以s节点为根的生成树T。树中从s到t的路径sTt被看作是所找到的第一条路径。在第二步（发现桥路径）中，通过在树中的部分节点间交换信息，每个路径节点都可以发现一些从其它路径节点到其自身的桥路径。在第三步（收集桥路径信息）中，沿着已发现的路径，s节点收集这些桥路径的信息。在第四步（寻找增广路径）中，s根据已发现的路径和所收集的桥路径的信息构建一个图G^*，并在G^*中搜索增广路径。

如果EDA在第四步中找到了一条增广路径，通过将增广路径添加到已发现

路径上，可以多得到一条不相交路径。之后，EDA进入第二步重新发现桥路径。整个循环被执行到EDA找到k条不相交路径或者EDA在第四步中找不到增广路径为止。整个算法由算法 2-1 中给出。

在算法描述中，“节点v广播一个消息”表示这个消息可以被v的所有邻居接收到，正如传感器网络中的广播。“节点v发送一个消息给节点u”表示这个消息只能被u接收到，而v的其它邻居接收不到这个消息，正如传感器网络中的单播。

2.4.1 建立树

在这一步，我们建立一棵以s节点为根的生成树T。在建立树的过程中，每个节点v发现：i) 它在树中的父亲$parent(v)$；ii) 它的源$origin(v)$；iii) 它的邻居节点和这些邻居节点的源。

```
   Input: 传感器网络和其中的节点s
   Output: 一棵以s为根的生成树
   /* 对于网络中每个节点v的伪代码                                    */
 1 if v是s节点 then
 2     广播BUILD-TREE消息{φ};
 3 while 接收到节点u发送的BUILD-TREE消息{origin(u)} do
 4     if 这是v ≠ s第一次接收到BUILD-TREE消息 then
 5         parent(v) ← u;
 6         if u = s then
 7             origin(v) ← v;
 8         else
 9             origin(v) ← origin(u);
10         广播BUILD-TREE消息{origin(v)};
11     记录邻居节点的信息[u, origin(u)];
```

算法 2-2 建立树

Algo. 2-2 Building Tree

建立树的伪代码由算法 2-2给出。树的建立是通过网络中节点交换BUILD-TREE消息来完成的。每个BUILD-TREE消息都包含了发送者的源。首先，s节点广播一个BUILD-TREE消息来初始化整个过程（算法 2-2中的第1-2行）。对于其

它节点v，如果v第一次接收到BUILD-TREE消息，它就将消息的发送者设定为它在生成树中的父亲节点$parent(v)$（算法 2-2中的第4-5行）并正确地设定它的源$origin(v)$（算法 2-2中的第6-9行）。接下来，v广播一个包含$origin(v)$的BUILD-TREE消息（算法 2-2中的第10行）。节点v可以通过监听其它节点发送的BUILD-TREE消息知道其邻居节点的ID和这些邻居节点的源（算法 2-2中的第11行）。

在如图 2-3 所示的网络中建立生成树之后，节点e知道：它的父亲是节点d；它的源是节点a；它的邻居节点有d和f，它们的源分别是a和b。

在这一步的最后，生成树中从s到t的路径sTt被看作是找到的第一条路径。

2.4.2 发现桥路径

让$c_1, ..., c_n$表示已发现路径上的第一跳节点。所以，它们一定是s在树中的孩子节点。在这一步中，位于$T(c_1), \ldots, T(c_n)$中的节点交换信息来发现桥路径。

为了节省通信量，我们不要求发现网络中的所有桥路径，而只是去发现那些对寻找增广路径来说是必要的桥路径。让v表示$T(c_1), \ldots, T(c_n)$中的任意一个节点。对于每条已经发现的路径P_i，v只需要发现一条从u_i到v的桥路径，其中u_i是在集合$\{u \mid u \in P_i$并且存在从u到v的桥路径$\}$中跳数最小（与s节点最近）的节点。此外，如果v发现了一条从s到v的桥路径，它不用再发现任何其它的桥路径。对于每条已发现的路径P_i，v记录变量$bridge(v, P_i)$和$LBH(v, P_i)$。对于源节点s，v记录变量$bridge(v, s)$和$LBH(v, s)$。$bridge(v, P_i) = x$表示v发现了一条从u_i到v的桥路径，其中u_i是P_i上的第x跳节点。$LBH(v, P_i)$记录了这条从u_i到v的桥路径中v的上一跳节点。$bridge(v, s) = TRUE$表示v发现了一条从s到v的桥路径。$LBH(v, s)$记录了这条桥路径中v的上一跳节点。

对于$T(c_1), \ldots, T(c_n)$中的每个节点v，它根据以下规则发现桥路径：

规则1 如果v有一个邻居u不在$T(c_1), \ldots, T(c_n)$中，那么存在一条从s到v的桥路径，桥路径中v的上一跳节点是u。

规则2 如果v有一个邻居u是路径节点并且v不是u的上一跳或者下一跳节点，那么存在一条从u到v的桥路径，桥路径中v的上一跳节点是u。

规则3 如果v有一个邻居u是非路径节点并且存在一条从w到u的桥路径，那么也存在一条从w到v的桥路径，桥路径中v的上一跳节点是u。

根据上面的描述，为了使算法的通信效率更高，我们添加了两条规则：

规则4 如果v已经发现了一条从s到v的桥路径，它不需要再发现其它的桥路径。

规则5 P_i表示一条已发现的路径。如果v已经发现了一条从P_i上第x跳节点到v的桥路径，v不需要再发现任何从P_i上第y跳节点到v的桥路径，其中$y \geq x$。

Input: 一棵以s为根的生成树；已发现的不相交路径$P_1, \ldots, P_n$
Output: 网络中一些桥路径信息
/* 对于$T(c_1), ..., T(c_n)$中任意节点v的伪代码。$c_1, \ldots, c_n$为已发现路径上的第一跳节点。 */
1 **if** v有一个邻居w满足$origin(w) \neq c_1, ..., c_n$ **then**
2 $bridge(v, s) \leftarrow TRUE$;
3 $LBH(v, s) \leftarrow w$;
4 **if** v是一个非路径节点 **then**
5 广播一个FIND-BRIDGE消息$\{s, 0\}$;
6 **if** v是已发现路径P_i上的第x跳节点 **then**
7 给除了v的上一跳和下一跳节点之外的所有邻居节点发送FIND-BRIDGE消息$\{P_i, x\}$;
8 **while** 接收到了节点u发送的*FIND-BRIDGE*消息$\{Pid, Phops\}$ **do**
9 **if** $bridge(v, s) = TRUE$ **then**
10 忽略这个消息;
11 **else if** $Pid = s$ **then**
12 $bridge(v, s) \leftarrow TRUE$;
13 $LBH(v, s) \leftarrow u$;
14 删除所记录的其它桥路径的信息;
15 **if** v是一个非路径节点 **then**
16 广播一个FIND-BRIDGE消息$\{s, 0\}$;
17 **else**
/* 假设$Pid = P_i$ */
18 **if** $brdige(v, P_i)$不存在或者$bridge(v, P_i) > Phops$ **then**
19 $bridge(v, P_i) \leftarrow Phops$;
20 $LBH(v, P_i) \leftarrow u$;
21 **if** v是非路径节点 **then**
22 广播一个FIND-BRIDGE消息$\{P_i, bridge(v, P_i)\}$;

算法 2-3 发现桥路径
Algo. 2-3 Finding Bridge Paths

发现桥路径的伪代码由算法 2-3给出。其中，“P_i”表示已发现路径P_i的路径ID。网络中的桥路径是通过节点间交换FIND-BRIDGE消息来发现的。每个FIND-BRIDGE消息的格式是$\{Pid, Phops\}$。如果$Pid = P_i$，则表示这个消息的发送者已经发现了一条从P_i上第$Phops$跳节点到其自身的桥路径。如果$Pid = s$，则表示这个消息的发送者发现了一条从s到其自身的桥路径。

根据规则3，如果节点v是一个非路径节点并且发现了一条从s到v的桥路径，v广播一个FIND-BRIDGE消息$\{s, 0\}$（算法 2-3中第4-5、15-16行）。接收到这个消息之后，v的所有邻居节点都知道存在一条从s到其自身的桥路径，这些桥路径中它们的上一跳节点是v。根据规则2，如果v是已发现路径P_i上第x跳节点，v就向除了其上一跳和下一跳节点以外的所有邻居节点发送FIND-BRIDGE消息$\{P_i, x\}$（算法 2-3中第6-7行）。接收到了这个消息之后，v的这些邻居节点都知道存在一条从P_i上第x跳节点到其自身的桥路径，这些桥路径中它们的上一跳节点是v。还是根据规则3，如果v是一个非路径节点并且发现了一条从P_i上第x跳节点到其自身的桥路径，v就广播FIND-BRIDGE消息$\{P_i, x\}$（算法 2-3中第21-22行）。接收到这个消息之后，v的所有邻居节点都知道存在一条从P_i上第x跳节点到其自身的桥路径，这些桥路径中它们的上一跳节点是v。

在算法 2-3中，第1-5行实现了规则1，第9-16行实现了规则4，第17-22行实现了规则5。

2.4.3 收集桥路径信息

在这一步中，沿着发现的路径，s节点收集路径节点所记录的桥路径的信息。这些信息被放在COLLECT-BRIDGE消息中。对于一个路径节点v，当其接收到了其下一跳节点发送的COLLECT-BRIDGE消息后，v将自己发现的桥路径的信息添加到这个消息中，并将这个消息转发给它的上一跳节点。每个被收集的桥路径信息的格式是$[v, Pid, Phops]$，记录了一条从Pid上第$Phops$跳节点到节点v的桥路径。如果$Pid = s$，则记录了一条从s到v的桥路径。

为了调高通信效率，我们不需要收集路径节点发现的所有的桥路径信息，而是只收集那些对于搜索增广路径来说必要的桥路径的信息。为此，我们制定了下面的规则：

规则6 如果已经发现了一条从P_i上第x_1跳节点到P_j上第y_1跳节点的桥路径，那么就不需要再发现任何从P_i上第x_2跳节点到P_j上第y_2跳节点的桥路径，其中$x_1 \le x_2$并且$y_1 \ge y_2$。

Input: 已发现的不相交路径$P_1, \ldots, P_n$；路径上节点记录的桥路径信息
Output: s节点收集到一些桥路径信息
/* 对于每个路径节点v的伪代码 */
1 **if** v是目的节点t **then**
2 构建一个COLLECT-BRIDGE消息M;
3 **for** t记录的每个$bridge(t, Pid)$ **do**
4 将$[t, Pid, bridge(t, Pid)]$添加到M中;
5 将M发送给t的所有上一跳节点;
6 **while** 接收到下一跳节点发送的$COLLECT\text{-}BRIDGE$消息M **do**
7 **if** M包含$[u, s, 0]$ **then**
8 不向M添加任何信息;
9 **else if** $bridge(v, s) = TRUE$ **then**
10 将$[v, s, 0]$添加到M中;
11 **else**
12 **for** v记录的每个$bridge(v, P_i)$ **do**
13 **if** M不包含$[u, P_i, bridge(u, P_i)]$满足$bridge(u, P_i) \leq bridge(v, P_i)$ **then**
14 将$[v, P_i, bridge(v, P_i)]$添加到M中;
15 将M发送给v的上一跳节点;

算法 2-4 收集桥路径信息

Algo. 2-4 Collecting Bridge Path Information

收集桥路径信息的伪代码由算法 2-4给出。需要注意s是所有已发现路径上的第0跳节点。假设节点u和v在同一条路径上并且u的跳数大于v的跳数（节点u距离t更近）。根据规则6，如果已经发现从s到u的桥路径，那么我们不需要发现任何到节点v桥路径（算法 2-4中第7-8行）。当v接收到它的下一跳节点发送的COLLECT-BRIDGE消息M后，对于v记录的每个$bridge(v, P_i)$或者$bridge(v, s)$，如果根据规则6这些桥路径是有用的，v将这些桥路径的信息$[v, P_i, bridge(v, P_i)]$或者$[v, s, 0]$添加到M中，并将M发送给它的上一跳节点（算法 2-4中第7-15行）。

2.4.4 寻找增广路径

在这一步中，s根据已经发现的路径和所收集到的桥路径信息，构建一个图G^*，并在图中搜索增广路径。

Input: 已发现的不相交路径$P_1,\ldots,P_n$； s节点收集到的桥路径信息
Output: 更多的不相交路径$P'_1,\ldots,P'_{n+1}$
/* 对于节点s的伪代码 */
1 构建一个包含所有已发现路径的图G^*;
2 **for** 每个收集到的桥路径信息$[v,P_i,bridge(v,P_i)]$ **do**
  /* 让u表示路径P_i上第$bridge(v,P_i)$跳节点 */
3 在图G^*中添加有向边$< u,v >$表示从u到v的桥路径;
4 **if** 在G^*中存在增广路径P_a **then**
5 将P_a加到已发现路径上构建更多的不相交路径;
6 通过这些路径的信息回溯这些路径;

算法 2-5 寻找增广路径
Algo. 2-5 Finding Augmenting Path

寻找增广路径的伪代码由算法 2-5给出。最初，G^*只包含了所有已发现的路径（算法 2-5中第1行）。对于s接收到的每个桥路径信息$[v,P_i,bridge(v,P_i)]$，在G^*中添加有向边$< u,v >$用来表示从u到v的桥路径（算法 2-5中第2-3行），其中u是路径P_i上第$bridge(v,P_i)$跳节点。这里，我们省略了对以下这种情况的描述：对于s接收到的桥路径信息$[v,s,0]$，在G^*中添加一条有向边$< s,v >$用来表示从s到v的桥路径。在接收到所有的桥路径信息之后，s节点在图G^*中搜索增广路径（算法 2-5中第4行）。根据增广路径的定义，可以设计一个简单的算法来完成这个工作。我们将找到的增广路径添加到已发现路径上，会得到更多的不相交路径（算法 2-5中第5行）。

假设这些新得到的路径是$P_1,\ldots,P_{n+1}$。由于我们只是在G^*中发现了这些路径，而路径并没有在网络中被建立。因此接下来，我们需要根据这些新路径的信息来回溯（构建）这些路径。所谓路径的信息是包含了新路径上所有边和桥路径的信息，并且我们能够区分那些是边的信息那些是桥路径的信息。首先，s节点将这些新路径的信息发送给节点t，由t开始追溯这些路径。t将包含路径信息的TRACE-PATH消息发送给各个路径的上一跳节点。每个节点在接收到TRACE-PATH消息之后将自己标记为路径节点，然后在根据路径信息将TRACE-PATH消息转发给自己的上一跳节点，直到这些消息被传递到s节点。其中的难点在于根据路径上的桥路径信息来追溯网络中的桥路径。例如，我们知道某条路径包含了从节点u到节点v的桥路径，但是我们不知道这条桥路径的具体信息。

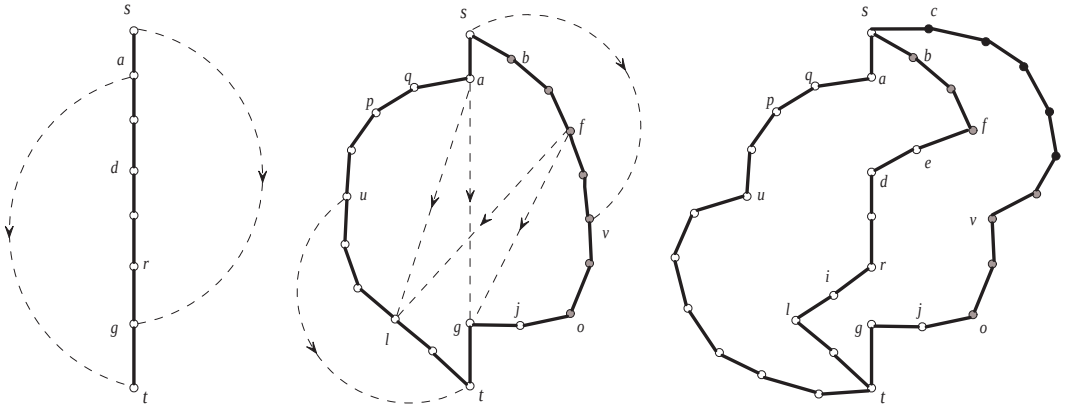
让我们简单描述一下追溯桥路径的机制。假设从节点v开始，我们要追溯一条从u到v的桥路径。其中，u是路径P_i上第$bridge(v, P_i)$跳节点。接下来，节点v就发送一个TRACE-PATH消息给节点$LBH(v, P_i)$，也就是v记录的这条桥路径中v的上一跳节点。每个接收到这个TRACE-PATH消息的节点都把消息转发给它所记录的$LBH(P_i)$。这个TRACE-PATH消息一直被转发到节点u。如果$u = s$，这个TRACE-PATH消息将会被转发到一个满足$origin(w) \neq c_1, \ldots, c_n$的节点$w$。这之后，这个消息将会沿着所建立的生成树从$w$被传送到节点$s$。

通过以上这些机制，网络中的路径就被建立起来。

2.4.5 算法执行的例子

假设用户想在给定的传感器网络中发现3条不相交的$s \sim t$路径。在第一步中，EDA构建了一棵如图 2-3 所示的生成树T。树中的路径sTt被看作是第一条发现的路径P_1。

接下来，EDA进入到了第二步。因为路径P_1上的第一跳节点是a，所以$T(a)$中的节点交换信息来发现网络中的桥路径。下面，让我们用四条FIND-BRIDGE消息交换的路线来示意整个过程。



a) 第一循环中构建的G^*
a) The G^* in the first iteration

b) 第二循环中构建的G^*
b) The G^* in the second iteration

c) 所产生的三条路径
c) The generated 3 paths

图 2-4 每个循环中s构建的G^*（实线表示路径上的边，虚线表示桥路径）

Fig. 2-4 The G^* constructed by s in each iteration (solid lines denote edges in the found paths, dashed lines denote bridge paths)

路线1：因为a是路径P_1上的第一跳节点，它发送FIND-BRIDGE消息$\{P_1, 1\}$给节点q。接收到了这个消息之后，节点q设置$bridge(q, P_1) = 1$和$LBH(q, P_1) =$

a，并广播FIND-BRIDGE消息$\{P_1, 1\}$。接收到q发送的这个消息之后，节点p设置$bridge(p, P_1) = 1$和$LBH(p, P_1) = q$，之后广播FIND-BRIDGE消息$\{P_1, 1\}$。

路线2：因为d是路径P_1上的第三跳节点，它发送FIND-BRIDGE消息$\{P_1, 3\}$给节点h。接收到了这个消息之后，节点h设置$bridge(h, P_1) = 3$和$LBH(h, P_1) = b$，之后广播FIND-BRIDGE消息$\{P_1, 3\}$。

路线3：d还发送FIND-BRIDGE消息$\{P_1, 3\}$给节点e。

路线4：因为节点e有一个源不是a的邻居f，e设置$bridge(e, s) = TRUE$和$LBH(e, s) = f$，之后广播FIND-BRIDGE消息$\{s, 0\}$。

下面路线3和路线4交叉了。当节点d接收到节点e发送的FIND-BRIDGE消息$\{s, 0\}$之后，它设置$bridge(d, s) = TRUE$和$LBH(d, s) = e$。当节点e接收到节点d发送的FIND-BRIDGE消息$\{P_1, 3\}$之后，因为e已经发现了一条从s到其自身的桥路径，所以它忽略这个消息。

路线1和路线2也交叉了。当节点p接收到了节点h发送的FIND-BRIDGE消息$\{P_1, 3\}$之后，因为p已经发现了一条从P_1上第一跳节点到其自身的桥路径，所以p忽略这个消息。当节点h接收到节点p发送的FIND-BRIDGE消息$\{P_1, 1\}$之后，因为$bridge(h, P_1) = 3 > 1$，h更新$bridge(h, P_1) = 1$和$LBH(h, P_1) = p$，之后广播FIND-BRIDGE消息$\{P_1, 1\}$。当节点d接收到这个消息之后，因为d已经发现了一条从s到其自身的桥路径，它忽略这个消息。

在第二步的末尾，路径上的节点保存了以下桥路径信息：节点a没保存任何信息；节点d保存$bridge(d, s) = TRUE$；节点r保存$bridge(r, P_1) = 1$；节点g保存$birdge(g, s) = TRUE$；节点t保存$bridge(t, P_1) = 1$。

接下来，EDA进入第三步，路径节点保存的桥路径信息被收集到节点s。为启动这个过程，节点t发送COLLECT-BRIDGE消息$\{[t, P_1, 1]\}$给它的上一跳节点g。接收到了这个消息之后，节点g向这个消息中添加$[g, s, 0]$，并将这个消息转发给节点r。当r接收到了这个消息之后，因为消息中包含了$[g, s, 0]$，根据规则6，r不向这个消息添加任何信息。节点d也是一样。当s接收到了这个COLLECT-BRIDGE消息之后，它构建一个如图 2-4 a) 所示的G^*。

接着，EDA进入第四步。s在G^*中搜索增广路径并找到了增广路径$P_a = (s \dashrightarrow g, \ldots, a \dashrightarrow t)$。其中“$x \dashrightarrow y$”表示经过一条从$x$到$y$的桥路径。将$P_a$添加到$P_1$上，我们可以得到两条新的路径$P_1' = (s, a \dashrightarrow t)$和$P_2' = (s \dashrightarrow g, t)$。通过对它们进行追溯，我们可以得到如图 2-4 b) 中粗线所示的路径。

接下来，EDA重新进入第二步。$T(a)$和$T(b)$中的节点交换消息来发现桥路径。在收集完桥路径信息之后，节点s构建的G^*如图 2-4 b) 所示。这一次，我们可以在G^*中发现增广路径$P_a = (s \dashrightarrow v, \ldots, f \dashrightarrow l, \ldots, u \dashrightarrow t)$。由此，我们可以得

到如图 2-4 c) 所示的3条路径。

2.5 算法正确性的证明

让$P_1,\ldots,P_n$表示算法EDA在第n次循环结束时所发现的路径，$c_1,\ldots,c_n$分别表示它们的第一跳节点。在路径表达式中出现的"$u \overset{B}{\dashrightarrow} v$"表示从$u$到$v$的桥路径$B$，在路径表达式中出现的"$u \xrightarrow{P_i} v$"表示已发现路径$P_i$上（沿着从$t$到$s$的方向）从$u$到$v$的片段。

由于已经有了定理 2.1，我们要证明EDA可以保证答案正确性只需要证明：i) EDA找到的每条增广路径都是一条正确的增广路径；ii) 如果网络中存在一条增广路径，EDA一定能够找到一条增广路径。

引理 2.1 $P_1,\ldots,P_n$上除s外的所有节点都在$T(c_1)\cup\cdots\cup T(c_n)$中。

证明： 我们可以对n进行数学归纳来证明这个引理。当$n=1$的时候，树中的路径sTt被看作是第一条被发现的路径P_1。显然，P_1上除s外的所有节点都在$T(c_1)$中。

假设当$n=m-1$的时候这个引理成立。也就是在第$m-1$次循环结束时，EDA发现的路径$P_1,\ldots,P_{m-1}$上除s外的所有节点都包含于$T(c_1)\cup\cdots\cup T(c_{m-1})$中。其中，$c_1,\ldots,c_{m-1}$分别是路径$P_1,\ldots,P_{m-1}$上第一跳节点。

接下来EDA进入第m次循环，$T(c_1)\cup\cdots\cup T(c_{m-1})$中的节点交换消息来发现桥路径。然后这些桥路径的信息被收集到s节点。对于所收集到的从u到v的桥路径B来说，如果$u\neq s$，则B上的所有节点都在$T(c_1)\cup\cdots\cup T(c_{m-1})$中。如果不是这样，假设$B$上有一个不在$T(c_1)\cup\cdots\cup T(c_{m-1})$中的节点$w$。根据EDA第二步中对规则1和规则3的实现，$v$就会发现一条从$s$到$v$的桥路径。而根据EDA第二步中对规则4的实现，$B$的信息就不会被记录下来。

接着EDA利用所收集的桥路径的信息寻找增广路径。根据增广路径的定义，增广路径P_a中至多包含一个从s到其它节点v的桥路径，让B'来表示这个桥路径。根据上面的描述，除了B'，P_a的其它部分都位于$T(c_1)\cup\cdots\cup T(c_{m-1})$中。当EDA追溯$B'$的时候，从节点$v$开始，每个节点都将TRACE-PATH消息发送给自己在桥路径中的上一跳节点$LBH(s)$，直到消息到达某个节点w满足$w\notin T(c_1)\cup\cdots\cup T(c_{m-1})$。假设$origin(w)=c_m$。接下来，从$w$开始，每个节点都将TRACE-PATH消息发送给自己的父亲节点。于是B'中从s到w的部分被建立为生成树中的路径sTw。除了这一部分，P_a中的其它部分都位于$T(c_1)\cup\cdots\cup T(c_{m-1})$中，而$sTw$上除了$s$以外的节点都位于$T(c_m)$中。通过将$P_a$添加到$P_1,\ldots,P_{m-1}$上，得到的新路径上除$s$以外的节点一定都位于$T(c_1)\cup\cdots\cup T(c_m)$中。 □

定理 2.2 EDA构建的图G^*中的增广路径代表了网络中的增广路径。

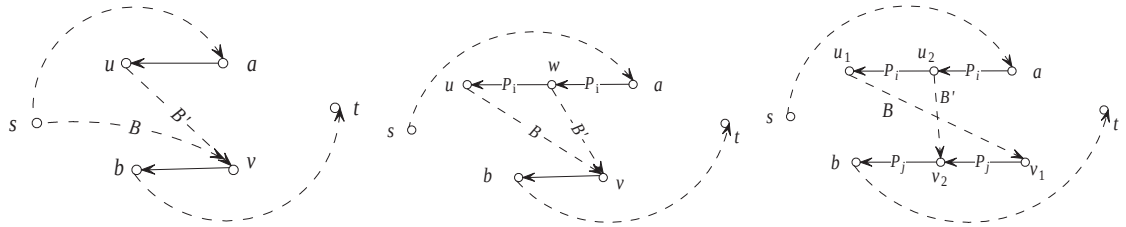
证明： 这个定理相当于：G^*中的桥路径代表网络中的桥路径。

EDA发现桥路径的基本规则是规则1、2和3，而规则4、5和6只是为了使算法更有效率而去掉一些无用的桥路径。因此我们只需证明规则1、2和3 是正确的。规则2和3的正确性可以从桥路径的定义中得到。而根据引理 2.1，规则1也是正确的。 □

引理 2.2 规则4不会影响正确寻找增广路径。

证明： 让v表示一个路径节点。有一条从s到v的桥路径B。根据规则4，其它到节点v的桥路径都是无用的。让$B' = (u, ..., v)$表示另一条到节点v的桥路径。假设有一条增广路径$P_a = (s, \ldots, u \overset{B'}{\dashrightarrow} v, \ldots, t)$包含了$B'$。通过将$P_a$中从$s$到$v$的部分替换为$B$，我们可以得到另一个路径$P'_a = (s \overset{B}{\dashrightarrow} v, \ldots, t)$。由增广路径的定义可知，$P'_a$也是一条增广路径。 □

作为一个应用规则4的例子，假设存在一条如图 2-5 a) 所示的增广路径$P_a = (s \dashrightarrow a \rightarrow u \overset{B'}{\dashrightarrow} v \rightarrow b \dashrightarrow t)$，其中$B'$是从$u$到$v$的桥路径。另有一条从$s$到$v$的桥路径$B$。于是，我们可以得到另一条增广路径$P'_a = (s \overset{B}{\dashrightarrow} v \rightarrow b \dashrightarrow t)$。这条增广路径不包含$B'$。



a) 规则4　　b) 规则5　　c) 规则6

a) Rule 4　　b) Rule 5　　c) Rule 6

图 2-5 应用规则4、5、6的例子(实线表示已知路径的片段，虚线表示桥路径)

Fig.2-5 Examples of applying Rule 4, 5, 6 (solid lines denote segments of found paths, dashed lines denote bridge paths)

引理 2.3 为了保证答案正确性，只需要$T(c_1), \ldots, T(c_n)$中的节点参与发现桥路径。

证明： 对于每个不在$T(c_1), \ldots, T(c_n)$中的节点v，算法设定存在一条从s到v的桥路径。根据引理 2.1，我们知道这是正确的。根据引理 2.2，这些节点不需要发现其它的桥路径。 □

引理 2.4 规则5不会影响正确寻找增广路径。

证明： 让v表示一个路径节点。已经有一条从u到v的桥路径B，其中u是已发现路径P_i上第x跳节点。根据规则5，从w 到v的桥路径B'是无用的，其中w是P_i上第y跳节点并且$y \geq x$。假设有一条增广路径$P_a = (s, \ldots, a \xrightarrow{P_i} w \overset{B'}{\dashrightarrow} v, \ldots, t)$包含$B'$。我们将$w \xrightarrow{P_i} u$和$B$连接并以此来替代$B'$，这样我们便得到了路径$P'_a = (s, \ldots, a \xrightarrow{P_i} w \xrightarrow{P_i} u \overset{B'}{\dashrightarrow} v, \ldots, t)$。根据增广路径的定义，$P'_a$也是一条增广路径。 □

作为应用规则5的例子，假设存在一条如图 2-5 b) 所示的增广路径$P_a = (s \dashrightarrow a \xrightarrow{P_i} w \overset{B'}{\dashrightarrow} v \rightarrow b \dashrightarrow t)$。其中，$B'$是从$w$到$v$的桥路径并且$w$是已发现路径$P_i$上第$y$跳节点。还有一条从$u$到$v$的桥路径$B$，这里$u$是$P_i$上第$x$跳节点并且$x < y$。于是，我们可以得到另一条增广路径$P'_a = (s \dashrightarrow a \xrightarrow{P_i} u \overset{B'}{\dashrightarrow} v \rightarrow b \dashrightarrow t)$。这条增广路径不包含$B'$。

引理 2.5 规则6不会影响正确寻找增广路径。

证明： 让u_1表示已发现路径P_i上第x_1跳节点，v_1表示已发现路径P_j上第y_1跳节点。已经发现从u_1到v_1的桥路径B。根据规则6，不需要再发现任何从u_2到v_2的桥路径B'。其中，u_2是P_i上第x_2跳节点，v_2是P_j上第y_2跳节点，并且满足$x_1 \leq x_2$和$y_1 \geq y_2$。假设有一条增广路径$P_a = (s, ..., a \xrightarrow{P_i} u_2 \overset{B'}{\dashrightarrow} v_2 \xrightarrow{P_j} b, ..., t)$包含$B'$。我们可以将$u_2 \xrightarrow{P_i} u_1$，$B$和$v_1 \xrightarrow{P_j} v_2$连接起来并以此来代替$B'$，这样我们就得到路径$P'_a = (s, ..., a \xrightarrow{P_i} u_2 \xrightarrow{P_i} u_1 \overset{B}{\dashrightarrow} v_1 \xrightarrow{P_j} v_2 \xrightarrow{P_j} b, ..., t)$。根据增广路径的定义，$P'_a$也是一条增广路径。 □

作为应用规则6的例子，假设存在如图2-5 c) 所示的增广路径$P_a = (s \dashrightarrow a \xrightarrow{P_i} u_2 \overset{B'}{\dashrightarrow} v_2 \xrightarrow{P_j} b \dashrightarrow t)$。这里$B'$是从$u_2$到$v_2$的桥路径。另外还有一条从$u_1$到$v_1$的桥路径。其中，$u_1$和$u_2$分别是已发现路径$P_i$上的第$x_1$和$x_2$跳节点，$v_1$和$v_2$分别是已发现路径$P_j$上的第$y_1$和$y_2$跳节点，$x_1 \leq x_2$，$y_1 \geq y_2$。于是，我们可以得到另一条增广路径$P'_a = (s \dashrightarrow a \xrightarrow{P_i} u_1 \overset{B'}{\dashrightarrow} v_1 \xrightarrow{P_j} b \dashrightarrow t)$。这条增广路径不包含$B'$。

定理 2.3 如果网络中存在一条增广路径，EDA一定会在图G^*中找到一条增广路径。

证明： EDA并不是发现网络中所有的桥路径，而是只让$T(c_1), \ldots, T(c_n)$中的节点参与发现桥路径，并根据规则4、5、6来去除不必要的桥路径。根据引理 2.2，2.3，2.4和2.5，这些措施都不会影响正确寻找增广路径。 □

2.6 实验结果

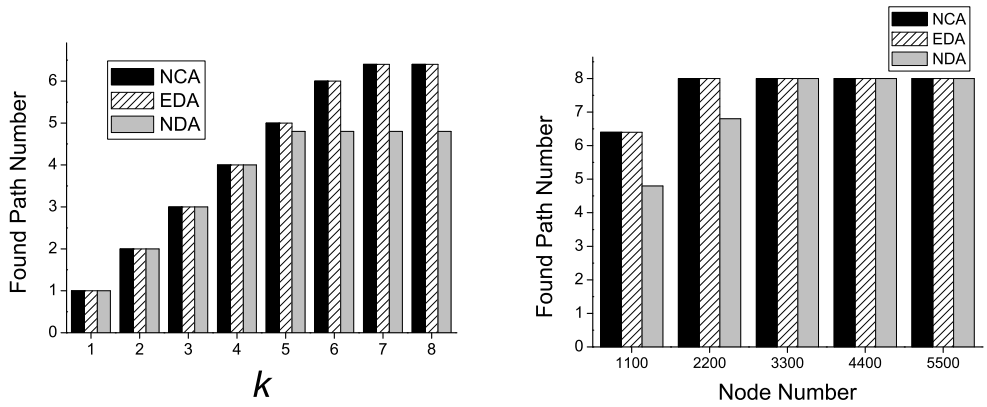
我们用一个C++编写的模拟器来验证算法的效率。这个模拟器是一个基于应用层的程序，忽略了MAC层的细节。在实验中，我们在一个1000m ×1000m的监测区域内布置传感器节点。每个节点的位置随机产生。每个节点的通信半径被设置为50m。每个数据包包含了4个字节的包头来分别表示这个数据包的：目的节点的ID；发送者ID；数据包类型（例如：BUILD-TREE消息、FIND-BRIDGE消息、COLLECT-BRIDGE消息等）；数据包长度。每个接收数据和时钟到时事件被加上时间戳统一放到处理事件的堆中。堆中的事件按其时间戳的先后顺序一个一个进行处理。通过这种方式，我们来模拟网络中节点间的并行操作。为了获取每个实验数据，我们做了10次不同的实验（每次都设置不同的s和t）来取它们结果的平均值。

算法的效率通过三个方面来衡量：算法找到的路径数量、算法的通信代价以及算法为找到每条路径消耗的通信代价。算法的通信代价由实验中总的通信量（网络中信息交换的字节数）除以节点的数量得到，也就是平均每个节点接收和发送的字节数。根据1.2.1中的描述，通信代价可以很好地表示节点的能量消耗。算法为找到每条路径消耗的通信代价由算法的通信代价除以算法找到的路径数量得到。

我们将算法EDA和其它两个算法相比较，它们是：文献[141]中的集中式算法（用NCA表示）和文献[131]中的分布式算法（用NDA表示）。NCA将整个网络的拓扑信息（每个节点的邻居列表）收集到基站，然后在基站节点利用文献[141]中的方法来搜索增广路径。NCA是文献[135–137, 139, 141]中这些集中式算法的一个代表。因为这些集中式方法都需要收集整个网络的拓扑信息，所以它们有相同的通信代价。另一方面，这些方法都可以保证答案正确性，所以它们找到的路径数量相同。NDA分布式地在网络中寻找不相交路径。每找到一条路径，NDA就将这条路径固定下来并将路径上的节点从网络中删除。在文献[127–134]给出的这些分布式方法中，NDA是最具通信效率的一个。注意，NDA不能保证答案正确性，也就是当网络存在k条不相交路径时，NDA可能只找到$n << k$条路径。

为了比较这些算法，我们设计了两组实验。在第一组实验中，我们在监测区域内（1000m×1000m）布置了1100个节点，因此每个节点平均有8.6个邻居。我们根据不同的k，也就是用户需要找到的路径数量，来比较这三个算法。在第二组实验中，我们固定$k = 8$，在监测区域内分别布置了1100、2200、3300、4400、5500个节点来比较这三个算法。

首先，我们来比较三个算法找到的路径的数量。第一组实验的结果如图 2-6 a) 所示。因为EDA和NCA都可以保证答案正确性，所以它们找到的路径数量都相同。当$k > 4$时，NDA找到的路径比其它两种算法要少，其不能保证答案正确性的劣势显示出来。具体来说，EDA和NCA都最多能够找到6.4条不相交路径（10次实验的平均值），这是网络中存在的最多的不相交路径的数量。而NDA最多只能够找到4.8条路径。第二组实验的结果如图 2-6 b) 所示。当节点数量为1100, 2200时，对比于NCA和EDA，NDA找到的路径数量要更少，这正是不能保证答案正确性的劣势。当节点数量大于3300时，这三种方法找到的路径数量相同。如果k被固定下来，在一个非常密集的网络中，EDA和NCA 在路径数量方面的优势不再那么明显。但是，在实际应用中，人们很难有能力布置如此密集的网络（例如在1000m×1000m的区域内布置3000 ~ 5000个节点）。



a) 第一组实验　　b) 第二组实验

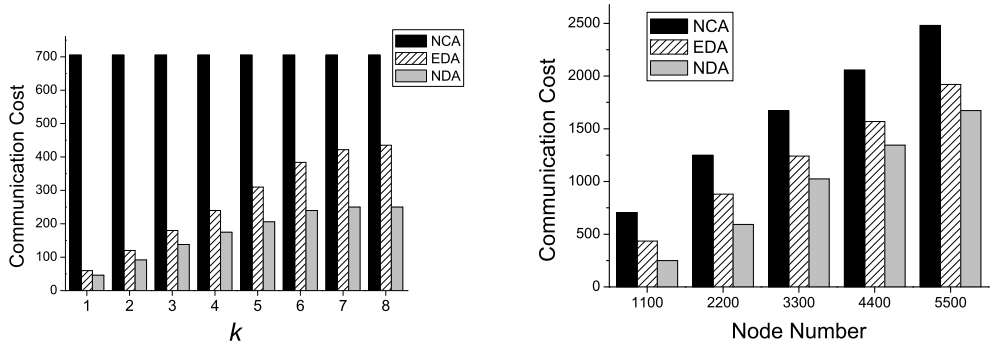
a) Group 1　　b) Group 2

图 2-6 三个算法找到的路径数量

Fig. 2-6 Number of the paths found by the 3 algorithms

接着，我们来比较这三个算法的通信代价，即平均每个节点接收和发送的字节数。第一组实验的结果如图 2-7 a) 所示。由于其集中式的处理方式，无论k多大，NCA都要收集整个网络的拓扑信息。因此，NCA的通信代价不随k的变化而变化。由于是增量的分布式算法，EDA和NDA的通信代价随着k的增长而增长。相比于NDA，EDA的通信代价要更高一些。但是EDA可以保证答案正确性而NDA不能。当$k > 4$时，这两个算法在通信代价上的差距比较明显。这是因为EDA找到了更多的路径。因此，EDA多消耗一些通信代价是完全可以接受的。第二组实验的结果如图 2-7 b) 所示。同样，在通信代价方面，EDA要比NCA好得多，但是比NDA要差一些。当节点数量为1100, 2200时，EDA和NDA的差距比

较明显是因为EDA找到了更多的路径。当节点数量大于2200时，EDA与NDA并没有很大的差距。



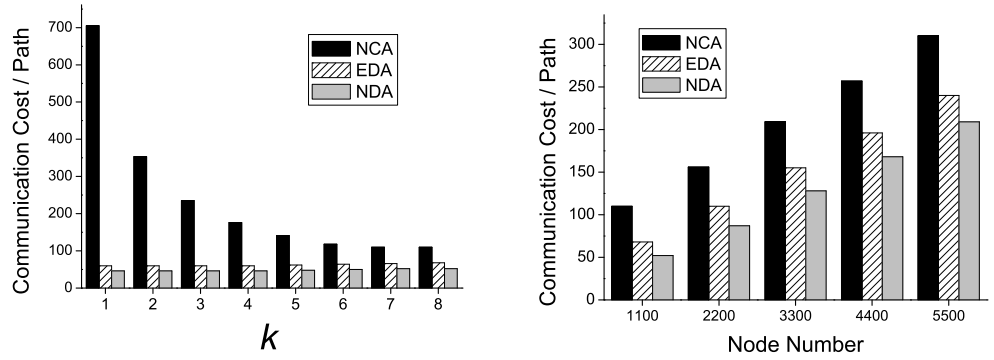
a) 第一组实验　　b) 第二组实验

a) Group 1　　b) Group 2

图 2-7 三个算法的通信代价

Fig.2-7 Communication cost of the 3 algorithms

最后，我们来比较这三个算法为找到每条路径平均消耗的通信代价，这也是一个更公平的比较通信代价的办法。两组实验的结果分别如图 2-8 a) 和图 2-8 b) 所示。相比于同样可以保证答案正确性的集中式算法，EDA在通信代价方面拥有明显的优势。同不能保证答案正确性的分布式方法相比，EDA的通信代价是完全可以比较的。



a) 第一组实验　　b) 第二组实验

a) Group 1　　b) Group 2

图 2-8 三个算法的为找到每条路径消耗的通信代价

Fig.2-8 Communication cost to find each path for the 3 algorithms

保证答案正确性的优势在一个稀疏的网络中体现的更加明显，但是通过随

机产生节点坐标的方式来产生一个稀疏网络几乎是不可能的。根据文献[172]，通过随机布置节点的方式，平均每个节点至少要有7个邻居才能构成一个连通的网络。而在许多应用中，考虑到费用等因素，节点被用户布置在指定的位置。通过调整节点的传输能量，也就是调整节点的通信半径，来构成一个连通的网络（这样网络中每个节点有数量很少的邻居）。在第三组实验中，我们模拟了一个如图 2-9 所示的网络并设定k = 3。在这个例子中，EDA发现了三条路径而NDA只发现一条路径。通过这个例子，我们更清楚地看到了保证答案正确性的优势。

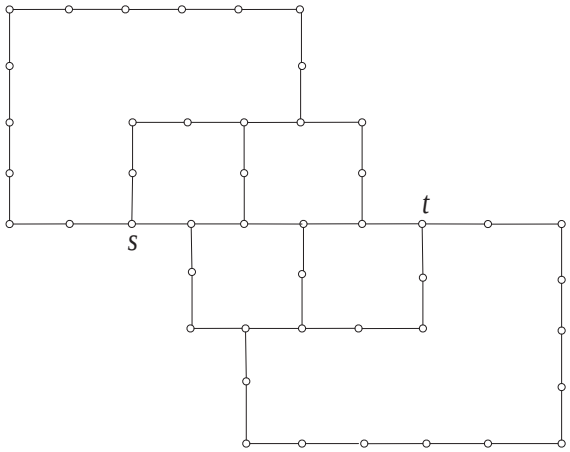


图 2-9 一个特殊的网络

Fig. 2-9 A Special Network

2.7 本章小节

本章给出了一个高效的分布式算法EDA用来寻找k条连接两个给定节点的不相交路径。不同于现有的其它分布式方法，EDA可以保证答案正确性，也就是EDA能找到网络中最多的不相交路径。相比于现有的同样可以保证答案正确性的集中式方法，EDA采取分布式的处理方式，具有更高的通信效率。对比于当前最有效率的分布式方法，EDA具有完全可以比较的通信效率，并且还有保证答案正确性的巨大优势，而该分布式方法却不能。结论是，EDA是传感器网络中正确高效的寻找不相交路径的算法。

第 3 章 无线传感器网络中优化的k不相交多路径路由

3.1 引言

由于传感器网络多跳自组织的特点，如何为其设计高效的路由策略就成为了传感器网络研究的关键问题。路由的目的就是寻找源节点和目的节点间的优化路径。在传感器网络中，利用两个节点间多条不相交路径进行路由可以有效的提升路由的吞吐量、可靠性、负载平衡和安全性。在两个节点间利用多条不相交路径并行传输多个数据包可以有效地增加它们之间数据传输的吞吐量；将数据备份成多份并沿着连接两个节点的多条不相交路径进行传输可以增加传输的可靠性；在两个节点间交替使用多条不相交路径传输数据可以使网络中的能量消耗更加平衡；为了提高信息的安全性，我们可以将敏感数据拆分为多个部分，然后将这些部分分别沿着多条不相交路径进行传输。

虽然采用不相交路径路由可以提升上述的网络性能，但同时我们也不能忽略传感器网络设计的主要目标—能量优化。在传感器网络中，每个无线链接都有自己的传输能量代价。假设用户对于吞吐量的需求要求我们必需在s节点和t节点之间用k条不相交路径并行传输大量数据。于是每条路径上的数据传输以“前赴后继”的方式在不停地进行着。在一个传输周期内，这些路径上的每个链接都被用作进行一次数据传输。在满足数据吞吐量的同时，用户总是希望单位传输周期内的能量消耗最小，也就是这些路径的能量代价之和最小。有些情况下，为了实现网络中的负载平衡，s节点和t节点需要交替使用k条不相交$s \sim t$路径来传输数据。每传输k个数据包的能量消耗等于这k条路径的能量代价之和。在实现网络负载平衡的同时，用户总是希望用于传输单位数据包的能量消耗最小，也就是这k条路径的能量代价之和最小。

另一方面，有些应用要求路由策略考虑对传输延迟的优化。网络中的每个无线链接都有自己的传输延迟。还是假设为了实现网络的负载平衡，我们在s节点和t节点之间交替使用k条不相交路径来传输数据。每传输k个数据包的时间延迟等于这k条路径的时间延迟之和。在实现网络负载平衡的同时，我们希望用于传输单位数据包的时间延迟最小，也就是这k条路径的延迟之和最小。

假设在给定的传感器网络中，每个无线链接都有一个长度（权值）来表示它的能量代价或者传输延迟，一条路径的长度是其上所有链接的长度之和。本

章中我们研究的是最优k不相交路径问题：给定网络中的节点s和t，以及正整数k，寻找k条长度之和最小的不相交$s \sim t$路径。

解决这个问题的难点在于如何保证答案正确性和结果最优性。所谓答案正确性是指如果网络中存在k条不相交$s \sim t$路径，算法一定会输出k条不相交$s \sim t$路径。也就是说这个算法能够找到最多的不相交$s \sim t$路径。所谓结果最优性是指算法所找到的k条不相交路径在所有的可能解中具有最小的长度之和。虽然寻找多条不相交$s \sim t$路径的问题得到了广泛的研究[127–137, 139, 141]，但是这些文献中所提出的算法要么没有考虑路径的长度或者不能保证答案正确性，要么是集中式算法而不适合应用于传感器网络中。

为此，本章提出了一个完全分布式的算法OFDP用来解决最优k不相交路径问题。OFDP既能保证答案正确性又能保证结果最优性。OFDP采用了完全分布式的处理方式，不需要收集任何网络信息，网络中的信息交换不依赖任何结构。每个节点通过与邻居交换短小的消息来发现路径，因此算法具有很高的通信效率。此外，通过对OFDP进行简化，本章还给出了完全分布式的算法FDP用来解决第 2 章中提出的k不相交路径问题，即寻找k条不相交的$s \sim t$路径而无需优化路径长度。FDP可以保证答案正确性。与第 2 章中提出的分布式算法EDA相比，由于采用完全分布式的处理方式，FDP具有更高的通信效率。因此，本章的贡献在于：

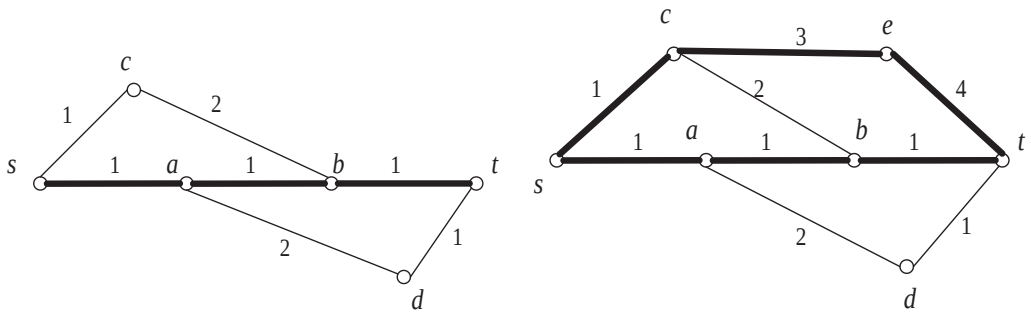
- 提出了一个高效的完全分布式算法OFDP用来解决最优k不相交路径问题，该方法可以保证答案正确性和结果最优性。
- 提出了一个高效的完全分布式算法FDP用来解决k不相交路径问题，该方法可以保证答案正确性。

本章的结构安排如下：3.2介绍了当前相关问题的研究现状；3.3给出了一些必要的定义和定理；3.4 给出了用于解决最优k不相交路径问题的完全分布式算法OFDP并证明了OFDP既能保证答案正确性又能保证结果最优性；在OFDP的基础上，3.5给出了用于解决k不相交路径问题的完全分布式算法FDP并证明了FDP可以保证答案正确性；3.6验证了这两个算法的效率；3.7总结全章。

3.2 相关工作

许多的分布式方法[127–134]被提出用来寻找多条不相交$s \sim t$路径。这些方法都有着相同的思想：循环地寻找与已经找到的路径不相交的$s \sim t$路径（通常是寻找当前长度最短的路径）。一旦找到一条路径，这条路径就被固定下来，其上节点不再参与其它路径的建立。这些方法固然简单且容易实现，但

是如果我们用这些方法来解决最优k不相交路径问题，得到的答案很可能缺少路径数量或者不是最优解。假设在一个如图 3-1 a) 所示的传感器网络中，用户想要寻找2条不相交$s \sim t$ 路径。应用上述的方法，我们找到的第一条路径是$P_1 = (s, a, b, t)$。于是，这条路径就被固定下来。接着，我们发现网络中不存在与P_1不相交的$s \sim t$路径，因此报告网络中只有1条不相交的$s \sim t$路径。而实际上，网络中存在2条不相交的$s \sim t$路径，它们是(s, c, b, t)和(s, a, d, t)。再假设在如图 3-1 b) 所示的传感器网络中，用户想要寻找2条长度之和最小的不相交$s \sim t$路径。采用上述方法，我们找到的第一条路径将会是$P_1 = (s, a, b, t)$，第二条路径将会是$P_2 = (s, c, e, t)$。这两条路径的长度之和为$3 + 8 = 11$。但实际上，最优解是路径$P'_1 = (s, c, b, t)$和$P'_2 = (s, a, d, t)$，它们的长度之和为$4 + 4 = 8$。



a) 例子1　　b) 例子2

a) Example 1　　b) Example 2

图 3-1 传感器网络的例子（边上的数字表示边的长度）

Fig. 3-1 Examples of sensor networks (numbers on edges denote edges' length)

在图论领域，文献[135–137, 139, 141]提出了集中式方法用来寻找k条不相交$s \sim t$路径。这些方法都可以保证答案正确性。其中，文献[137, 139, 141]中的算法可以找到长度之和最小的k条不相交路径。由于这些方法都采用了集中式的处理方式，如果我们用这些方法来解决传感器网络中的最优k不相交路径问题，我们将不得不收集整个网络的拓扑信息并根据网络的变化实时更新这些拓扑信息。对于节点功能简单且能量有限的传感器网络来说，这项工作的代价过于高昂，所以这些集中式的方法不适合应用在传感器网络中。

还是在图论领域，文献[146–148]考虑了长度受限的不相交$s \sim t$路径问题：在图中给定节点s和t，以及用户制定的长度限制L，寻找最多的长度不大于L的不相交$s \sim t$路径。当图中的每条边的长度都是1并且$L \geq 5$时，文献[146]证明了这个问题是NP难问题。还是在同样的假设情况下，文献[147]进一步证明了这个问题是APX完全问题，即不存在关于这个问题的多项式时间的近似模式。文

献[148]给出了这个问题的一个启发式算法。

对于k条$s \sim t$路径，我们给每条路径一个序列号，由此区分这些路径为第1条、第2条、...、第k条路径。网络中的每条边（链接）都有k个长度分别表示其位于第1条、第2条、...、第k条路径中的代价。这样的网络被称作多代价网络。文献[142–145]给出了集中式算法在多代价网络中寻找k条长度之和最小的不相交$s \sim t$路径。

到目前为止，对于最优k不相交路径问题，所有能保证答案正确性和结果最优性的方法都是集中式的方法。如果将这些方法应用在传感器网络中，我们需要收集整个网络的拓扑信息并实时更新这些信息，这项操作的高昂代价使得这些方法不适合于传感器网络。虽然有一些高效的分布式方法可以寻找多条不相交$s \sim t$路径，但是这些方法既不能保证答案正确性也不能保证结果最优性。

3.3 预备知识

我们假设给定的传感器网络是静态的；网络中所有节点共用一个信道；所有的链接都是双向的；网络是连通的。我们可以将给定的网络用其拓扑图$G = (V, E)$表示。其中，顶点（节点）集合为$V = \{v \mid v$ 是网络中的节点$\}$，边集合为$E = \{(u, v) \mid u$ 和v 之间存在无线链接$\}$。对于每条边（链接）$e \in E$，$l(e) \in \mathbb{R}$表示它的长度，或说代价。最初的时候，每条边的代价都是非负数。如果我们用到了有向边的概念，我们用$< u, v >$来表示从u到v的有向边。

图G中的一条路径P是G的一个子图，可以表示为一系列不同的节点，即$P = (v_1, v_2, \ldots, v_n)$。其中，$v_i \in V$，$(v_i, v_{i+1}) \in E$。这里，$v_1$和$v_n$是$P$的两端节点，$P$上的其它节点被称作中间节点。一条两端节点是$v_1$和$v_n$的路径被称作一条$v_1 \sim v_n$路径。对于网络中的每条路径$P$来说，它的长度为$l(P) = \sum_{e \in P} l(e)$。

不相交路径的概念已经在定义 2.1 中给出。假设我们已经在图G中找到了n条不相交$s \sim t$路径$P_1, \ldots, P_n$。在这些路径上的节点被称作路径节点，图G中的其它节点被称作非路径节点。让v表示已发现路径P_i上的一个节点。v的上一跳节点是P_i上距离s更近的v的邻居，v的下一跳节点是P_i上距离t更近的v的邻居。对于有一个端点是v的边来说，位于P_i上的边被称作v的内部边，其它的边被称作v的外部边。除了v的上一跳和下一跳节点外，v的其它邻居被称作v的外部邻居。

我们已经在定义 2.3 中给出过增广路径的概念，下面我们从另一个角度重新定义增广路径。在发现n条不相交$s \sim t$路径$P_1, \ldots, P_n$之后，我们通过以下的修改将图G转换为图G^*:

- 对于每个路径节点$v \neq s,t$：将v分裂为两个节点v_0和v_1，并添加有向边$< v_1, v_0 >$，定义这条边的长度为$l(< v_1, v_0 >) = 0$。
- 对于每条已发现路径上的边(u,v)（u是v的上一跳节点）：将其替换为有向边$< v_0, u_1 >$，它的长度是原有边的长度的相反数，即$l(< v_0, u_1 >) = -l((u,v))$。
- 对于每条在路径节点v上的外部边(w,v)：如果w是一个非路径节点（没有被分裂过），将(w,v)替换为有向边$< w, v_0 >$和$< v_1, w >$，它们的长度为$l(< w, v_0 >) = l(< v_1, w >) = l((w,v))$；如果$w$是一个路径节点（被分裂过），将$(w,v)$替换为有向边$< w_1, v_0 >$和$< v_1, w_0 >$，它们的长度为$l(< w_1, v_0 >) = l(< v_1, w_0 >) = l((w,v))$。

假设给定的传感器网络如图 3-1 a) 所示。在这个网络中，我们已经找到了1条$s \sim t$路径$P_1 = (s,a,b,t)$，如图 3-1 a) 中粗线所示。通过上述变换，我们可以得到如图 3-2 所示的图G^*。

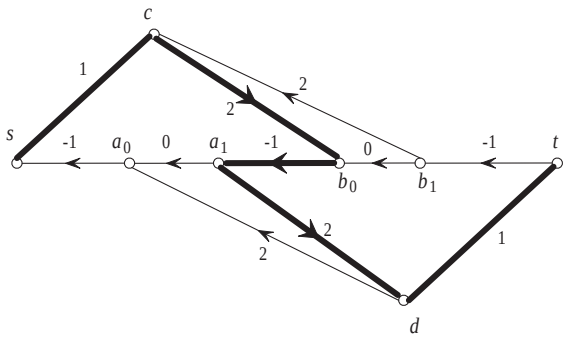


图 3-2 根据图 3-1 a) 的例子生成的G^*以及其中的增广路径

Fig.3-2 G^* generated by the example of Figure 3-1 a) and the augmenting path in it

如果G^*中存在一条从s到t的路径P_a^*，通过将每个v_0或者v_1替换为原来的节点v，我们可以得到P_a^*在G中对应的路径P_a。这样的P_a就是一条*增广路径*。通过将P_a加到已发现的路径$P_1, \ldots, P_n$上，我们可以得到$n+1$条不相交$s \sim t$路径。这里，“加”表示添加那些只属于P_a的边并删除那些P_a和$P_1, \ldots, P_n$共有的边。还是在如图 3-1 a) 所示的例子中，找到路径$P_1 = (s,a,b,t)$ 之后，我们可以在G^*中找到从s到t的路径P_a^*，如图 3-2 中粗线所示，它对应了增广路径$P_a = (s,c,b,a,d,t)$，通过将P_a加到P_1上，我们可以得到2条不相交的$s \sim t$路径(s,c,b,t)和(s,a,d,t)。

这种增广路径的定义与第 2 章中的定义 2.3 是等价的[139]。例如，在图 2-2 所示的例子中，在发现如图 2-2 a) 中粗线所示的2条路径之后，我们将G转换为如图 3-3所示的G^*。这里，我们省略了对边的长度的描述。在G^* 中，我们可以发现一条从s到t的路径P_a^*，如图 3-3 中粗线所示。这条路径对应了图 2-2 b)中的

增广路径P_a。通过将P_a添加到已发现的路径上，我们可以得到如图 2-2 c) 中所示的3条不相交路径。

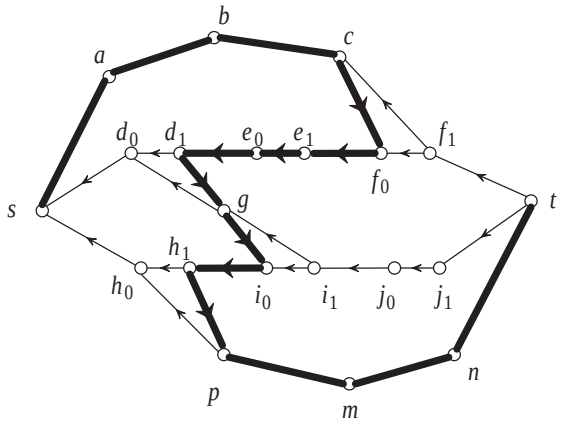


图 3-3 根据图 2-2 a) 的例子生成的G^*以及其中的增广路径

Fig.3-3 G^* generated by the example of in Figure 2-2 a) and the augmenting path in it

引理 3.1 假设在发现n条不相交$s \sim t$路径$P_1, \ldots, P_n$之后，我们在G^*中找到了从s到t的路径P_a^*，它对应增广路径P_a。通过将P_a加到$P_1, \ldots, P_n$上，我们得到了$n+1$条不相交$s \sim t$路径$P'_1, \ldots, P'_{n+1}$。让$l(P_a^*)$表示P_a^*在G^*中的长度。于是，有$l(P_a^*) + \sum_{i=1}^{n} l(P_i) = \sum_{i=1}^{n+1} P'_i$。

证明： 对于每条P_a与$P_1, \ldots, P_n$共有的边(u, v)：在P_a^*中这条边的代价被设置为$l(u, v)$的相反数，在计算$l(P_a^*) + \sum_{i=1}^{n} l(P_i)$时，相当于删除了这条边。对于每条只属于$P_a$的边$(u', v')$：在$P_a^*$中这条边的代价等于原来边的代价，在计算$l(P_a^*) + \sum_{i=1}^{n} l(P_i)$时，相当于添加了这条边。通过向$P_1, \ldots, P_n$上添加那些只属于$P_a$的边并删除那些$P_a$与$P_1, \ldots, P_n$共有的边，我们得到了$P'_1, \ldots, P'_{n+1}$。因此，有$l(P_a^*) + \sum_{i=1}^{n} l(P_i) = \sum_{i=1}^{n+1} P'_i$。 □

根据定理 2.1 和引理 3.1，我们知道可以通过以下方法寻找长度之和最小的k条不相交$s \sim t$路径，得到的结果既能保证答案正确性又能保证结果最优性：循环地在当前的G^*中寻找最短的从s到t的路径P_a^*；将P_a^*在G中对应的增广路径P_a加到已发现路径上产生更多的不相交路径。

3.4 完全分布式算法OFDP

在本节中，我们将给出一个用来解决最优k不相交路径问题的完全分布式算法OFDP。在给定的网络中，给定节点s和t以及用户指定的正整数k，OFDP能

够找到长度之和最小的k条不相交路径。OFDP能保证：i) 输出$\min\{k, k^*\}$条不相交$s \sim t$路径。这里假设网络中最多存在k^*条不相交$s \sim t$路径；ii) 输出的路径在所有可能结果中具有最小的长度之和。OFDP采用完全分布式的处理方式，通过在节点间交换少量短小的消息来发现路径，且这些信息交换不依赖于任何特定的结构。

OFDP的基本思想是循环地在网络中寻找当前的G^*中最短的从s到t的路径P_a^*（P_a^*在G中对应了增广路径P_a），然后将增广路径P_a添加到已知路径上构建更多的不相交路径，直至找到k条不相交路径为止。每个循环中有两步：寻找最短增广路径（第一步）和回溯增广路径（第二步）。第一步负责找到当前的G^*中最短的从s到t的路径P_a^*。第二步负责将P_a^*所对应的增广路径P_a加到已知路径上构建更多的不相交路径。整个算法由算法 3-1给出。由于真实网络的拓扑图是G而不是G^*，所以算法的难点在于如何在真实的网络中模拟一个拓扑图是G^*的虚拟网络。

Input: 两个节点s和t；正整数k
Output: $\min\{k, k^*\}$条长度之和最小的不相交$s \sim t$路径

```
repeat
    调用寻找最短增广路径算法;
    if 找不到增广路径 then
        停止并输出所有已发现的路径;
    调用回溯增广路径算法;
until 找到k条不相交s ~ t路径;
```

算法 3-1 OFDP
Algo. 3-1 OFDP

让v表示网络中的任意节点。我们假设v知道它的每个邻居u以及链接$l(u,v)$的代价。此外，v保存了变量$state(v) \in \{FREE, INPATH\}$。$state(v) = FREE$表示$v$是一个非路径节点，即$v$不在已发现的路径上。$state(v) = INPATH$表示$v$是一个路径节点。如果$v$是一个路径节点，它还保存了变量$prehop(v)$和$nexthop(v)$，分别用来记录所在路径上$v$的上一跳和下一跳节点。在每次循环中，算法都要找到当前的G^*中最短的从s到t的路径P_a^*。为此，我们需要找到G^*中从s到每个节点的最短路径。如果v是一个非路径节点，由于它在G^*中仍然对应了节点v，v保存变量$phap(v)$和$len_ap(v)$。$phap(v)$表示从s到v的最短路径中v的上一跳节点，$len_ap(v)$表示这条最短路径的长度。如果v是一个

路径节点，由于在G^*中它被分裂为两个节点v_0和v_1，所以它保存变量$phap(v_0)$，$leng\_ap(v_0)$，$phap(v_1)$和$leng_ap(v_1)$。$phap(v_0)$表示从s到v_0的最短路径中v_0的上一跳节点，$len_ap(v_0)$表示这条最短路径的长度。$phap(v_1)$表示从s到v_1的最短路径中v_1的上一跳节点，$len_ap(v_1)$表示这条最短路径的长度。

在算法的描述中，“节点v广播一个消息”表示这个消息可以被v的所有邻居接收到，正如传感器网络中的广播。“节点v发送一个消息给节点u” 表示这个消息只能被u接收到，v的其它邻居接收不到这个消息，正如传感器网络中的单播。

3.4.1 寻找最短增广路径

在这一步中，我们将找到当前的G^*中最短的从s到t的路径P_a^*。通过与其邻居交换FIND-PHAP 消息，如果v是一个非路径节点，它可以获知$phap(v)$；如果v是一个路径节点，它可以获知$phap(v_0)$和$phap(v_1)$。这一步相当于在G^*中建立以s节点为根的最短生成树，树中s到每个节点的路径都是最短路径。变量$phap$相当于节点在树中的父亲。对于每个路径节点v，它需要模拟两个节点v_0和v_1的行为，其中v_0和v_1为G^*中由v分裂出的两个节点。每个FIND-PHAP消息都包含了发送者的len_ap，即G^*中从s到发送者最短路径的长度。

寻找最短增广路径的伪代码由算法 3-2给出。整个过程由s节点广播FIND-PHAP消息{0}来初始化。

当一个非路径节点v接收到节点u发出的FIND-PHAP消息$\{len_ap(u)\}$时：由于G^*中从u到v的边长度等于$l(u,v)$，所以这条从s到u再到v的路径的长度为$len_ap(u) + l(u,v)$，我们用变量cur_len来记录之（算法 3-2 中第3行）。如果$cur_len < len_ap(v)$，这条从s到u再到v的路径就是当前G^*中的从s到v的最短路径，于是我们将$phap(v)$设置为u，更新$len_ap(v)$，并让v广播FIND-PHAP消息$\{len_ap(v)\}$（算法 3-2中第4-7行）。

当一个路径节点v接收到节点u发出的FIND-PHAP消息$\{len_ap(u)\}$时，如果u不是v的上一跳或者下一跳节点，则这个消息是沿着v 的一条外部边(u,v)到达v的。在G^*中，这个消息到达的是节点v_0。由于G^*中从u到v_0的边的长度等于$l(u,v)$，所以这条从s到u再到v_0的路径的长度为$len_ap(u) + l(u,v)$，我们用变量cur_len来记录之（算法 3-2 中第10行）。如果$cur_len < len_ap(v_0)$，这条从s到u再到v_0的路径就是当前G^*中从s到v_0的最短路径，于是我们将$phap(v_0)$设置为u并更新$len_ap(v_0)$（算法 3-2中第11-13行）。因为在G^*中v_0只有一个出边到v的上一跳节点$prehop(v)$，如果更新了$len_ap(v_0)$，v就发送FIND-PHAP消

Input: 已经发现的不相交路径$P_1,\ldots,P_n$

Output: 当前的G^*中最短的从s到t的路径P_a^*

/* 对于每个节点$v \neq s$的伪代码；整个过程由s节点广播FIND-PHAP消息{0}来初始化 */

1 **while** 接收到u发送的*FIND-PHAP*消息$\{len_ap(u)\}$ **do**
2 **if** $state(v) = FREE$ **then**
3 $cur\_len \leftarrow len\_ap(u) + l(u,v)$;
4 **if** $cur\_len < len\_ap(v)$ **then**
5 $phap(v) \leftarrow u$;
6 $len\_ap(v) \leftarrow cur\_len$;
7 广播FIND-PHAP消息$\{len_ap(v)\}$;
8 **if** $state(v) = INPATH$ **then**
9 **if** $u \neq nexthop(v)$ 并且$u \neq prehop(v)$ **then**
10 $cur\_len \leftarrow len\_ap(u) + l(u,v)$;
11 **if** $cur\_len < len\_ap(v_0)$ **then**
12 $phap(v_0) \leftarrow u$;
13 $len\_ap(v_0) \leftarrow cur\_len$;
14 发送FIND-PHAP消息$\{len_ap(v_0)\}$给$prehop(v)$;
15 **else if** $u = nexthop(v)$ **then**
16 $cur\_len \leftarrow len\_ap(u) - l(u,v)$;
17 **if** $cur\_len < len\_ap(v_1)$ **then**
18 $phap(v_1) \leftarrow u$;
19 $len\_ap(v_1) \leftarrow cur\_len$;
20 **if** $cur\_len < len\_ap(v_0)$ **then**
21 $phap(v_0) \leftarrow v_1$;
22 $len\_ap(v_0) \leftarrow len\_ap(v_1)$;
23 广播FIND-PHAP消息$\{len_ap(v_1)\}$;

算法 3-2 寻找最短增广路径

Algo. 3-2 Finding Shortest Augmenting Path

息$\{len_ap(v_0)\}$给$prehop(v)$（算法 3-2中第14行）。

当一个路径节点v接收到节点u发出的FIND-PHAP消息$\{len_ap(u)\}$时，如果u是v的下一跳节点，则在G^*中这个消息到达的是节点v_1。由于G^*中从u到v的边的长度等于$l(u,v)$的相反数，所以这条从s到u再到v_1的路径的长度为$len_ap(u) - l(u,v)$，我们用变量cur_len来记录之（算法 3-2中第16行）。如果$cur_len < len_ap(v_1)$，这条从s到u再到v_1的路径就是当前G^*中从s到v_1的最短路径，于是我们将$phap(v_1)$设置为u并更新$len_ap(v_1)$（算法 3-2中第17-19行）。接下来，在G^*中，v_1将广播FIND-PHAP消息，而这个消息将被v_0和v的所有外部邻居接收到。所以对于节点v_0，我们还需判断是否更新从s到v_0的最短路径。为了模拟这些操作，我们让节点v执行算法 3-2中第20-23行。

在OFDP的每次循环开始的时候，我们假设除s以外，网络中每个节点的len_ap都是正无穷。这可以通过以下的机制来实现。每个节点v保存着变量$round(v)$来指示当前循环的次数。在每次循环开始的时候，s节点将循环的次数更新，即在原来的循环次数上加1。这个循环次数被包含在每个FIND-PHAP消息中，这样网络中的所有节点都可以更新自己的$round(v)$。当v接收到一个FIND-PHAP消息并且其中的循环次数大于$round(v)$时，这表示当前$len_ap(v)$，或者$len_ap(v_0)$和$len_ap(v_1)$，等于正无穷。

3.4.2 回溯增广路径

假设上一步找到的G^*中最短的从s到t的路径是P_a^*。在这一步中，通过回溯相关节点的$phap$变量，我们将找到P_a^*在G中对应的增广路径P_a，并将P_a加到已发现的路径上来产生更多的不相交路径。这里，“加”表示取P_a和$P_1,\ldots,P_n$的对称差，即添加那些只属于P_a的边并删除那些P_a与$P_1,\ldots,P_n$共有的边。需要注意的是，对P_a的回溯是沿着从t到s的方向。在G^*中，这个回溯是沿着每条边的相反方向进行的。

回溯增广路径的伪代码由算法 3-3给出。整个过程由t节点向$phap(t)$发送一个TRACE-AP消息来初始化。

当一个非路径节点v接收到u发出的TRACE-AP消息时：这表明v是增广路径P_a上的一个非路径节点。通过将P_a加到已发现路径上，v应该成为一个路径节点（算法 3-3中第4行）。消息的发送者v应该成为新路径上v的下一跳节点（算法 3-3中第5行），v在增广路径P_a中的上一跳节点$phap(v)$应该成为新路径上v的上一跳节点（算法 3-3中第6行）。接着，v应该发送一个TRACE-AP消息给自己在增广路径上的上一跳节点$phap(v)$（算法 3-3中第3、20行）。

Input: 已经发现的不相交路径$P_1,\ldots,P_n$；当前的G^*中最短的从s到t的路径P_a^*

Output: 更多的不相交路径$P'_1,\ldots,P'_{n+1}$

/* 对于每个节点$v \neq t$的伪代码；整个过程由t节点向$phap(t)$发送一个TRACE-AP消息来初始化 */

```
1  while 接收到u发出的TRACE-AP消息 do
2      if state(v) = FREE then
3          dest ← phap(v);
4          state(v) ← INPATH;
5          nexthop(v) ← u;
6          prehop(v) ← phap(v);
7      if state(v) = INPATH then
8          if u = prehop(v) then
9              if phap(v_0) = v_1 then
10                 dest ← phap(v_1);
11                 state(v) ← FREE;
12             else
13                 dest ← phap(v_0);
14                 state(v) ← INPATH;
15                 prehop(v) ← phap(v_0);
16         else
17             dest ← phap(v_1);
18             state(v) ← INPATH;
19             nexthop(v) ← u;
20     发送TRACE-AP消息给dest;
```

算法 3-3 回溯增广路径

Algo. 3-3 Tracing Augmenting Path

如果一个路径节点v接收到u发出的TRACE-AP消息并且u是v的上一跳节点，在G^*中这个消息到达的是节点v_0。接下来，我们将分两种情况讨论。情况1：$phap(v_0) = v_1$。也就是在P_a^*中，v_0的上一跳节点是v_1。在P_a^*所对应的增广路径P_a中，v的上一跳节点应该是$phap(v_1)$。于是v发送一个TRACE-AP消息给$phap(v_1)$（算法 3-3中第10、20行）。根据G^*的特点，$phap(v_1)$一定等于$nexthop(v)$。因此，边$(prehop(v), v)$和$(v, nexthop(v))$都属于P_a。通过将P_a加到已发现路径上，v应该成为一个非路径节点（算法 3-3中第11行）。情况2：$phap(v_0) \neq v_1$。也就是在P_a^*中，v_0的上一跳节点不是v_1。在P_a^*所对应的增广路径P_a中，v的上一跳节点应该是$phap(v_0)$。于是v发送一个TRACE-AP消息给$phap(v_0)$（算法 3-3中第13、20行）。既然$phap(v_0) \neq v_1$，根据G^*的特点，$phap(v_0)$一定是v的某个外部邻居。因此，边$(prehop(v), v)$在P_a中而边$(v, nexthhop(v))$不在P_a中。通过将P_a加到已发现路径上，v应该成为一个路径节点（算法 3-3 中第14行）。v在新路径上的下一跳节点仍然是$nexthop(v)$，而v在新路径上的上一跳节点将变为$phap(v_0)$（算法 3-3中第15行）。

如果一个路径节点v接收到u发出的TRACE-AP消息并且u不是v的上一跳节点，在G^*中这个消息到达的是节点v_1。在增广路径P_a中，v的上一跳节点应该是$phap(v_1)$。于是v发送一个TRACE-AP消息给$phap(v_1)$（算法 3-3 中第17、20行）。根据G^*的特点，$phap(v_1)$一定等于$nexthop(v)$。因此，边$(prehop(v), v)$不属于P_a而边$(v, nexthop(v))$属于P_a。通过将P_a加到已发现路径上，v应该成为一个路径节点（算法 3-3中第18行）。v在新路径上的上一跳节点仍然是$prehop(v)$，而v在新路径上的下一跳节点将变为u（算法 3-3中第19行）。

3.4.3 OFDP的优点

与已有的集中式方法相比，OFDP具有小得多的通信代价，这将在 3.6 中被证实。除了这个优点以外，OFDP还有其它的优点。

OFDP是一条一条地增量地寻找k条不相交路径，所以它可以被用来应对k实时增长的情况。假设开始阶段，用户只需要2条不相交$s \sim t$路径，所以OFDP输出2条路径。过了一段时间，用户发现需要更多的不相交路径于是将k增加为3。这种情况下，OFDP只需要再执行一次循环，在网络中寻找一条增广路径，并将增广路径加到已知路径上产生3条不相交路径。随着需求的不断变换，用户可能多次的增加k，OFDP可以增量地处理这些变化。与之相反，即使$k = 1$，集中式方法也需要收集整个网络的拓扑信息，这必将会造成极大的资源浪费。

因其完全分布式的处理方式，OFDP可以更好地应对网络中的拓扑变化。在传感器网络中，节点死亡、链接失效、新节点加入、链接恢复等现象发生得十分频繁，所以网络的拓扑结构在不断地发生变化。为了保证答案正确性和结果最优性，集中式方法需要随着每个拓扑变化即时更新网络的拓扑结构，这样做必将会带来极其高昂的代价。与之相反，在发生拓扑变化时，OFDP在保证输出准确结果的同时不需要任何额外的代价。

3.4.4 OFDP正确性证明

G表示整个网络的拓扑图。让$P_1, \ldots, P_n$表示已经发现的长度之和最小的不相交$s \sim t$路径。在发现这些路径之后，如3.3中的描述，我们将G转换为G^*。

在一个网络中，给定节点s，我们称以下过程为"分布式建立以s节点为根的最短生成树"：树是通过节点间交换特定的消息来建立。每个消息中包含了当前从s节点到发送者的最短路径长度。首先，s节点广播一个消息{0}。当网络中的任意节点$v \neq s$接收到节点u发送的消息时，相当于新找到了一条从s到u再到v的路径。如果这条新路径的长度小于当前从s到v的最短路径长度，v将u设置为它的父亲，并广播一个包含这个新路径长度的消息。

引理 3.2 OFDP的第一步（寻找最短增广路径）等价于在一个拓扑图为G^*的网络中分布式建立以s节点为根的最短生成树。其中，$phap(v)$相当于节点v在树中的父亲。

证明： 在G转换为G^*的过程中，非路径节点没有发生变化。为了分布式建立以s节点为根的最短生成树，非路径节点正应该执行算法3-2中第2-7行的伪代码。

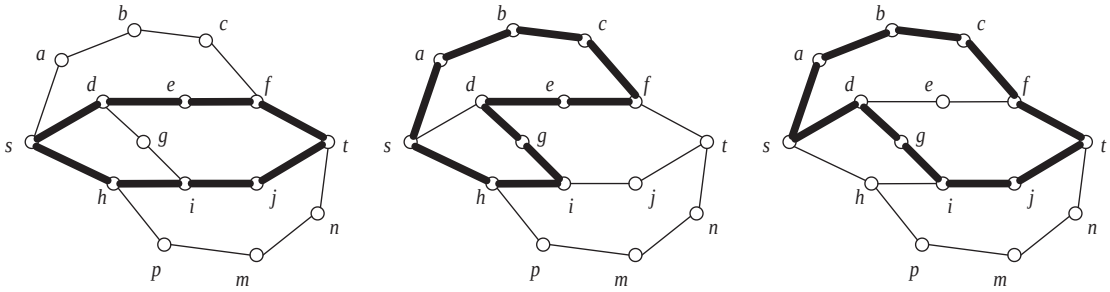
下面我们对每个路径节点v来证明该引理。让x和y分别表示v的上一跳和下一跳节点。在G^*中，v被分裂为两个节点v_0和v_1。v_1只有一条入边$< y_0, v_1 >$，而所有v_0的入边都是从v的外部邻居到v_0的。

当v接收到u发送的FIND-PHAP消息时，如果$u \neq x, y$，在G^*中这个消息到达的是节点v_0。为了在G^*中分布式建立以s节点为根的最短生成树，v_0正应该执行算法3-2中第9-14行的伪代码。

当v接收到u发送的FIND-PHAP消息时，如果$u = x$，在G^*中这个消息不应该被v_0或者v_1接收到。因此算法3-2在这种情况下不做任何处理。

当v接收到u发送的FIND-PHAP消息时，如果$u = y$，在G^*中这个消息到达的是节点v_1。为了在G^*中分布式建立以s节点为根的最短生成树，如果v_1更新了最短路径，它应该发送FIND-PHAP消息给v_0以及所有v在G中的外部邻居。

当v_0接收到v_1发送的这个FIND-PHAP消息后，如果v_0更新了最短路径，它应该发送FIND-PHAP消息给x。节点v执行的是算法 3-2中第15-23行的伪代码。这里，如果v_1更新了最短路径，我们直接让v广播一个FIND-PHAP消息M。对比于G^*中v_0和v_1的操作，M被节点y多余地接收了。如果v_0没有更新最短路径，M也被x多余地接收了。根据上一段的描述，节点y将会忽略M。如果v_0没有更新最短路径，x一定已经接收过v_0 发送的FIND-PHAP消息，因此当前从s到x的最短路径一定比从s到v_1再到x的路径短。这种情况下，x将会忽略消息M。 □



a) 已经找到的两条路径　　b) 环C　　c) 新的两条路径

a) The 2 found paths　　b) Cycle C　　c) The 2 new paths

图 3-4 环的例子

Fig.3-4 An example of cycle

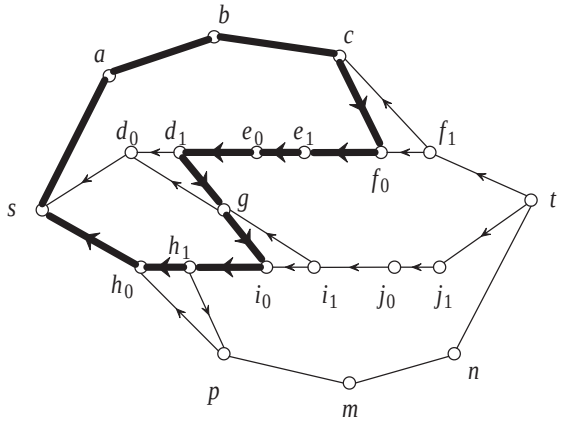


图 3-5 根据图 3-4 a) 的例子生成的G^*以及其中的环C^*

Fig.3-5 G^* generated by the example of Figure 3-4 a) and cycle C^* in it

引理 3.3 OFDP的第一步一定会停止。

证明： 根据引理 3.2，OFDP的第一步是一个在G^*中分布式建立以s节点为根的最短生成树的过程。要证明这个引理只需证明在G^*中从s到每个节点的最短路径长度不会被无限次更新，也就是G^*中不存在长度为负的环。

让C^*表示G^*中的任意一个环。通过将C^*在G中所对应的环C加到$P_1,\ldots,P_n$上，我们可以得到另一组不相交$s \sim t$路径$P'_1,\ldots,P'_n$。并且，$P'_1,\ldots,P'_n$的长度之和等于$P_1,\ldots,P_n$的长度之和加上C^*的长度。例如在一个如图 3-4 所示的图G中，我们发现了如图 3-4 a) 中粗线所示的2条路径$P_1 = (s,d,e,f,t)$和$P_2 = (s,h,i,j,t)$。接着，我们将G转换为如图 3-5 所示的G^*，图中的粗线表示G^*中的环C^*。C^*在G中对应的环是$C = (s,a,b,c,f,e,d,g,i,h,s)$，如图 3-4 b) 中粗线所示。通过将$C$加到$P_1$和$P_2$上，我们可以得到两条新的路径$P'_1 = (s,a,b,c,f,t)$和$P'_2 = (s,d,g,i,j,t)$，如图 3-4 c)中粗线所示。在$G^*$中，边$< f_0,e_1 >$、$< e_0,d_1 >$和$< i_0,h_1 >$的长度等于原来边长度的相反数，边$< e_1,e_0 >$和$< h_1,h_0 >$的长度为0，其它边的长度不变。因此，$P'_1$和$P'_2$的长度之和等于$P_1$和$P_2$ 的长度之和加上C^*的长度。

假设G^*中存在长度非负的环C^*。通过将C^*加到已发现路径$P_1,\ldots,P_n$上，我们可以得到n条长度之和更小的不相交路径。这与$P_1,\ldots,P_n$是长度之和最小的n条不相交路径的事实不符。 □

引理 3.4 让P^*_a表示G^*中最短的从s到t的路径，P_a表示P^*_a在G中所对应的增广路径。于是，OFDP的第二步（回溯增广路径）是一个将P_a加到$P_1,\ldots,P_n$上的过程。

证明： 如果一个非路径节点v接收到节点u发送的TRACE-AP消息，这表示节点v是增广路径P_a上的非路径节点。通过将P_a加到$P_1,\ldots,P_n$上，v将成为一个新的路径节点。v在新路径上的上一跳节点是$phap(v)$，下一跳节点是u。这正对应了算法 3-3 中第2-6行的伪代码。

下面我们让v表示任意路径节点，x和y分别表示v的上一跳和下一跳节点。

如果v接收到u发送的TRACE-AP消息并且$u = x$，边(x,v)在增广路径P_a上。在G^*中这个消息到达的是节点v_0。下面分两种情况讨论。情况1：$phap(v_0) = v_1$。那么v在P_a上的上一跳节点应该是$phap(v_1)$。根据G^*的特点，一定有$phap(v_1) = y$，所以边(v,y)也在P_a上。这种情况下，通过将P_a加到$P_1,\ldots,P_n$上，v将成为一个非路径节点，正对应了算法 3-3中第8-11行的伪代码。情况2：$phap(v_0) \neq v_1$。那么v在P_a上的上一跳节点一定是v的某个外部邻居，所以边(v,y)不在P_a上。这种情况下，通过将P_a加到$P_1,\ldots,P_n$上，v仍然是一个路径节点，它的下一跳节点没有变化，它在新路径上的上一跳节点是$phap(v_0)$，正对应了算法 3-3中第12-15行的伪代码。

如果v接收到u发送的TRACE-AP消息并且$u \neq x$，边(x,v)不在增广路径P_a上。通过将P_a加到$P_1,\ldots,P_n$上，v仍然是一个路径节点，它的上一跳节点没有变化，

它在新路径上的下一跳节点是u，正对应了算法 3-3中第16-19行的伪代码。 □

定理 3.1 OFDP一定会终止，并且能够保证答案正确性和结果最优性。

证明： 根据定理 2.1 和引理 3.1，我们可以通过以下方法寻找k条长度之和最小的不相交$s \sim t$路径，所得到的结果既能保证答案正确性又能保证结果最优性：循环地在当前的G^*中寻找最短的从s到t的路径P_a^*；将P_a^*在G中对应的增广路径P_a加到已发现路径上产生更多的不相交路径。

根据引理 3.2，在每次循环中，OFDP的第一步正是在G^*中寻找最短的从s到t的路径P_a^*。根据引理 3.3 这个过程一定会终止。根据引理 3.4，每次循环中，OFDP的第二步正是将P_a加到已发现的路径上。 □

3.5 完全分布式算法FDP

```
Input: 两个节点s和t；正整数k
Output: min{k, k*}条不相交s ~ t路径
1 repeat
2     调用寻找增广路径算法;
3     if 找不到增广路径 then
4         停止并输出所有已发现的路径;
5     调用回溯增广路径算法;
6 until 找到k条不相交s ~ t路径;
```

算法 3-4 FDP
Algo. 3-4 FDP

如果用户不需要优化路径的长度，那么本章研究的问题就变为了与第 2 章中相同的k不相交路径问题：给定网络中的节点s和t 以及正整数k，寻找k条不相交的$s \sim t$路径。根据本章中对于增广路径的新的定义，我们可以简化OFDP，得到一个可以解决k 不相交路径问题的完全分布式算法FDP。FDP可以保证输出$\min\{k, k^*\}$条不相交$s \sim t$路径。其中，网络中最多存在k^*条不相交$s \sim t$路径。

FDP基本思想仍然是循环地寻找当前的G^*中的从s到t的路径P_a^*（P_a^*在G中对应了增广路径P_a），然后将增广路径P_a添加到已知路径上构建更多的不相交路径，直至找到k条不相交路径为止。只不过不再要求P_a^*是G^*中最短的从s到t的路径，而是任意一条从s到t的路径。FDP在每个循环中仍然有两步：寻找增广路径

（第一步）和回溯增广路径（第二步）。第一步负责寻找当前G^*中的任意一条从s到t的路径P_a^*。第二步与OFDP的第二步相同，负责将P_a^*对应的增广路径P_a加到已发现路径上。整个算法由算法 3-4给出。

让v表示网络中的任意节点。与OFDP相比，v不需要知道它的每个邻居u以及链接$l(u,v)$的代价。此外，无论v是路径节点还是非路径节点，它不用再记录从s到v的最短路径长度，也就是不用保存len_ap变量。

3.5.1 寻找增广路径

Input: 已经发现的不相交路径$P_1,\ldots,P_n$
Output: 当前的G^*中一条从s到t的路径P_a^*
/* 对于每个节点$v \neq s$的代码；整个过程由s节点广播FIND-PHAP消息来初始化 */
while 接收到u发出的*FIND-PHAP*消息 **do**
  if $state(v) = FREE$ **then**
    if $phap(v) = NULL$ **then**
      $phap(v) \leftarrow u$;
      广播FIND-PHAP消息;
  if $state(v) = INPATH$ **then**
    if $phap(v_0) = NULL$ *and* $u \neq nexthop(v)$ *and* $u \neq prehop(v)$ **then**
      $phap(v_0) \leftarrow u$;
      发送FIND-PHAP消息给$prehop(v)$;
    else if $phap(v_1) = NULL$ *and* $u = nexthop(v)$ **then**
      $phap(v_1) \leftarrow u$;
      if $phap(v_0) = NULL$ **then**
        $phap(v_0) \leftarrow v_1$;
      广播FIND-PHAP消息;

算法 3-5 寻找增广路径
Algo. 3-5 Finding Augmenting Path

在这一步中，我们在当前的G^*中寻找一条从s到t的路径P_a^*。通过与其邻居交换FIND-PHAP 消息，每个节点v可以获得它的$phap$变量的赋值。这一步相当于在G^*中建立以s节点为根的生成树（不需要是最短生成树）。变量$phap$相当于

节点在树中的父亲。与OFDP的第一步相比，每个FIND-PHAP消息不再需要包含发送者的len_ap，而只需要包含发送者的ID。

寻找增广路径的伪代码由算法 3-5给出。整个过程由s广播FIND-PHAP消息来初始化。

当一个非路径节点v第一次接收到FIND-PHAP消息时：v将$phap(v)$设置为消息的发送者并广播FIND-PHAP消息（算法 3-5中第2-5行）。

当一个路径节点v接收到节点u发出的FIND-PHAP消息时，如果u不是v的上一跳或者下一跳节点，则这个消息是沿着v的一条外部边(u,v)到达v的。在G^*中，这个消息到达的是节点v_0。如果$phap(v_0)$还没有被赋值，v就将$phap(v_0)$设置为u（算法 3-5中第7-8行）。因为在G^*中v_0只有一个出边到v的上一跳节点$prehop(v)$，所以v发送FIND-PHAP消息给$prehop(v)$（算法 3-5中第9行）。

当一个路径节点v接收到节点u发出的FIND-PHAP消息时，如果u是v的下一跳节点，则在G^*中这个消息到达的是节点v_1。如果$phap(v_1)$还没有被赋值，$phap(v_1)$将会被设置为u（算法 3-5中第10-11行）。在G^*中，v_1将广播FIND-PHAP消息，而这个消息将被v_0和v的所有外部邻居接收到。如果$phap(v_0)$还没有被赋值，$phap(v_0)$将被设置为v_1。为了模拟这些操作，我们让节点v执行算法 3-5中第12-14行。

在FDP的每次循环开始的时候，我们假设除s以外，网络中每个节点的$phap$都未被赋值，也就是等于NULL。这可以通过以下的机制来实现。每个节点v保存着变量$round(v)$来指示当前循环的次数。在每次循环开始的时候，s节点将循环的次数更新，即在原来的循环次数上加1。这个循环次数被包含在每个FIND-PHAP消息中，这样网络中的所有节点都可以更新自己的$round(v)$。当v接收到一个FIND-PHAP消息并且其中的循环次数大于$round(v)$时，这表示当前$phap(v)$，或者$phap(v_0)$和$phap(v_1)$，等于NULL。

FDP的第二步与OFDP的第二步相同，同样可以由算法 3-3给出，所以我们就不再次描述。

3.5.2 FDP的优点及正确性

FDP可以保证答案正确性。FDP的第一步是一个在G^*中分布式建立以s节点为根的生成树的过程，如果G^*存在一条从s到t的路径，FDP一定会在第一步中找到这样一条路径P_a^*。每个节点v都将自己第一次接收到的FIND-PHAP消息的发送者作为$phap(v)$，也就是在v树中的父亲。如果$phap(v)$已经被赋值，v将忽略接收到的其它FIND-PHAP消息。这样的机制将保证这个建立树的过程一定会终

止。FDP的第二步是一个将P_a^*所对应的增广路径P_a加到已知路径上的过程。因此，根据定理 2.1，FDP可以保证答案正确性。

与第 2 章中同样可以保证答案正确性的分布式算法EDA相比，FDP不依赖于任何预先建立的结构，也不需要收集任何网络信息，是一个完全的分布式方法。因此，FDP可以更好地适应网络的拓扑变化。3.6 中的实验结果也表明，大多数情况下，FDP具有更高的通信效率。

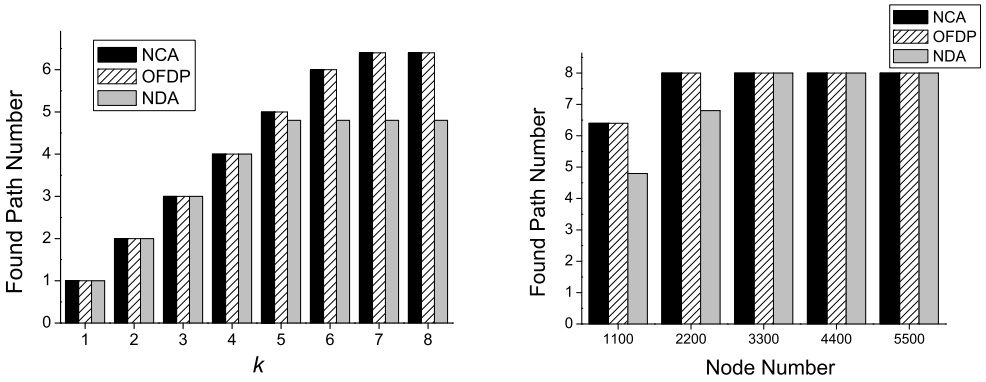
3.6 实验结果

我们用一个C++编写的模拟器来验证算法的效率。这个模拟器是一个基于应用层的程序，忽略了MAC层的细节。在实验中，我们在一个1000m ×1000m的监测区域内布置传感器节点。每个节点的位置随机产生。每个节点的通信半径被设置为50m。当两个节点间距离小于50m时，节点间存在无线链接。每个链接的长度（代价）是1到10之间的随机整数。每个数据包包含了4个字节的包头来分别表示这个数据包的：目的节点的ID；发送者ID；数据包类型（例如：FIND-PHAP消息、TRACE-AP消息等）；数据包长度。每个接收数据和时钟到时事件被加上时间戳统一放到处理事件的堆中。堆中的事件按其时间戳的先后顺序一个一个进行处理。通过这种方式，我们来模拟网络中节点间的并行操作。每个实验数据都是10次实验（每次都设置不同的s和t）结果的平均值。

算法的效率通过三个方面来衡量：算法找到的路径数量、算法找到的路径的长度之和以及算法的通信代价。算法的通信代价由实验中总的通信量（网络中信息交换的字节数）除以节点的数量得到，也就是平均每个节点接收和发送的字节数。根据1.2.1 中的描述，通信代价可以很好地表示节点的能量消耗。

针对最优k不相交路径问题，我们将OFDP与另外两个算法相比较，它们是：文献[141]中的集中式算法（用NCA表示）和文献[131]中的分布式方法（用NDA表示）。NCA将整个网络的拓扑信息收集到基站节点，然后在基站节点用文献[141]中的方法来寻找长度之和最小的k条不相交$s \sim t$路径。该算法既能保证答案正确性又能保证结果最优性。NDA分布式地在网络中寻找不相交路径。每找到一条路径，NDA就将这条路径固定下来并将路径上的节点从网络中删除。在文献[127–134]给出的这些分布式方法中，NDA是最具通信效率的一个。这里，我们对NDA做一个小小的改动，让其在每次执行中都以贪心的方式寻找当前长度最短的路径。需要注意的是NDA既不能保证答案正确性也不能保证结果最优性。

首先，我们来比较三个算法找到的路径数量。为此我们设计了两组实验。在第一组实验中，我们在监测区域内（1000m×1000m）布置了1100个节点，因此每个节点平均有8.6个邻居。我们根据不同的k，也就是用户需要找到的路径数量，来比较这三个算法。实验结果如图 3-6 a) 所示。因为OFDP和NCA 都可以保证答案的正确性，也就是它们都能够找到网络中最多的不相交$s \sim t$路径，所以它们找到的路径数量都相同。由于NDA采取了简单的贪心策略，当$k > 4$时，它找到的路径明显比其它两种方法要少，不能保证答案正确性的劣势显示出来。在第二组实验中，我们固定$k = 8$，并在监测区域内分别布置了1100、2200、3300、4400、5500个节点，来比较这三个算法找到的路径数量。实验结果如图 3-6 b) 所示。当节点数量为1100, 2200时，对比于NCA和OFDP，NDA找到的路径数量要更少，这正是不能保证答案正确性的劣势。当节点数量大于3300时，这三个方法都能找到网络中的8条不相交$s \sim t$路径。如果k被固定下来，在一个非常密集的网络中，OFDP和NCA 在路径数量方面的优势不再那么明显。但是，在实际应用中，人们很难有能力布置如此密集的网络（例如在1000m×1000m的区域内布置3000 ~ 5000个节点）。



a) 第一组实验　　b) 第二组实验

a) Group 1　　b) Group 2

图 3-6 NCA、OFDP和NDA找到的路径数量

Fig.3-6 Number of the paths found by NCA, OFDP and NDA

接着，我们来比较这三个算法找到的路径的长度之和。这三个算法找到的路径数量有可能不相等，所以公平的比较办法应该是当这三个算法找到相同数量的路径时，比较它们找到的路径的长度之和。为此我们设计了三组实验。在第一组实验中，我们在监测区域内布置1100个节点，固定$k = 4$。在这种情况下，三个算法都能找到4条不相交路径。在第二组实验中，我们在监测区域内

布置2200个节点，固定$k = 6$。在这种情况下，三个算法都能找到6条不相交路径。在第三组实验中，我们在监测区域内布置3300个节点，固定$k = 8$。在这种情况下，三个算法都能找到8条不相交路径。实验结果如图 3-7 所示。由于都能保证结果最优性，NCA和OFDP找到的路径都有相同的总长度。由于不能保证结果最优性，相比于其它两个算法，NDA找到的路径的长度之和明显更大。在找到的路径的质量方面，NDA的劣势显露无疑。

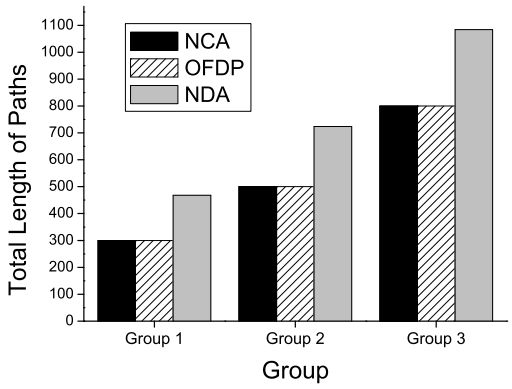
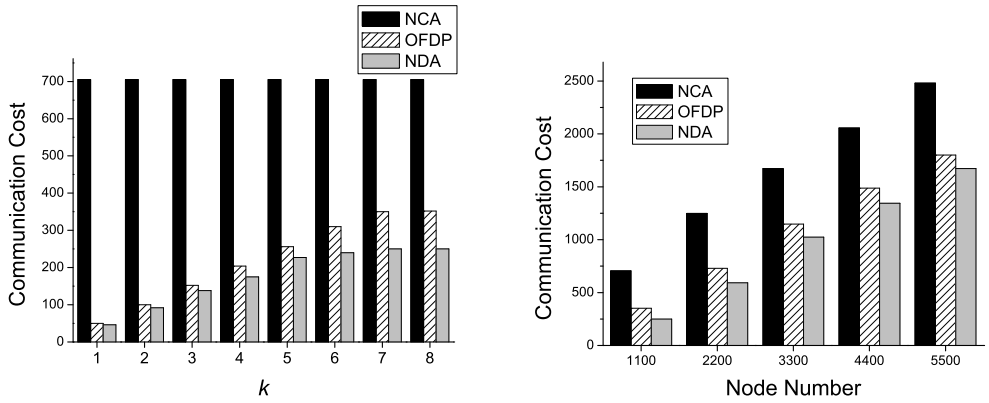


图 3-7 NCA、OFDP和NDA找到的路径的长度之和

Fig. 3-7 Total length of the paths found by NCA, OFDP and NDA

最后，我们比较这三个算法的通信代价。这个通信代价用平均每个节点接收和发送的字节数来衡量。为此我们进行了两组实验。在第一组实验中，我们在监测区域内布置了1100个节点，根据不同的k来比较这三个算法。实验结果由图 3-8 a) 给出。无论k多大，NCA都需要收集整个网络的拓扑信息，所以它的通信代价一直稳定。OFDP和NDA都是增量的分布式算法，所以它们的通信代价随着找到路径数量的增加而增加。由于采用分布式的处理方式，这两种方法的通信代价远远小于NCA的通信代价。当$k \le 4$时，OFDP的通信代价仅仅比NDA的通信代价略高。当$k > 4$时，OFDP与NDA在通信代价上的差距逐渐明显，这是因为OFDP找到了更多的路径。OFDP既能保证答案正确性又能保证结果最优性。无论是在找到的路径数量方面还是在路径的总长度方面，NDA的表现都远远不如OFDP，因此OFDP比NDA多消耗一些通信是完全可以接受的。在第二组实验中，我们固定$k = 8$，并在监测区域内分别布置了1100、2200、3300、4400、5500个节点，来比较这三个算法的通信代价。实验结果如图 3-8 b) 所示。这组实验再次验证了OFDP的通信效率。

针对k不相交路径问题，也就是不考虑路径的长度，我们将FDP与其它三个算法相比较，它们是：第2章中的算法EDA、文献[141]中的集中式算法（用NCA表示）和文献[131]中的分布式方法（用NDA表示）。算法的效率通过三



a) 第一组实验　　b) 第二组实验

a) Group 1　　b) Group 2

图 3-8 NCA、OFDP和NDA的通信代价

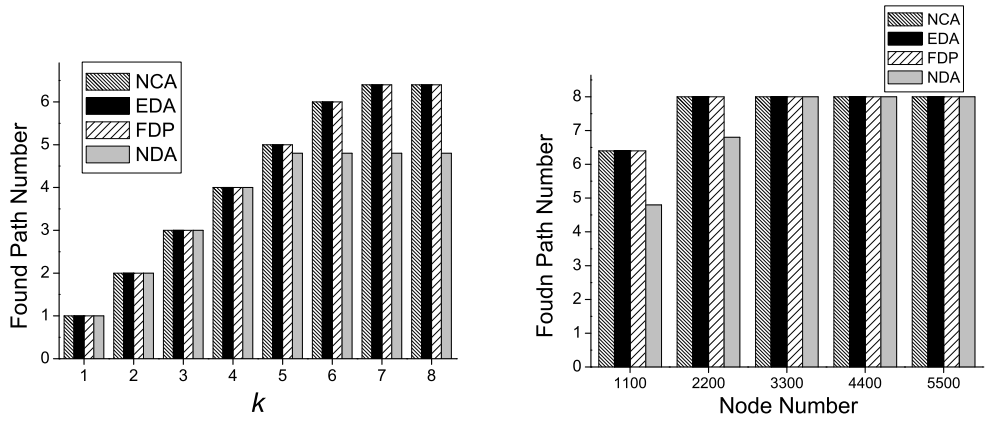
Fig.3-8 Communication cost of NCA, OFDP and NDA

个方面来衡量：算法找到的路径数量、算法的通信代价以及算法为找到每条路径消耗的通信代价。算法的通信代价由实验中总的通信量（网络中信息交换的字节数）除以节点的数量得到，也就是平均每个节点接收和发送的字节数。算法为找到每条路径消耗的通信代价由算法的通信代价除以算法找到的路径数量得到。NCA是文献[135–137, 139, 141]中这些集中式方法的一个代表。因为这些集中式方法都需要收集整个网络的拓扑信息，所以它们有相同的通信代价。另一方面，这些方法都可以保证答案正确性，所以它们找到的路径数量相同。

我们设计了两组实验来比较这些算法。在第一组实验中，我们在监测区域内布置了1100个节点，根据不同的k，来比较这四个算法。在第二组实验中，我们固定$k = 8$，分别在监测区域内布置1100、2200、3300、4400、5500个节点，来比较这四个算法。

首先，我们比较这四个算法找到的路径数量。第一组实验的结果由图 3-9 a)给出。因为NDA、EDA和FDP都可以保证答案正确性，所以它们找到的路径的数量都相同。当$k > 4$时，相比于其它三个算法，NDA找到的路径要更少，其不能保证答案正确性的劣势显现出来。第二组实验的结果由图 3-9 b) 给出。在节点密度不是非常大的时候，NDA的劣势依旧比较明显。

接着，我们来比较这四个算法的通信代价。第一组实验的结果由图 3-10 a)给出。NCA由于其集中式的处理方式所以通信代价比较稳定。因为是增量的分布式算法，EDA、FDP和NDA的通信代价随着k的增长而增长，直到它们找到网络中最多的路径为止。相比于EDA，FDP的通信代价更接近于最简单



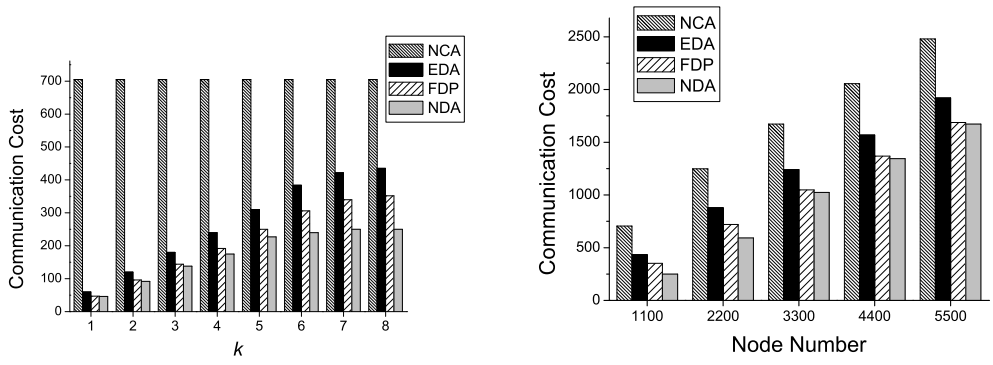
a) 第一组实验　　b) 第二组实验

a) Group 1　　b) Group 2

图 3-9 NCA、EDA、FDP和NDA找到的路径数量

Fig. 3-9 Number of the paths found by NCA, EDA, FDP and NDA

的NDA算法。当$k > 4$时，FDP与NDA在通信代价上的差距比较明显，这是因为FDP找到了更多的路径，这也正是FDP可以保证答案正确性的优势。第二组实验结果由图 3-10 b) 给出。在通信效率方面，FDP比EDA更具优势，其通信代价甚至接近于不能保证答案正确性的NDA算法。



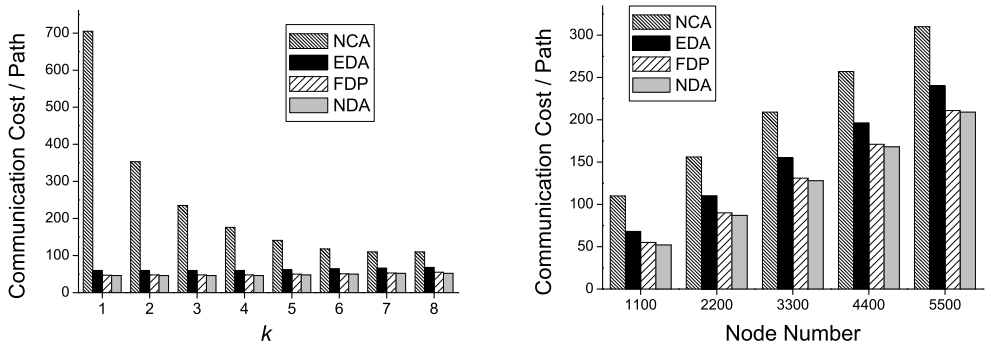
a) 第一组实验　　b) 第二组实验

a) Group 1　　b) Group 2

图 3-10 NCA、EDA、FDP和NDA的通信代价

Fig. 3-10 Communication cost of NCA, EDA, FDP and NDA

最后，我们来比较这四个算法为找到每条路径所消耗的通信代价，这也是一个更为公平的比较通信代价的办法。两组实验的结果分别由图 3-11 a) 和图 3-11 b) 给出。从中可以看出，相较于EDA，FDP在通信效率方面更具优势。



a) 第一组实验　　b) 第二组实验

a) Group 1　　b) Group 2

图 3-11 NCA、EDA、FDP和NDA为找到每条路径消耗的通信代价

Fig.3-11 Communication cost to find each path for NCA, EDA, FDP and NDA

3.7 本章小节

本章研究了最优k不相交路径问题，即给定网络中的节点s和t、正整数k，寻找网络中长度之和最小的k条不相交$s \sim t$路径。关于这个问题本章给出了一个高效的分布式算法OFDP。OFDP既能找到网络中最多的不相交路径又能保证找到路径的长度之和是所有可能结果中最小的。与现有的集中式方法相比，由于采用完全分布式的处理方式，OFDP具有更高的通信效率。通过对OFDP进行简化，我们提出了一个完全分布式的算法FDP用来解决k不相交路径问题，即不考虑路径长度的不相交路径问题。FDP可以保证答案正确性，也就是能找到网络中最多的不相交路径。与第2章中提出的算法EDA相比，FDP的分布式处理不依赖任何结构，也不需要收集任何信息，因此具有更高的通信效率。

第 4 章 无线传感器网络中长度受限的不相交多路径路由

4.1 引言

传感器网络是一个多跳自组织的网络。因此，如果网络中的s节点要向t节点传送数据，这个数据的传输通常要沿着一条连接s和t的路径进行多次转发。传统的路由策略就是在给定的两个节点间建立一条用于数据传输的优化路径。但是由于传感器网络中的功能有限性和不确定性，只采用一条路径传输数据会遇到很多问题。例如，由于网络中的每个无线链接的带宽十分有限，如果要在给定节点间高速传输数据，只用一条路径就不能达到要求；由于网络中的节点和链接经常失效或发生错误，只用一条路径传输数据很可能造成数据的丢失；如果两个节点使用一条路径大量传输数据，这条路径上节点的能量将会被很快地用光。针对上述问题，研究者们提出了在给定节点间用多条不相交路径来进行路由的策略[123–125, 127–134]。如 2.1节所述，这种路由策略可以有效的提升路由的吞吐量、可靠性、负载平衡和安全性。

网络中的每个无线链接都有一定的传输延迟。因此，如果某条路径上的边（无线链接）过多，用其传输数据的时间延迟对用户来说就可能变得无法忍受。另外，由于传感器节点功能简单，每次无线传输都有一定的丢包率。如果某条路径上的边过多，数据沿着这条路径到达目的地的概率就会变得非常小，从而使这条路径变得没有意义。因此，在大多数应用中，都有必要对用于路由的路径的长度加以限制。本章中我们研究的问题是长度受限的不相交路径问题：给定网络中的节点s和t以及长度限制L，寻找最多的长度小于等于L的不相交$s \sim t$路径。假设给定网络的拓扑图为$G = (V, E)$。长度受限的不相交路径问题又可以表示为图中的问题。

文献[146]证明了当$L \geq 5$时，这个问题是一个NP难问题。文献[147]进一步证明了当$L \geq 5$时，这个问题是一个APX完全问题，也就是没有关于这个问题的多项式时间近似模式。因此，我们不可能以任意的近似比来近似这个问题，并且也不太可能以常数的近似比来近似这个问题。据我们所知，迄今为止，关于这个问题，只有一个启发式算法被提出[148]。并且由于采用了集中式的处理方式，这个启发式算法不适合应用在传感器网络中。

在本章中，针对长度受限的不相交路径问题，我们首先提出了一个$\sqrt{n}$近似算法GDA，并给出了这个算法的分布式实现。其中，n为G中节点的数量。此外，我们还给出了一个分布式的启发算法OPDA。实验结果表明，在找到的路径数量和通信效率方面，OPDA都有很好的表现。

在以上问题的描述中，路径的长度是用路径上边的数量来衡量的。有时，我们需要用更复杂的标准，例如能量消耗或者传输延迟，来衡量路径的长度。假设网络中的每个无线链接（图G中的每条边e）都有一个代价$l(e) \in \mathbb{R}$来表示其能量消耗或者传输延迟。一条路径的长度是其上所有边的代价之和。在这种情况下，我们称图G为一个加权图。在实际应用中，用户希望用于路由的路径的能量代价或者传输延迟小于指定的阀值。只需要做小小的改动，OPDA就可以很好地解决加权图中长度受限的不相交路径问题。

本章的贡献在于：

- 针对长度受限的不相交路径问题，提出了一个$\sqrt{n}$近似算法GDA，并给出了GDA的分布式实现。
- 针对长度受限的不相交路径问题，提出了一个分布式的启发算法OPDA。OPDA具有很高的效率并且能处理加权图的情况。

本章的内容安排如下：4.2介绍了相关的研究工作；4.3给出了近似算法GDA以及近似比的分析；4.4给出了分布式的启发算法OPDA；4.5 讨论了加权图的情况；4.6通过实验结果验证了所提出算法的优越性；4.7总结全章。

4.2 相关工作

在图论领域，有一些关于长度受限的不相交路径问题的研究。文献[146]证明了当$L \geq 5$时，这个问题是一个NP难问题。文献[147]进一步证明了当$L \geq 5$时，这个问题是一个APX完全问题。在这种情况下，没有关于这个问题的多项式时间近似模式，也很难给出这个问题的近似比为常数的算法。一个集中式的启发算法在文献[148] 中被提出。该算法逐条的寻找长度小于等于L的$s \sim t$路径并删除其上节点。如果找不到长度小于等于L的路径，该算法尝试破坏某条已发现的路径P_i来产生一条新的长度小于等于L的路径P_i'，但必须保证P_i'上的长度大于P_i的长度。这样直到产生不了任何新的路径为止。该算法可以保证当$L \leq 4$时，算法输出的是最优解。当$L \geq 5$时，该算法没有一个可分析的下界。由于采用集中式的处理方式，如果我们将这个算法应用到传感器网络中，我们将不得不收集整个网络的拓扑信息，而这项操作的通信代价对于传感器网络来说过于高昂。此外，这个算法也不能很好应对加权图中长度受限的不相交路径问题。

如果给定的图是一个有向加权图，并且需要的路径是边不相交路径，那么本章研究的问题就成为了有向加权图中长度受限的边不相交路径问题。文献[149]证明了对于任意的$\epsilon > 0$，以$m^{1/2-\epsilon}$来近似有向加权图中长度受限的边不相交路径问题是NP难的。此外，该文献还给出了这个问题的一个$\sqrt{m}$近似算法。算法的基本思想是逐条寻找长度小于等于L但是其上边数最少的$s \sim t$路径，并将路径的边从网络中删除。其中的核心子程序就是寻找当前网络中长度小于等于L但是其上边数最少的$s \sim t$路径。这个程序的运算十分复杂，使得我们不可能将其分布式地实现于传感器网络中。因此，即使修改这个算法用于解决本章中的问题，算法也不适合于传感器网络。

在网络领域，许多的分布式方法[127–134]被提出用来寻找k条不相交$s \sim t$路径。但是这些方法都没有考虑到路径的长度。

在图论领域，文献[135–137, 139, 141]提出了集中式方法用来寻找k条不相交$s \sim t$路径。其中，文献[137, 139, 141]中的算法可以找到长度之和最小的k条不相交路径。即使这样，这些算法的结果中仍然不能避免出现一些长度过大而不适合用于路由的路径。此外，集中式的处理方式也使这些算法不适合于传感器网络。

对于k条$s \sim t$路径，我们给每条路径一个序列号，由此区分这些路径为第1条、第2条、...、第k条路径。网络中的每条边（链接）都有k个长度分别表示其位于第1条、第2条、...、第k条路径中的代价。这样的网络被称作多代价网络。文献[142–145]给出了集中式算法在多代价网络中寻找k条长度之和最小的不相交$s \sim t$路径。

到目前为止，关于长度受限的不相交路径问题，只有一个集中式的启发算法[148]被提出。由于其集中式的处理方式，该算法也不适合应用在传感器网络中。虽然我们可以通过修改文献[149] 中的算法来解决长度受限的不相交路径问题，但是算法的计算过于复杂，而且必需以集中式的方式进行，因此同样不适合应用在传感器网络中。

4.3 近似算法GDA

假设给定网络的拓扑图为无向连通图$G = (V, E)$，其中节点集合为$V = \{v \mid v$是网络中的节点$\}$，边集合为$E = \{(u, v) \mid u$和v之间存在无线链接$\}$。定义$n = |V|$和$m = |E|$。对于网络中的某条路径P，它的长度$l(P)$为P上边的数量。已经找到的$s \sim t$路径上的节点被称作路径节点，其它节点被称作非路径节点。

在算法的描述中，“节点v广播一个消息”表示这个消息可以被v的所有邻居接收到，正如传感器网络中的广播。“节点v发送一个消息给节点u”表示这个消息只能被u接收到，v的其它邻居接收不到这个消息，正如传感器网络中的单播。

4.3.1 算法的描述

这里，我们给出用来解决长度受限的不相交路径问题的近似算法GDA。给定网络中的节点s和t以及长度限制L，GDA将会输出多条长度小于等于L的不相交路径。GDA由算法 4-1给出。GDA循环地寻找最短的$s \sim t$路径P。如果P的长度小于等于L，则输出P并从网络中删除P的中间节点使它们不能参与到其它路径中。算法执行到再也找不到长度小于等于L的$s \sim t$路径为止。

Input: 两个节点s和t；长度限制L
Output: 一些长度小于等于L的不相交$s \sim t$路径

```
1 repeat
2     找到最短的s ~ t路径P;
3     if l(P) ≤ L then
4         输出P并从网络中删除P上的中间节点;
5     else
6         停止;
7 until 不存在s ~ t路径;
```

算法 4-1 GDA
Algo. 4-1 GDA

为了分布式地实现GDA，网络中的每个节点v保存了三个变量$state(v)$、$len(v)$和$lasthop(v)$。其中，$len(v)$表示当前从s到v的最短路径的长度，$lasthop(v)$表示这条最短路径上v的上一跳节点（距离s更近的节点）。变量$state \in \{FREE, INPATH\}$指示了节点$v$是否是路径节点。

GDA中的“找到最短的$s \sim t$路径P”可以由算法 4-2来分布式地实现。网络中的节点通过交换FIND-PATH消息来寻找从s到每个节点的最短路径。每个FIND-PATH 消息的包含了发送者的len变量信息。整个过程由s节点广播FIND-PATH消息{0}来初始化。如果非路径v发现了更短的从s到v的路径，v就更新其$len(v)$和$lasthop(v)$，并广播FIND-PATH消息{$len(v)$}。

GDA中“输出P并从网络中删除P上的中间节点”可以由算法 4-3来分布式

```
Input: 一个传感器网络；网络中的节点s和t
Output: 网络中最短的s ~ t路径
/* 对于网络中每个节点v的伪代码 */
1 if v是节点s then
2     广播FIND-PATH消息{0};
3 else
4     while 接收到u发送的FIND-PATH消息{len(u)} do
5         if state(v) = FREE 并且len(u) + 1 < len(v) then
6             len(v) ← len(u) + 1;
7             lasthop(v) ← u;
8             广播FIND-PATH消息{len(v)};
```

算法 4-2 寻找最短$s \sim t$路径

Algo. 4-2 Finding Shortest $s \sim t$ Path

```
Input: 一个传感器网络；网络中一条从s到t的路径P
Output: 一个删除了P上中间节点的网络
/* 对于网络中每个节点v的伪代码 */
1 if v是节点t then
2     if len(t) ≤ L then
3         发送TRACE-PATH消息给lasthop(t);
4 else
5     while 接收到TRACE-PATH消息 do
6         state(v) ← INPATH;
7         发送TRACE-PATH消息给lasthop(v);
```

算法 4-3 回溯路径

Algo. 4-3 Tracing Path

地实现。首先，节点t判断当前最短$s \sim t$路径的长度是否小于等于L。如果是，t就发送TRACE-PATH消息来建立这条路径。每个接受到TRACE-PATH消息的节点v将自己标记为路径节点，并发送TRACE-PATH消息给$lasthop(v)$。

4.3.2 近似比分析

现在，让我们来证明GDA的近似比。

定理 4.1 GDA的近似比为$\sqrt{n}$。

证明： 给定一个长度受限的不相交路径问题的实例，让$\mathcal{O}$表示其最优解，$\mathcal{A}$表示GDA输出的解。对于$\mathcal{O}$中的每条路径Q来说，它一定与$\mathcal{A}$中的某条路径共享某些中间节点，否则将Q添加到$\mathcal{A}$中可以继续扩展$\mathcal{A}$。对于任意一条$s \sim t$路径P来说，其上中间节点的数量为$l(P) - 1$。

在$\mathcal{A}$中，$s \sim t$路径P_i表示GDA在第i次循环中产生的路径。让$\mathcal{O}_i$表示$\mathcal{O}$中那些和P_i共享中间节点但是不和$P_1, \ldots, P_{i-1}$共享中间节点的路径。定义$x_i = |\mathcal{O}_i|$。于是，我们有$|\mathcal{O}| = \sum_{i=1}^{|\mathcal{A}|} x_i$。

对于每条P_i来说，$\mathcal{O}$中与其共享中间节点的路径的数量不会多于P_i上中间节点的数量，所以有

$$x_i = |\mathcal{O}_i| \leq l(P_i) - 1 \tag{4-1}$$

对于$\mathcal{O}_i$中的每条路径Q，一定有

$$l(Q) \geq l(P_i) \tag{4-2}$$

否则Q将取代P_i被GDA在第i次循环中输出。

结合公式4-1和4-2，对于每个$\mathcal{O}_i$，有

$$\sum_{Q \in \mathcal{O}_i} [l(Q) - 1] \geq x_i^2 \tag{4-3}$$

由于最优解$\mathcal{O}$中任意两条路径都不共享任何中间节点，所以有

$$\sum_{Q \in \mathcal{O}} [l(Q) - 1] = \sum_{i=1}^{|\mathcal{A}|} \sum_{Q \in \mathcal{O}_i} [l(Q) - 1] \leq n - 2 < n \tag{4-4}$$

这样，我们可以得到

$$n > \sum_{i=1}^{|\mathcal{A}|} x_i^2 \geq \frac{(\sum_{i=1}^{|\mathcal{A}|} x_i)^2}{|\mathcal{A}|} \geq (\frac{\sum_{i=1}^{|\mathcal{A}|} x_i}{|\mathcal{A}|})^2 = (\frac{|\mathcal{O}|}{|\mathcal{A}|})^2 \tag{4-5}$$

在公式4-5中，第一个不等式是合并公式4-3和4-4的结果，第二个不等式是Cauchy-Schwarz不等式的应用，第三个不等式成立是因为$|\mathcal{A}| \geq 1$。最后，我们有$|\mathcal{O}| < \sqrt{n} \cdot |\mathcal{A}|$。 □

4.4 分布式启发算法OPDA

在这里，我们给出分布式的启发算法OPDA用来解决长度受限的不相交路径问题。给定网络中的节点s和t以及长度限制L，OPDA将会输出多条长度小于等于L的不相交路径。

OPDA的设计基于以下的观察：在无线传感器网络中，两个节点能否直接通信（有无线链接）主要取决于这两个节点间的物理距离。如果节点在监测区域内分布得比较均匀，s和t的临近区域通常会成为多条不相交$s \sim t$路径存在的瓶颈，也就是不相交$s \sim t$路径的数量主要被s和t的邻居个数所限制。

OPDA由算法 4-4给出。OPDA主要有三个步骤。在第一步中，我们在网络中分别建立以s和t为根的树T_s和T_t。两棵树的建立终止于这两棵树的交叉节点。在第二步中，交叉节点的一些信息被收集到s节点。这些信息揭示了T_s与T_t是如何交叉的。在第三步中，s 根据这些交叉信息计算一个本地的最优解。根据这个最优解，沿着T_s和T_t，多条$s \sim t$路径被建立起来。

Input: 两个节点s和t；正整数L
Output: 一些长度小于等于L的不相交$s \sim t$路径
1 调用**建立树**算法同时建立以s和t为根的树;
2 调用**收集交叉信息**算法;
3 调用**寻找不相交路径**算法;

算法 4-4 OPDA
Algo. 4-4 OPDA

4.4.1 建立树

在这一步中，我们建立一棵以s节点为根的树T_s和一棵以t节点为根的树T_t。对于T_s中的节点v：它在T_s中的父亲用$parent_s(v)$表示；它的s起源$origin_s(v)$是它在T_s中的祖先a，其中a是s在T_s中的孩子；$len_s(v)$表示T_s中从s到v的路径长度。对于T_t中的节点u：它在T_t中的父亲用$parent_t(u)$表示；它的t 起源$origin_t(u)$是它在T_t中的祖先a'，其中a'是t在T_t中的孩子；$len_t(u)$表示T_t中从t到u的路径长度。

假设在给定的传感器网络中，我们构建了如图 4-1所示的T_s和T_t。对于节点g：$len_s(g) = 4$，$len_t(g) = 4$，$origin_s(g) = a$，$origin_t(g) = d$。对于节点h：$len_s(h) = 4$，$len_t(h) = 4$，$origin_s(h) = b$，$origin_t(h) = d$。

如果一个节点v既是T_s中的节点又是T_t中的节点，我们称节点v是T_s与T_t的交

```
Input: 一个传感器网络；网络中的节点s
Output: 网络中一棵以s为根的树
/* 对于每个节点v的伪代码                                              */
1  if v是s节点 then
2      广播BUILD-S-TREE消息{s, ∅, 0};
3  else
4      while 接收到BUILD-S-TREE消息{u, origin_s(u), len_s(u)} do
5          if u = s then
6              parent_s(v) ← s;
7              origin_s(v) ← v;
8              len_s(v) ← 1;
9              广播BUILD-S-TREE消息{v, origin_s(v), len_s(v)};
10         else
11             将u记录为一个潜在父亲;
12             if 这是v第一次接收到BUILD-S-TREE消息 then
13                 根据len_s(u) + 1延迟一段时间;
14     if 延迟结束 then
           /* 假设v的潜在父亲来自与n个s起源a_1, ..., a_n                */
15         从a_1, ..., a_n中随机等概率地选择某个a_i;
16         在所有s起源是a_i的潜在父亲中选择len_s最小的节点w;
17         parent_s(v) ← w;
18         origin_s(v) ← a_i;
19         len_s(v) ← len_s(w) + 1;
20         if v不在T_t中并且len_s(v) < L then
21             广播BUILD-S-TREE消息{v, origin_s(v), len_s(v)};
```

算法 4-5 建立树(T_s)

Algo. 4-5 Building Tree (T_s)

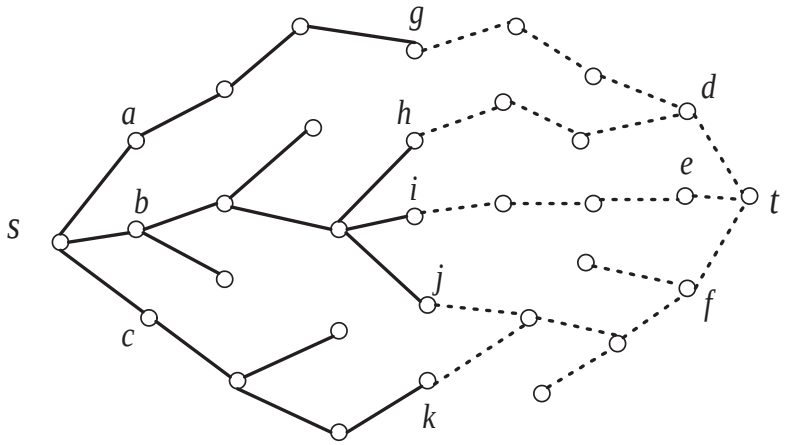


图 4-1 OPDA构建的T_s和T_t（实线表示T_s中的边，虚线表示T_t中的边）

Fig.4-1 T_s and T_t constructed by OPDA (solid lines denote the edges in T_s, dashed lines denote the edges in T_t)

叉节点。在图 4-1的例子中，节点g, h, i, j, k是T_s与T_t的交叉节点。

对于T_s中的两个节点u和v，我们用uT_sv来表示T_s中连接u和v的路径。如果两条路径P_1和P_2有一个公共的端点，我们用$P_1 + P_2$来表示它们的连接。

T_s和T_t的建立是同时进行的，分别从s和t开始向外扩展。以T_s为例，它的建立通过节点间交换BUILD-S-TREE消息来进行的。任意节点v发送的BUILD-S-TREE消息的格式为$\{v, origin_s(v), len_s(v)\}$，包含了其ID、$s$起源、$T_s$中从$s$到$v$的路径长度。整个过程以$s$节点广播$\{s, \emptyset, 0\}$来启动。每个接收到这个消息的节点作为$s$的孩子加入到$T_s$中并广播BUILD-S-TREE消息。每个新加入$T_s$的节点$v$广播BUILD-S-TREE消息以使$v$的那些没有在$T_s$中的邻居可以作为$v$的孩子加入到$T_s$中。

T_s的建立终止于：i) T_s和T_t的交叉节点；ii) T_s中满足$len_s(v) \geq L$的节点v。同样，T_t的建立终止于：i) T_s和T_t的交叉节点；ii) T_t中满足$len_t(u) \geq L$的节点u。

为了找到更多满足条件的不相交路径，我们希望树的建立尽量平衡。例如，假设s在T_s中有三个孩子a、b和c，我们希望以a、b和c作为s起源的节点的数量趋近于相等。基于这个原因，我们设计了一个建立树的延迟和随机选择机制。以T_s的构建为例，如果一个不在T_s中的节点第一次接收到BUILD-S-TREE消息（假设消息的发送者是u），v并不立即作为u的孩子加入到T_s中，而是延迟一段时间来等待接收更多的BUILD-S-TREE消息。延迟的时间根据当前从s到v的路径长度$len_s(u) + 1$来决定。在这段延迟的时间内，v记录下它所接收的每一个BUILD-S-TREE消息。并将这些消息的发送者作为其在T_s中潜在的父亲。假设在延迟的时间内，节点v接收到了节点u_1、u_2、u_3和u_4发送的BUILD-S-TREE消

息，它们的s起源分别是a、b、b和c。于是，v将u_1、u_2、u_3和u_4作为其在T_s中潜在的父亲。当延迟结束后，v随机地从$\{a,b,c\}$中等概率地选择一个s起源。假设v选择的是b。因为b是潜在父亲u_2和u_3的s起源，所以v从u_2和u_3 中选择len_s较小的节点作为自己在T_s中的父亲。

建立树T_s的伪代码由算法 4-5给出，建立树T_t的伪代码可以由此推出。整个过程由s广播BUILD-S-TREE消息$\{s,\emptyset,0\}$来初始化（算法 4-5中的第1-2行）。每个接收到这个消息的节点将自己设置为s在T_s中的孩子并广播BUILD-S-TREE消息（算法 4-5中的4-9行）。对于其它节点v，如果v第一次接收到BUILD-S-TREE消息，v延迟一段时间等待接收更多的BUILD-S-TREE消息，延迟的时间由当前从s到v的路径长度$len_s(u)+1$决定（算法 4-5中的第12-13行）。在延迟期间，v记录下所接收到的每个BUILD-S-TREE消息并将发送者作为自己潜在的父亲（算法 4-5中的第11行）。当延迟结束时，假设v的潜在父亲来自于n个s起源$a_1,\ldots,a_n$。接下来，v在$a_1,\ldots,a_n$中随机等概率地选择某个a_i，并在所有s起源是a_i的潜在父亲中选择len_s值最小的节点w（算法 4-5中第15-16行）。于是，v将w作为自己在T_s中的父亲，将a_i作为自己的s起源（算法 4-5中的第17-19行）。T_s的建立终止于：i) T_s和T_t的交叉节点；ii) T_s中满足$len_s(v) \geq L$的节点v（算法 4-5中的第20-21行）。

4.4.2 收集交叉信息

```
Input: 网络中分别以s和t为根的树T_s和T_t
Output: s收集到两棵树的交叉信息
/* 对于每个节点v的伪代码                                          */
1 if v是T_s和T_t的交叉节点 then
2     if len_s(v) + len_t(v) ≤ L then
3         将交叉信息{origin_s(v), origin_t(v)}发送给parent_s(v);
4 while 接收到某个交叉信息 do
5     记录下这个交叉信息和其发送者;
6     将这个交叉信息转发给parent_s(v);
```

算法 4-6 收集交叉信息

Algo. 4-6 Collecting Intersecting Information

在这一步中，我们收集交叉节点上的交叉信息。首先，每个T_s和T_t的交叉节点v检测是否有$len_s(v)+len_t(v) \leq L$。如果满足这个条件，我们可以

通过连接sT_sv和vT_tt来得到一条长度小于等于L的$s \sim t$路径P，于是v将交叉信息$\{origin_s(v), origin_s(t)\}$沿着$T_s$传送给$s$。这个交叉信息表明了$P$经过了$s$的邻居$origin_s(v)$和$t$的邻居$origin_t(v)$。每个转发交叉消息的节点记录下这个交叉消息和其发送者。收集交叉信息的伪代码由算法 4-6给出。

假设在图 4-1的例子中，用户设置$L = 8$。构建完T_s和T_t之后，在算法的第二步，交叉节点g将交叉信息$\{a, d\}$沿着T_s传送给节点s。同样地，交叉节点h、i、j、k分别将交叉信息$\{b, d\}$、$\{b, e\}$、$\{b, f\}$、$\{c, f\}$沿着T_s传送给s。

4.4.3 寻找不相交路径

这一步负责找到多条满足长度限制的不相交路径并在网络中建立这些路径。在描述这一步的程序之前，我们先给出下面一个命题，该命题可以由T_s和T_t的性质得出。

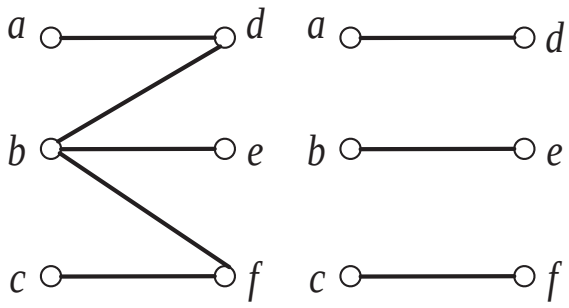
命题 4.1 对于T_s和T_t的交叉节点u和v来说，如果$origin_s(u) \neq origin_s(v)$并且$origin_t(u) \neq origin_t(v)$，那么路径$P_1 = sT_su + uT_tt$和$P_2 = sT_sv + vT_tt$是两条不相交路径。

```
Input: s收集到的交叉信息
Output: 一些长度小于等于L的不相交s ~ t路径
/* 对于节点s的伪代码                                              */
1 构建图G* = (V*, E*)，其中V* = {s和t的所有邻居}，E* = ∅;
2 for 每个接收到的交叉信息{origin_s(v), origin_t(v)} do
3     在G*中添加边(origin_s(v), origin_t(v));
4 寻找G*中的最大匹配;
5 for 最大匹配中的每条边(origin_s(v), origin_t(v)) do
6     向v传送一个BUILD-PATH消息;
```

算法 4-7 寻找不相交路径
Algo. 4-7 Finding Disjoint Paths

寻找不相交路径的伪代码由算法 4-7给出。首先，s构建一个不包含任何边的二部图G^*（算法 4-7中的第1行）。G^*的一个部集是s的邻居集合，另一个部集是t的邻居集合。如果某个节点v既是s的邻居又是t的邻居，则直接输出路径(s, v, t)，并且在以后的计算中不再考虑节点v。如果s接收到了交叉信息$\{origin_s(v), origin_t(v)\}$，它就在$G^*$中添加一条连接$origin_s(v)$和$origin_t(v)$的边

（算法 4-7 中的第2-3行）。根据命题4.1，G^*中的每个匹配都代表了网络中一个不相交$s \sim t$路径的集合。接着，s在本地应用现有的算法，如匈牙利算法[173]，搜索G^*中的最大匹配（算法 4-7 中的第4行）。对于最大匹配中的每条边$(origin_s(v), origin_t(v))$，$s$向$v$传送一个BUILD-PATH消息来构建这条$s \sim t$路径（算法 4-7中的第5-6行）。这个BUILD-PATH消息被沿着T_s传送到节点v。在此过程中，sT_sv被加入到路径中。接着，v将这个消息沿着T_t传送给节点t。在此过程中，vT_tt被加入到路径中。



a) G^*　　b) 最大匹配

a) G^*　　b) The maximum matching

图 4-2 G^*与最大匹配

Fig.4-2 G^* and the maximum matching

在图 4-1的例子中，接收完交叉信息后，s节点构建了如图 4-2 a)所示的G^*。在这个G^*中可以找到如图 4-2 b)所示的最大匹配(a,b)、(b,e)和(c,f)。因为这些匹配是分别由节点g,i,k传送到s的，所以s沿着T_s向节点g,i,k传送BUILD-PATH消息。在接收到BUILD-PATH消息之后，节点g,i,k沿着T_t向t传送BUILD-PATH消息。于是，如图 4-3所示的3条长度小于等于8的不相交$s \sim t$路径被建立起来。

让我们简单描述一下用于求二部图中最大匹配的匈牙利算法。假设在二部图G^*中已经发现了某个匹配M^*。我们定义图G^*中的路径P^*是一条相对于M^*的交错轨如果P^*连通了图G^*中两个未匹配的节点，并且属于M^*的边和不属于M^*的边在P上交替出现。通过取一个交错轨P^*和已知匹配M^*的对称差，我们可以得到G^*中一个更大的匹配。文献[173]证明了M^*是图G^*中的最大匹配当且仅当不存在相对于M^*的交错轨。于是，我们可以通过反复寻找交错轨的方式来寻找G^*中的最大匹配。

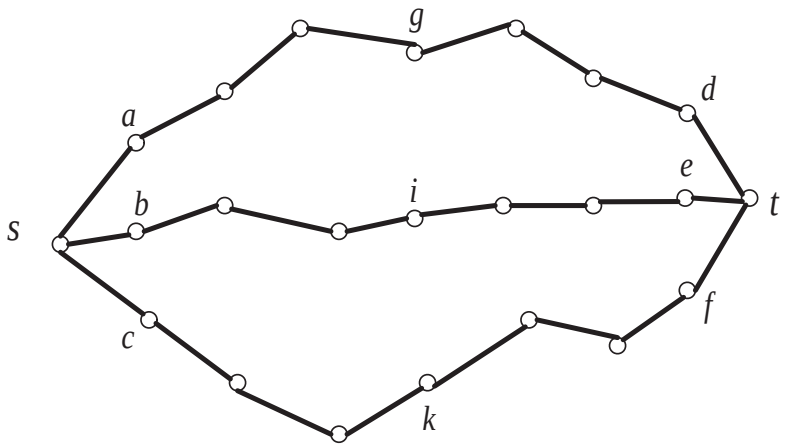


图 4-3 OPDA建立的3条不相交路径

Fig.4-3 The 3 disjoint paths built by OPDA

4.4.4 OPDA的优点

GDA是逐条地寻找不相交路径。每次寻找不相交路径的过程都需要在整个网络的节点间进行一次信息交换。与之相对的是，OPDA采用一次信息交换就可以找到多条不相交路径。在通信效率方面，OPDA无疑更具优势。另外，由于采用了一次处理的方式，OPDA只需要很短的执行时间，能够以很快的速度来满足用户的需求。

4.5 加权图的情况

假设在网络的拓扑图$G = (V, E)$中，每条边$e \in E$的长度不再是1而是一个任意的非负数$l(e)$。这种情况下，图G就成为了一个加权图。边的长度$l(e)$可以用来表示e所对应的链接的能量代价或者时间延迟等等。对于加权图G中的每条路径P来说，它的长度为$l(P) = \sum_{e \in P} l(e)$。

根据这个重新定义的路径长度，我们将GDA略作修改，让其在每次循环中寻找新定义下的最短$s \sim t$路径。这样，新得到的算法就可以解决加权图中长度受限的不相交路径问题。但是，这样一来，定理 4.1就不再成立，算法暂时还没有可分析的近似比。

通过将构建T_s的伪代码中的$len_s(u)+1$替换为$len_s(u)+l((u,v))$，将$len_s(w)+1$替换为$len_s(w) + l((w,v))$，我们可以得到在加权图中建立树的伪代码。这样的修改不会对OPDA的效率造成很大影响，也就是说OPDA可以很好地适应加权图中的情况。

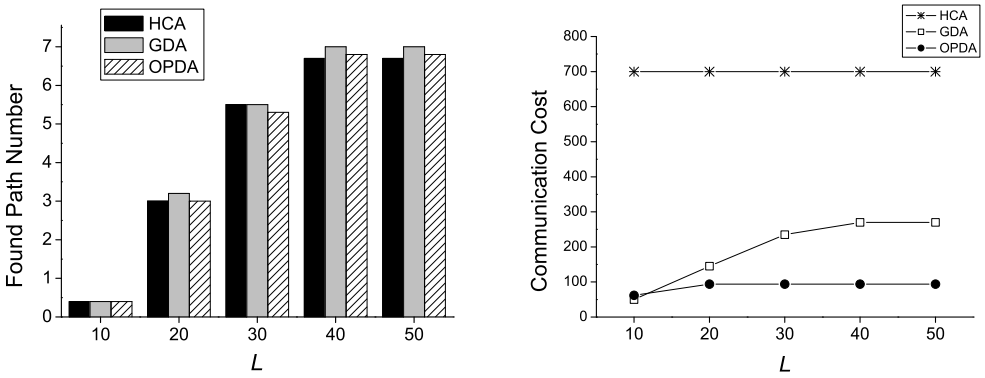
4.6 实验结果

我们用一个C++编写的模拟器来验证算法的效率。这个模拟器是一个基于应用层的程序，忽略了MAC层的细节。我们在一个1000m×1000m的监测区域内随机布置传感器节点。每个节点的通信半径被设置为50m。当两个节点间距离小于50m时，节点间存在无线链接。每个数据包包含了4个字节的包头来分别表示这个数据包的：目的节点的ID；发送者ID；数据包类型；数据包长度。每个接收数据和时钟到时事件被加上时间戳统一放到处理事件的堆中。堆中的事件按其时间戳的先后顺序一个一个进行处理。通过这种方式，我们来模拟网络中节点间的并行操作。每个实验数据都是10次实验（每次都设置不同的s和t）结果的平均值。

我们将GDA和OPDA与文献[148]中提出的集中式启发算法（用HCA表示）。算法的效率通过两个方面来衡量：算法找到的路径数量和算法的通信代价。算法的通信代价由实验中总的通信量（网络中信息交换的字节数）除以节点的数量得到，也就是平均每个节点接收和发送的字节数。根据1.2.1中的描述，通信代价可以很好表示节点的能量消耗。

在第一组实验中，我们在1000m×1000m的监测区域内随机布置了1000个传感器节点。网络中的每个链接的长度都被设置为1。根据不同的路径长度限制L，来比较三个算法。三种算法找到的路径数量在图 4-4 a)中给出。在找到的路径数量方面，三种算法的表现几乎相同，虽然GDA的表现略好一些。三种算法的通信代价在图 4-4 b)中给出。无论L的取值如何，HCA都需要收集整个网络的拓扑信息，所以它的通信代价是稳定的。GDA的通信代价随着找到的路径数量的增加而增加。找到更多的路径意味着GDA执行更多次循环，即网络中进行更多的信息交换。OPDA的通信代价也比较稳定。这是因为OPDA采取了一次信息交换的处理方式。当L = 10的时候，由于多次实验中网络中不存在长度小于等于10的路径，所以OPDA的通信代价比其它情况下低。从三种算法的对比可以看出，在通信代价方面，OPDA是三种算法中最优秀的，而HCA的表现最差。

在第二组实验中，我们在第一组实验的基础上为网络中的每个链接指定一个1到10之间的正整数来表示其长度。还是根据不同的L来比较三个算法。三种算法找到的路径数量在图 4-5 a)中给出。在加权的网络中，在找到的路径数量方面，OPDA的表现要比HCA和GDA的表现略好一些，而后两者的表现几乎相同。三种算法的通信代价在图 4-5 b)中给出。三种算法通信代价的比较再次验证了OPDA在这一方面的优越性。即使是GDA的通信代价，也比集中式的方法



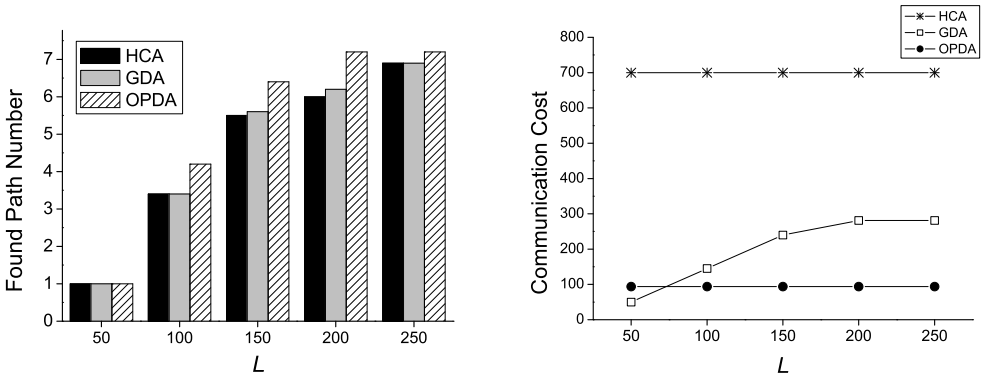
a) 找到的路径数量　　b) 通信代价

a) Found Path Number　　b) Communication Cost

图 4-4 第一组实验结果

Fig.4-4 Results of the 1st group experiments

小得多。



a) 找到的路径数量　　b) 通信代价

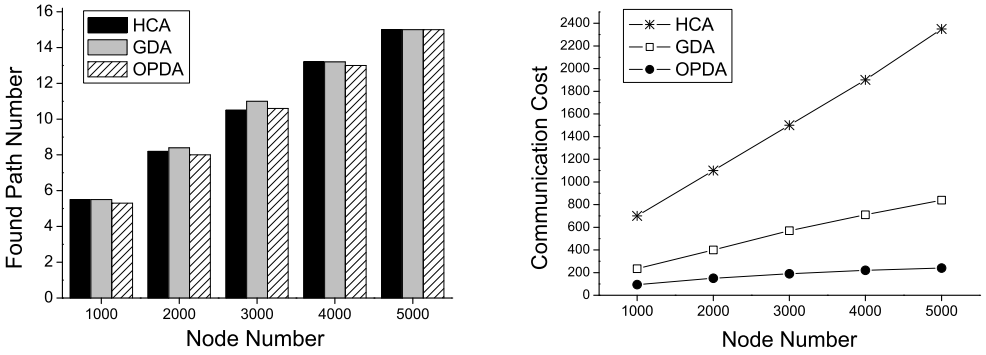
a) Found Path Number　　b) Communication Cost

图 4-5 第二组实验结果

Fig.4-5 Results of the 2nd group experiments

在第三组实验中，我们固定$L = 30$。在监测区域内随机布置$1000, 2000, 3000, 4000, 5000$个节点，网络中的每个链接的长度都被设置为1。根据不同的网络规模，来比较三个算法。三种算法找到的路径数量在图 4-6 a)中给出。从中我们还是可以看出，三种算法找到的路径数量几乎相同。三种算法的通信代价在图 4-6 b)中给出。HCA的通信代价随着节点数量的增加而明显增加。这是因为更多的节点数量意味这更大的网络规模和更复杂的拓扑结

构，而收集整个网络拓扑结构的代价就变得更高。GDA的通信代价也随着节点数量的增加而增加。这是因为在更密集的网络中，GDA找到的路径数量更多，从而节点间的信息交换次数更多。随着节点数量的增加，OPDA的通信代价只是略有增加。这是因为OPDA采取了一次信息交换的处理方式。虽然在更大规模的网络中参与信息交换的节点更多，但平均到每个节点上的通信代价只有少量的增长。比较三个算法，可以看出在通信代价方面，OPDA具有更好可扩展性。OPDA和GDA的通信代价比集中式的HCA要少得多。



a) 找到的路径数量　　b) 通信代价

a) Found Path Number　　b) Communication Cost

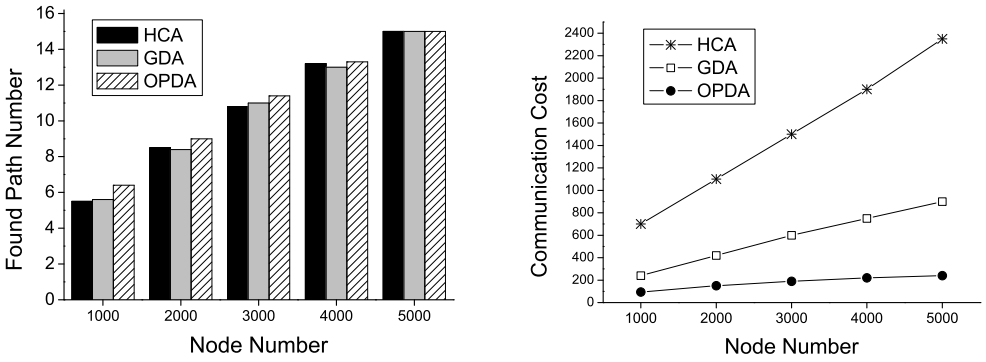
图 4-6 第三组实验结果

Fig.4-6 Results of the 3th group experiments

在第四组实验中，我们固定$L = 150$，并在第三组实验的基础上为网络中的每个链接指定一个1到10之间的正整数来表示其长度。三种算法找到的路径数量在图 4-7 a)中给出。虽然三种算法找到的路径数量基本相同，但在这样的加权网络中，OPDA的表现还是要略好一些。三种算法的通信代价在图 4-7 b)中给出。实验结果再一次验证了OPDA在通信效率方面的优越性。

4.7 本章小结

在本章中，我们研究了长度受限的不相交路径问题：给定网络中的节点s和t以及长度限制L，找到最多的长度小于等于L的不相交$s \sim t$路径。迄今为止，我们只知道这个问题不但是NP难的而且不存在多项式时间的近似模式。本章给出了这个问题的一个$\sqrt{n}$近似算法GDA以及GDA的分布式实现。其中，n是网络中节点的数量。另外，本章还给出了一个分布式的启发算法OPDA，实验结果表明OPDA无论是在找到的路径数量方面还是在通信效率方面都有着优异



a) 找到的路径数量　　b) 通信代价

a) Found Path Number　　b) Communication Cost

图 4-7 第四组实验结果

Fig.4-7 Results of the 4th group experiments

的表现。

第 5 章 无线传感器网络中的非干扰多路径路由

5.1 引言

传感器网络通常是一个大型的网络，网络中节点的数量可以达到成千上万。为了到达理想的监测效果，网络中往往会伴随着频繁的信息交换。于是在一个传感器网络中，通常需要在一定时间内满足许多的路由需求。每个路由需求是一对想要通过网络中的一条路径来建立通信的节点。由于传感器网络多跳自组织的特点，设计者需要制定路由策略为每个路由需求优化地建立路径。除了能量优化以外，设计者总是希望路由策略在以下几个方面有好的表现：吞吐量、鲁棒性、负载平衡和信息安全性。实现这些目标最直接的方法就是通过相互独立的路径来连接这些路由需求，使得每条路径都不受其它路径的“干扰”。于是，在传感器网络中用不相交路径来进行路由就成为了一种趋势[123–125, 127–134]。不相交路径是指不共享任何公共节点的路径。用不相交路径进行路由的想法起源于有线网络，通过增加路径间的独立性来提升路由的性能。但是在传感器网络这种无线网络中，不相交路径的特点在提升路由性能方面就受到了各种各样的限制，这是因为没有考虑到无线网络中的无线干扰问题。

在大多数传感器网络中，所有节点共享同一个信道，如果一个节点同时接收到两个或两个以上的无线信号，就会产生信号干扰，使得该节点不能正确接收任何信号。如图 1-5 a) 所示，假设有两条不相交的路径P_1和P_2，P_1上的节点a和P_2上的节点b在彼此的通信范围内（即它们互为邻居节点）。当节点a接收路径P_1上的数据时，节点b就不能转发路径P_2上的数据，反之亦然。不相交路径带来的路由性能的改善会因为无线干扰的发生而变得微不足道。如图 1-5 b) 所示，如果P_1和P_2上有很多互为邻居的节点，两条路径上的数据传输就会因为频繁的退避等待而大大降低传输速度，网络的吞吐量很难得到有效的提高。在这种情况下，处于无线干扰中的节点的能量也会因为竞争信道而被大量消耗，造成网络中某些节点能量消耗过快。另一方面，如果a是敌方布置的间谍节点，它不但可以获得路径P_1上的数据，还可以通过监听的方式获得节点b转发的路径P_2上的数据，传输安全性也会因为无线干扰而降低。

基于以上原因，研究者[118, 119]提出了非干扰路径的概念。用来描述无线网络中高度独立的路径。通常我们将无线网络表示为无向图$G = (V, E)$。图中的顶点表示网络中的节点，边表示网络中的无线链接。让$n = |V|$为图中节点的数量，

$m = |E|$为图中边的数量。图G中的路径$P_1, \ldots, P_k$是非干扰路径如果它们中的任意两条既没有公共节点也没有相邻节点。本章提到的“路径”是指无弦路径，即路径是一个诱导子图。本章研究的问题是广义非干扰路径问题：在无向图G中，给定节点对$\{s_1, t_1\}, \ldots, \{s_k, t_k\}$，用非干扰路径来连接这些节点对。据我们所知，现有的关于这个问题仅有的研究成果是：i) 这个问题是一个NP完全问题；ii) 当k是常数并且给定的图是可平面图时，这个问题有多项式时间的算法。迄今为止，对于普通的k 以及在普通的图中，还没有关于这个问题的近似算法被提出，这个问题的近似难度也不知道。

广义非干扰路径问题的优化版本就是最多非干扰路径问题（以下简称非干扰路径问题），即用非干扰路径来连接最多的给定的节点对。本章工作的贡献在于：

- 证明了即使当$k = 2$时，非干扰路径问题也是NP难的。
- 证明了对于任意的k和$\epsilon > 0$，以$m^{1/2-\epsilon}$来近似非干扰路径问题是NP难的。
- 给出了一个非干扰路径问题的$\sqrt{m}$近似算法GSPW。
- 给出了一个非干扰路径问题的一个在线算法OSBPW，满足$|O| \leq \sqrt{m} \cdot (|\mathcal{A}'| + 1)$。其中，$|O|$为最优解中路径数量，$|\mathcal{A}'|$是OSBPW所发现的路径数量。

本章的结构如下：5.2介绍了本章所研究问题的相关工作；5.3给出了一些必要的预备知识；5.4证明了非干扰路径问题的近似比下界；5.5给出了近似算法GSPW并证明了GSPW的近似比；5.6给出了在线算法OSBPW，并给出了OSBPW的分布式实现；5.7 提出了松弛非干扰路径的概念用来屏蔽掉网络本身造成的干扰；5.8通过模拟实验验证了所提出的算法的效率；5.9总结全章。

5.2 相关工作

无线干扰给传感器网络带来的危害在文献[162, 164]中有详尽的阐述。

文献[118, 171]证明了当给定的图是一个可平面图并且k是一个常数的时候，非干扰路径问题有多项式时间的解。但是，这两个限制并不适用于传感器网络。在传感器网络这种大型无线网络中，通常需要在一段时间内同时满足许多的路由需求，所以网络中的路由需求数量k更可能是一个变量而不是一个常数。另一方面，在大多数情况下，并不能用一个可平面图开表示给定网络。

非干扰路径问题的一个简化版本是2非干扰$s \sim t$路径问题：在网络中给定节点s和t，找到两条连接s和t的非干扰路径。这里，发生在s和t周围的相互干扰被忽略不计。文献[163]证明了2非干扰$s \sim t$路径问题是NP难的。文献[164, 168–

170]给出了这个问题启发式算法。文献[119] 给出了这个问题的随机算法，在节点分布均匀的情况下，有一定概率可以找到2条非干扰的$s \sim t$路径。当节点都装备了定向天线的时候，文献[167] 给出了这个问题的启发式算法。2非干扰$s \sim t$路径问题只是非干扰路径问题的一个特例。对比于2非干扰$s \sim t$路径问题，最多非干扰路径问题要复杂的多。

与非干扰路径问题密切相关的一个问题就是不相交路径问题。其问题定义为：在一个有向图或无向图$G = (V, E)$中，给定k个节点对$\{s_1, t_1\}, \ldots, \{s_k, t_k\}$，用节点不相交路径或者边不相交路径来连接最多的节点对。不相交路径问题被进行了广泛的研究[149, 150, 156–159]。无论给定的图是有向图还是无向图，无论要寻找的是边不相交路径还是节点不相交路径，不相交路径问题都是NP难的[151]。非干扰路径问题可以看作是无向图中节点不相交路径问题的一个扩展。这是因为通过将每条边“细分”为两条边（添加一个度为2的中间节点），每个无向图中节点不相交路径问题的实例都可以被转化为一个非干扰问题的实例。因此，非干扰路径问题一定是NP难的。

对于无向图中的边不相交路径问题，文献[157]给出了一个近似比为$O(\sqrt{m})$的近似算法。文献[150] 给出了一个近似比为$O(d')$的在线算法。其中，$d' = \max\{D_G, \sqrt{m}\}$，$D_G$为给定图$G$的直径。文献[158, 159]给出了这个问题的近似比为$O(\sqrt{n})$的近似算法，这是现今已知的最好的近似算法。我们不能将这些算法应用到非干扰路径问题中，因为非干扰路径问题与不相交路径问题不同，而且要更加复杂。

根据文献[156]，除非$NP \subseteq ZPTIME(n^{poly\log n})$，否则对于任意的$\epsilon > 0$，无向图中边不相交路径问题不能被$\log^{1/2-\epsilon} n$近似。根据文献[149]，除非$NP = P$，否则对于任意的$\epsilon > 0$，有向图中的边不相交路径问题不能被$m^{1/2-\epsilon}$近似。这些难度结论不能被直接应用到非干扰路径问题中，因为非干扰路径问题是无向图中节点不相交路径问题的一个扩展而不是有向图中边不相交路径问题或者无向图中边不相交路径问题的扩展。但是，我们在证明非干扰路径问题难度的时候，用到了文献[149] 的证明思想。

5.3 预备知识

我们假设给定的传感器网络是静态的；网络中所有的节点共用一个信道；所有的链接都是双向的；网络是连通的。我们可以将给定的网络用其拓扑图$G = (V, E)$表示。其中，顶点（节点）集合为$V = \{v \mid v$ 是网络中的节点$\}$，边集合为$E = \{(u, v) \mid u$ 和v 之间存在无线链接$\}$。定义$n = |V|$和$m = |E|$。

图G中的一条路径P是G的一个子图，可以表示为一系列不同的节点，即$P = (v_1, v_2, \ldots, v_n)$。其中，$v_i \in V$，$(v_i, v_{i+1}) \in E$。这里，$v_1$和$v_n$是$P$的两端节点，$P$上的其它节点被称作中间节点。一条两端节点是$v_1$和$v_n$的路径被称作一条$v_1 \sim v_n$路径。对于路径$P$，$|P|$表示$P$上节点的数量。一条边$(u, v) \in E$被称作$P$的弦如果$u$和$v$是$P$上的两个不相邻的节点。路径$P$被称作是无弦的如果图$G$中不存在$P$的弦，也就是$P$是$G$的一个诱导子图。本章中提到"路径"都是指无弦路径。

定义 5.1 在图$G = (V, E)$中，路径$P_1, \ldots, P_k$是非干扰路径如果对于任意的$0 \le i < j \le k$，P_i和P_j没有相邻节点，即$\{(u, v) \in E \mid u \in P_i, v \in P_j\} = \emptyset$。

从路径P_i和P_j没有相邻节点可以推倒出它们没有公共节点，即$\{(u, v) \in E \mid u \in P_i, v \in P_j\} = \emptyset \Rightarrow P_i \cap P_j = \emptyset$。如果两条路径有公共节点或者相邻节点，我们说它们相互干扰。

定义 5.2 对于图$G = (V, E)$中的一条路径P，$G_I[P] = (V_I[P], E_I[P])$是$P$的干扰图。其中，$V_I[P] = \{v \mid v \in P$ 或者$\exists (u, v) \in E$ 满足$u \in P\}$，$E_I[P] = \{(u, v) \mid u \in P$ 或者$v \in P\}$。$V_I[P]$被称作P的干扰节点集，$E_I[P]$被称作P的干扰边集。

引理 5.1 对于两条路径P_1和P_2来说，下列表述是等价的。

1. P_1和P_2是非干扰路径。
2. $E_I[P] \cap E_I[P] = \emptyset$。
3. P_1不包含$V_I[P_2]$中的节点。
4. P_2不包含$V_I[P_1]$中的节点。

根据非干扰路径的定义，我们可以得到引理 5.1。

（最多）非干扰路径问题可以形式化的表示为：

输入：图$G = (V, E)$中的k对节点$\{s_1, t_1\}, \ldots, \{s_k, t_k\}$。

输出：η 条非干扰路径$P_{i_1}, P_{i_2} \ldots, P_{i_\eta}$（$1 \le i_1 < i_2 < \cdots < i_\eta \le k$），其中对于$x = 1, \ldots, \eta$，$P_{i_x}$是一条$s_{i_x} \sim t_{i_x}$路径。

优化目标：最大化η。

5.4 问题的难度

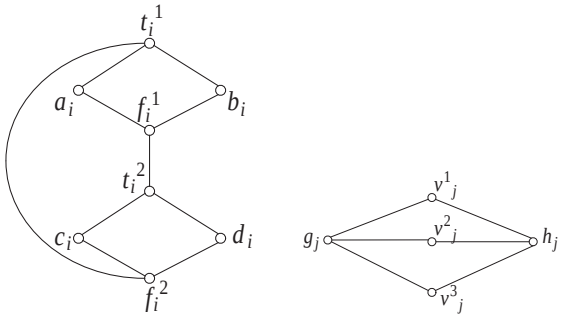
5.4.1 $k = 2$的时候是NP难的

在这一部分，我们将证明即使当$k = 2$时，非干扰路径问题也是NP难的。当$k = 2$时，我们称这个问题的判定版本为2非干扰路径问题，其定义为：给

定图$G = (V, E)$中4个不同的节点s_1, t_1, s_2, t_2，判断图G中是否存在非干扰路径的$s_1 \sim t_1$路径P_1和$s_2 \sim t_2$路径P_2。

现在，让我们来证明2非干扰路径问题是NP完全问题。很明显，给定$s_1 \sim t_1$路径P_1和$s_2 \sim t_2$路径P_2，我们可以在多项式时间内验证P_1和P_2是否是非干扰路径，所以2非干扰路径问题是一个NP问题。下面我们将一个3满足问题（NP完全问题）在多项式时间内规约到2非干扰路径问题。3满足问题的定义为：我们有N个逻辑变量$x_1, \ldots, x_N$和M个子句$C_1, \ldots, C_M$。每个子句C_j（$1 \le j \le M$）都有相同的形式$C_j = z_j^1 \vee z_j^2 \vee z_j^3$。这里，对于$1 \le l \le 3$，$z_j^l = x_i$或者$z_j^l = \overline{x_i}$，而$x_i$是某个逻辑变量。这样的每个$z_j^l$被称作字符。我们想要知道是否存在对变量$x_1, \ldots, x_N$的一个赋值可以满足表达式$C_1 \wedge \cdots \wedge C_M$。

给定一个3满足问题的实例，我们通过三个步骤来构造一个图G。



a) α_i　　b) β_j

a) α_i　　b) β_j

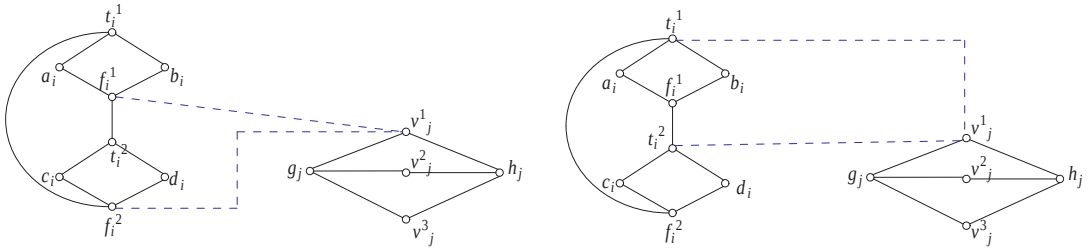
图 5-1 变量和子句的子图

Fig.5-1 Subgraphs of variables and clauses

在第一步中，如图 5-1 a) 所示，对于每个变量x_i，我们构造图G的一个子图α_i。在构造完成后的图G中，$s_1 \sim t_1$路径P_1和$s_2 \sim t_2$路径P_2会从左至右地经过这个子图。P_1经过(a_i, t_i^1, b_i)以及P_2经过(c_i, t_i^2, d_i)代表x_i被赋值为真。P_1经过(a_i, f_i^1, b_i) 以及P_2经过(c_i, f_i^2, d_i)代表x_i被赋值为假。如图 5-1 b) 所示，对于每个子句$C_j = z_j^1 \vee z_j^2 \vee z_j^3$，我们构造图$G$的一个子图$\beta_j$。其中，节点$v_j^1, v_j^2, v_j^3$分别对应字符$z_j^1, z_j^2, z_j^3$。

在第二步中，对于每个子句$C_j = z_j^1 \vee z_j^2 \vee z_j^3$，如果$z_j^l = x_i$，如图 5-2 a) 所示，我们在图$G$中添加连接$\alpha_i$和$\beta_j$的边$(v_j^l, f_i^1)$和$(v_j^l, f_i^2)$。如果$z_j^l = \overline{x_i}$，如图 5-2 b) 所示，我们在$G$中添加边$(v_j^l, t_i^1)$和$(v_j^l, t_i^2)$。

在第三步中，我们将所有的α_i和β_j与4个额外的节点s_1, t_1, s_2, t_2连接到一起。连接的方式是在图G中添加以下这些边：(s_1, a_1)，(s_2, c_1)；对于$i = 1, \ldots, N -$



a) 当$z_j^1 = x_i$时　　b) 当$z_j^1 = \overline{x_i}$时

a) Case $z_j^1 = x_i$　　b) Case $z_j^1 = \overline{x_i}$

图 5-2 连接变量和子句的边（用虚线表示）

Fig.5-2 Edges connecting variables and clauses (denoted by dashed lines)

1，添加(b_i, a_{i+1})和(d_i, c_{i+1})；(b_N, t_1)和(d_N, g_1)；对于$j = 1, \ldots, M-1$，添加(h_i, g_{i+1})；(h_M, t_2)。构造好的图G如图 5-3所示。

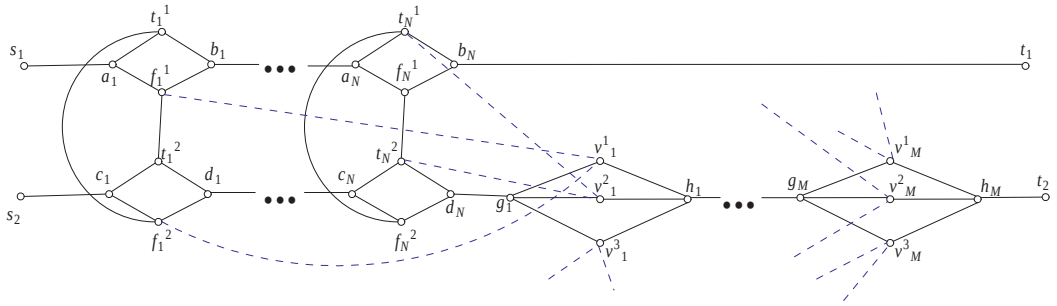


图 5-3 构造的图G（$C_1 = z_1^1 \vee z_1^2 \vee z_1^3$，$z_1^1 = x_1$，$z_1^2 = \overline{x_N}$）

Fig.5-3 The constructed G ($C_1 = z_1^1 \vee z_1^2 \vee z_1^3$，$z_1^1 = x_1$，$z_1^2 = \overline{x_N}$)

显然，图G可以在多项式时间内构造完成。

引理 5.2 $s_1 \sim t_1$路径P_1和$s_2 \sim t_2$路径P_2是构造的图G中的两条非干扰路径。那么对于$i = 1, \ldots, N$，一定有下列情况之一存在：

1. P_1包含(a_i, t_i^1, b_i)，P_2包含(c_i, t_i^2, d_i)。
2. P_1包含(a_i, f_i^1, b_i)，P_2包含(c_i, f_i^2, d_i)。

证明： 让我们沿着从s到t的方向，枚举P_1和P_2上的各条边。显然，P_1上的第一条边一定是(s_1, a_1)，P_2上的第一条边一定是(s_2, c_1)。下面，我们将通过对i进行归纳来证明这个引理。

当$i = 1$时：P_1上的第二条边一定是(a_1, t_1^1)和(a_1, f_1^1)中的一条，P_2上的第二条边一定是(c_1, t_1^2)和(c_1, f_1^2)中的一条。不失一般性地，我们假设P_1上第二条边是(a_1, f_1^1)。为了保证非干扰的性质，P_2上的第二条边一定是(c_1, f_1^2)。接下来，P_1上的第三条边可能是以下几种情况：i) (f_1^1, b_1) ii) (f_1^1, t_1^2) iii) 某条连接变量

和子句的边(f_1^1, v_j^l)（$1 \le l \le 3, 1 \le j \le M$）。如果$P_1$上的第三条边是$(f_1^1, t_1^2)$，因为$c_1 \in P_2$，$P_1$和$P_2$将会互相干扰。根据我们构造图$G$的方法，如果$(f_1^1, v_j^l) \in E$，那么$(f_1^2, v_j^l) \in E$。所以，如果$P_1$的第三条边是某个$(f_1^1, v_j^l)$，因为$f_1^2 \in P_2$，$P_1$和$P_2$将会互相干扰。因此，$P_1$上的第三条边一定是$(f_1^1, b_1)$。为了与$P_1$不互相干扰，$P_2$上的第三条边一定是$(f_1^2, d_1)$。

假设引理2对于$i-1$成立。P_1上的下一条边一定是(b_{i-1}, a_i)，P_2上的下一条边一定是(d_{i-1}, c_i)，接下来的证明过程与$i = 1$的情况相同。 □

引理 5.3 $C_1 \wedge \cdots \wedge C_M$可以被满足当且仅当在图$G$中存在非干扰的$s_1 \sim t_1$路径$P_1$和$s_2 \sim t_2$路径$P_2$。

证明： “⇒”。假设有一个对变量$x_1, \ldots, x_N$的赋值满足$C_1 \wedge \cdots \wedge C_M$。根据这个赋值，沿着从$s$到$t$的方向，我们来构造$s_1 \sim t_1$路径$P_1$和$s_2 \sim t_2$路径$P_2$。添加$(s_1, a_1)$到$P_1$，添加$(s_2, c_1)$到$P_2$。对于每个变量$x_i$：如果$x_i$的赋值为真，添加路径$(a_i, t_i^1, b_i)$到$P_1$，添加路径$(c_i, t_i^2, d_i)$到$P_2$；如果$x_i$的赋值为假，添加路径$(a_i, f_i^1, b_i)$到$P_1$，添加路径$(c_i, f_i^2, d_i)$到$P_2$。到目前为止，$P_1$和$P_2$是非干扰的。接下来，添加$(b_N, t_1)$到$P_1$，添加$(d_N, g_1)$到$P_2$。对于每个子句$C_j = z_j^1 \vee z_j^2 \vee z_j^3$：选择一个被赋值为真的字符$z_j^l$（$1 \le l \le 3$），添加路径$(g_j, v_j^l, h_j)$到$P_2$。最后，添加$(h_M, t_2)$到$P_2$。因为$C_1 \wedge \cdots \wedge C_M$被满足，所以在每个子句$C_j$中，其字符$z_j^1, z_j^2, z_j^3$至少有一个被赋值为真。不失一般性地，假设$z_j^1 = x_i$被赋值为真，$P_2$包含路径$(g_j, v_j^1, h_j)$。于是，$P_1$包含路径$(a_i, t_i^1, b_i)$，$P_2$包含路径$(c_i, t_i^2, d_i)$。图$G$中不存在边$(v_j^1, t_i^1)$或$(v_j^1, t_i^2)$。因此$s_1 \sim t_1$路径$P_1$和$s_2 \sim t_2$路径$P_2$是非干扰路径。

“⇐”。假设在图G中存在非干扰的$s_1 \sim t_1$路径P_1和$s_2 \sim t_2$路径P_2。根据引理2，对于$i = 1, \ldots, N$，下列两种情况之一成立：i) P_1包含(a_i, t_i^1, b_i)，P_2包含(c_i, t_i^2, d_i)；ii) P_1包含(a_i, f_i^1, b_i)，P_2包含(c_i, f_i^2, d_i)。如果是第一种情况，我们将变量x_i赋值为真。如果是第二种情况，我们将变量x_i赋值为假。沿着从s到t的方向，P_1的最后一条边一定是(b_N, t_1)。在离开d_N之后，对于每个子句$C_j = z_j^1 \vee z_j^2 \vee z_j^3$，$P_2$一定包含一条$(g_j, v_j^l, h_j)$（$1 \le l \le 3$）路径。不失一般性地，我们假设$P_2$包含$(g_j, v_j^1, h_j)$并且$z_j^1 = x_i$。于是，图$G$中存在边$(v_j^1, f_i^1)$和$(v_j^1, f_i^2)$。因为$P_1$和$P_2$是非干扰路径，所以一定有$P_1$包含$(a_i, t_i^1, b_i)$，$P_2$包含$(c_i, t_i^2, d_i)$。这就意味着$x_i$的赋值为真，子句$C_j$被满足。 □

定理 5.1 2非干扰路径问题是NP完全问题。

由于图G可以在多项式时间内构造完成，我们可以通过引理 5.3 得到定理 5.1。

5.4.2 近似比下界分析

在这一部分，我们将证明当k取任意正整数的时候，对于任意的$\epsilon > 0$，以$m^{1/2-\epsilon}$来近似非干扰路径问题是NP难的。这里，m是给定图G中边的数量。

一个$k\times k$网格是一个$k\times k$的棋盘图。它的节点集合是$\{v_{ij} \mid i,j \in \{0,1,\ldots,k-1\}\}$，边集合为$\{(v_{ij},v_{xy}) \mid |x-i|+|y-j|=1\}$。

我们证明的基本思想是：给定一个2非干扰路径问题实例和$\epsilon > 0$，我们构造一个包含节点对$\{s_1,t_1\},\ldots,\{s_k,t_k\}$的图$G=(V,E)$，满足$k \geq |E|^{1/2-\epsilon}$并且：i) 2非干扰路径问题答案为“是”当且仅当所有的这些节点对都可以被G中的非干扰路径连接；ii) 2非干扰路径问题的答案为“否”当且仅当这些节点对中最多只有一对可以被非干扰路径连接。

假设给定的2非干扰路径问题的实例是：在图$H=(V_H,E_H)$中给定4个不同的节点a,b,c,d，问H中是否存在非干扰的$a\sim b$路径R_1和$c\sim d$路径R_2。对于任意的$\epsilon > 0$，我们分两步来构造图$G=(V,E)$。首先设定$k=\lceil |E_H|^{1/\epsilon} \rceil$。

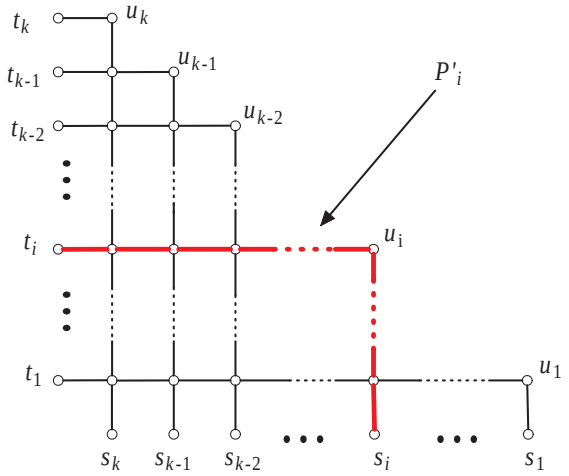


图 5-4 构造的图G'（粗线表示P'_i）

Fig.5-4 The constructed G' (bold lines denote P'_i)

在第一步中，我们构造一个$(k+1)\times(k+1)$网格的子图$G'=(V',E')$。它的节点集合为$V'=\{v_{ij} \mid i,j \in \{0,1,\ldots,k\}$ 并且$1 \leq i+j \leq k+1\}$，边集合为$E'=\{(v_{ij},v_{xy}) \mid |x-i|+|y-j|=1$ 并且$x+i \geq 1, y+j \geq 1\}$。对于$i=1,\ldots,k$，我们设定$t_i=v_{0,i}$，$s_i=v_{k+1-i,0}$以及$u_i=v_{k+1-i,i}$。构造好的图G'如图 5-4 所示。

在第二步中，我们通过对G'进行以下改动来构造图G。i) 将图G'中每个度为4的节点v_{ij}替换为图H的一个副本。该副本用H_{ij}来表示。其中，a,b,c,d 对应的节点分别用$a_{ij},b_{ij},c_{ij},d_{ij}$来表示。ii) 将图$G'$中的每条$(v_{ij},v_{i+1,j})$形式的边替换为$(d_{ij},c_{i+1,j})$。将图$G'$中每条$(v_{ij},v_{i,j+1})$形式的边替换为$(a_{ij},b_{i,j+1})$。对于每个没有

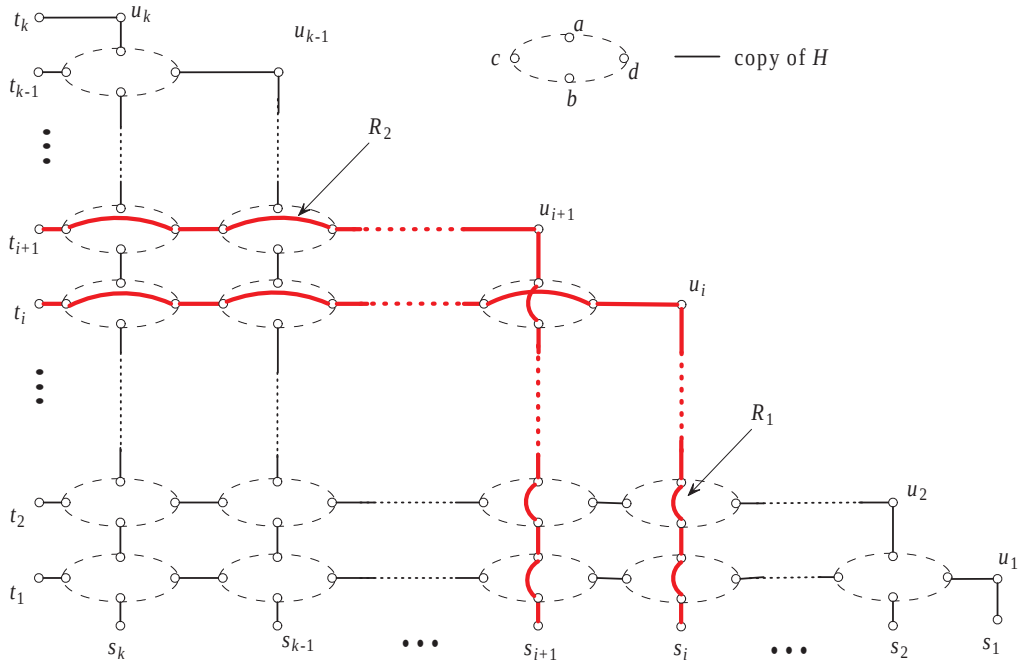


图 5-5 构造的图G（粗线表示P_i和P_{i+1}）

Fig. 5-5 The constructed G (bold lines denote P_i and P_{i+1})

被H副本替换的节点v_{ij}，即v_{ij}是s, t, u，我们设定$v_{ij} = a_{ij} = b_{ij} = c_{ij} = d_{ij}$。构造好的图$G$如图 5-5 所示。

引理 5.4 所有的$\{s_1, t_1\}, \ldots, \{s_k, t_k\}$都可以被图$G$中的非干扰路径连接当且仅当图$H$中存在非干扰的$a \sim b$路径和$c \sim d$路径。

证明： “⇐”。假设图H中存在非干扰的$a \sim b$路径R_1和$c \sim d$路径R_2。在图H的每个副本H_{xy}中，让R^1_{xy}和R^2_{xy}分别表示R_1和R_2所对应的路径。在图G'中，对于每个节点对$\{s_i, t_i\}$，我们定义如图 5-4 中粗线所示的路径：

$$P'_i = (t_i,\ v_{1,i},\ v_{2,i},\ \ldots,\ v_{k-i,i},\ u_i, v_{k-i+1,i-1},\ \ldots,\ v_{k-i+1,1},\ s_i)$$

通过将P'_i中的每个节点$v_{x,i}$（$1 \le x \le k - i$）替换为路径$R^2_{x,i}$，将P'_i中的每个节点$v_{k-i+1,y}$（$i - 1 \ge y \ge 1$）替换为路径$R^1_{k-i+1,y}$，我们可以得到图G中的$s_i \sim t_i$路径P_i。路径P_i和P_{i+1}如图 5-5 中粗线所示。因为R_1和R_2是非干扰路径，对于任意的$1 \le i < j \le k$，P_i和P_j一定是非干扰路径。

“⇒”。假设图G中存在k条非干扰路径$Q_1, \ldots, Q_k$，其中Q_i（$1 \le i \le k$）是一条$s_i \sim t_i$路径。如果将G以如图 5-5所示的方式“嵌入”到平面上，对于任意的$1 \le i < j \le k$，Q_i形成了一个“障碍”使得s_j和t_j分别位于其两侧。因此，一定存在某个H的副本H_{xy}满足：Q_i和Q_j中的一个包含一条$a_{xy} \sim b_{xy}$路径，它们中的另一个包含一条$c_{xy} \sim d_{xy}$路径。如果不是这样，Q_j不可能穿过Q_i这个“障碍”

连接s_j和t_j。因为Q_i和Q_j是非干扰路径，所以H_{xy}中存在非干扰的$a_{xy} \sim b_{xy}$路径和$c_{xy} \sim d_{xy}$路径，也就是H中存在非干扰的$a \sim b$路径和$c \sim d$路径。 □

推论 5.1 $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$中至多只有一对节点可以被图$G$中的非干扰路径连接当且仅当$H$中不存在非干扰的$a \sim b$路径和$c \sim d$路径。

推论 5.1 可以从定理 5.4 的证明中得到。

图G'中边的数量为$|E'| = k(k+1)$。度为4的节点的数量为$k(k-1)/2$。因此对于图G，我们有

$$m = |E| = k(k+1) + \frac{k(k-1)}{2} \cdot |E_H| \leq k^2 |E_H|$$

其中的不等式成立是因为$|E_H| \geq 2$。由于$k = \lceil |E_H|^{1/\epsilon} \rceil \geq |E_H|^{1/\epsilon}$，我们有$m \leq k^{2+\epsilon}$和$k \geq m^{\frac{1}{2+\epsilon}}$。$\epsilon$可以是任意小的正数，所以我们可以得到下面的定理 5.2。

定理 5.2 存在一些列的图$G = (V, E)$以及图G中的k个节点对$\{s_1, t_1\}, \ldots, \{s_k, t_k\}$满足对于任意的$\epsilon > 0$，区分以下两种情况是NP 难的：i) 所有的这些节点对都可以被G中的非干扰路径连接；ii) 这些节点对中至多只有$\frac{1}{m^{1/2-\epsilon}}$个可以被$G$中的非干扰路径连接。

根据定理 5.2，我们知道除非P=NP，否则对于非干扰路径问题，$\sqrt{m}$是我们可能得到的最好的近似比。

5.5 近似算法GSPW

基于贪心的策略，我们给出非干扰路径问题的一个近似比为$\sqrt{m}$的算法GSPW。

对于图G中的每个节点v，$d_G(v)$表示节点v的度。我们定义一个函数$w : V \to \mathbb{N}$为$w(v) = d_G(v) - 1$。对于G中的一条路径P，他的w长度被定义为$l_w(P) = \sum_{v \in P} w(v)$。$w$最短路径是指相对于$w$长度的最短路径。

GSPW如算法 5-1 所示。算法循环的搜索非干扰路径。在每次循环中：对于每个尚未被连接的$\{s_i, t_i\}$，寻找一条w最短的$s_i \sim t_i$路径P_i（算法 5-1 中第3-4行）。选择$\{s_\lambda, t_\lambda\}$满足P_λ在所有的P_i（$\{s_i, t_i\}$尚未被连接）中具有最小的w长度（算法 5-1 中第5行）。用P_λ连接$\{s_\lambda, t_\lambda\}$（算法 5-1 中第8-9行）。将P_λ的干扰节点集$V_I[P_\lambda]$从G中删除（算法 5-1 中第10行）。算法执行到所有的节点对都被连接或者图G中不存在可以被连接的节点对。

很明显，GSPW产生的这些路径都是无弦路径并且这些路径是非干扰的。

Input: 图G；G中的k个节点对$\{s_1,t_1\},\ldots,\{s_k,t_k\}$
Output: 一些连接这些节点对的非干扰路径

$\mathcal{T}=\{\{s_1,t_1\},\ldots,\{s_k,t_k\}\}$;
while $\mathcal{T}\neq\emptyset$ **do**
  for $\mathcal{T}$中的每个$\{s_i,t_i\}$ **do**
    找到一条w最短的$s_i\sim t_i$路径P_i;
  选择P_λ满足$l_w(P_\lambda)=\min\{l_w(P_i)\mid\{s_i,t_i\}\in\mathcal{T}\}$;
  if 没有这样的P_λ存在 **then**
    算法停止;
  输出P_λ ;
  $\mathcal{T}=\mathcal{T}\setminus\{\{s_\lambda,t_\lambda\}\}$;
  $G=G-V_I[P_\lambda]$;

算法 5-1 GSPW
Algo. 5-1 GSPW

引理 5.5 在图G中，让$\{P_1,\ldots,P_\eta\}$表示任意一个非干扰路径集合。于是，我们有$\sum_{i=1}^{\eta}[l_w(P_i)+1]\leq m$。

证明： 因为$\{P_1,\ldots,P_\eta\}$是非干扰路径，根据引理1，它们的干扰边集没有任何公共边。另一方面，$E_I[P_1],E_I[P_2],\ldots,E_I[P_\eta]$ 都是E的子集，所以有

$$\sum_{i=1}^{\eta}|E_I[P_i]|\leq|E|=m \tag{5-1}$$

对于每个P_i，

$$|E_I[P_i]|=\sum_{v\in P_i}d_G(v)-|P_i|+1=l_w(P_i)+1 \tag{5-2}$$

通过将每个$v\in P_i$的$d_G(v)$相加，两个端点都在P_i上的边被计算了两次，只有一个端点在P_i上的边被计算了一次。因为P_i是无弦的，有两个端点在P_i上的边一共有$|P_i|-1$条，所以公式 5-2 成立。 □

引理 5.6 在图G中，让$\mathcal{O}$表示非干扰路径问题的一个最优解（非干扰路径集合），于是任意一条路径P至多干扰$\mathcal{O}$中的$l_w(P)+1$ 条路径。

证明： 根据公式 5-2，$|E_I[P]|=l_w(P)+1$。对于$\mathcal{O}$中被P干扰的每条路径Q，$E_I[P]$和$E_I[Q]$至少有一条公共边。对于$\mathcal{O}$中的任意两条路径Q_1和Q_2，$E_I[Q_1]\cap E_I[Q_2]=\emptyset$。因此，$P$至多干扰$\mathcal{O}$中的$l_w(P)+1$ 条路径。 □

定理 5.3 GSPW是一个$\sqrt{m}$近似算法。

证明： 让O表示非干扰路径问题的一个最优解。让$\mathcal{A}$表示GSPW产生的非干扰路径集合。O中的每条路径Q一定被$\mathcal{A}$中的某条路径所干扰，否则通过将Q添加到$\mathcal{A}$中，GSPW可以进一步扩展$\mathcal{A}$。注意如果$P \in \mathcal{A}$也是O中的一条路径，P干扰其自身。

在$\mathcal{A}$中，让P_i表示GSPW在第i循环中产生的路径。$O_i \subseteq O$包含O中那些与P_i互相干扰但是与$P_1, \ldots, P_{i-1}$没有干扰的路径。设定$x_i = |O_i|$，于是$|O| = \sum_{i=1}^{|\mathcal{A}|} x_i$。根据引理 5.6，

$$x_i \leq l_w(P_i) + 1 \tag{5-3}$$

对于O_i中的每条路径Q来说，$l_w(Q) + 1 \geq x_i$，否则我们有$l_w(Q) < l_w(P_i)$，这样Q就应该代替P_i被GSPW选中。因此，对于每个O_i，

$$\sum_{Q \in O_i} [l_w(Q) + 1] \geq x_i^2 \tag{5-4}$$

根据引理 5.5，

$$\sum_{Q \in O} [l_w(Q) + 1] = \sum_{i=1}^{|\mathcal{A}|} \sum_{Q \in O_i} [l_w(Q) + 1] \leq m \tag{5-5}$$

因此，

$$m \geq \sum_{i=1}^{|\mathcal{A}|} x_i^2 \geq \frac{(\sum_{i=1}^{|\mathcal{A}|} x_i)^2}{|\mathcal{A}|} \geq (\frac{\sum_{i=1}^{|\mathcal{A}|} x_i}{|\mathcal{A}|})^2 = (\frac{|O|}{|\mathcal{A}|})^2 \tag{5-6}$$

在公式 5-6 中，第一个不等式是合并公式 5-4 和公式 5-5 的结果，第二个不等式是Cauchy-Schwarz 不等式的应用，第三个不等式成立是因为$|\mathcal{A}| \geq 1$。最后，我们有$|O| \leq \sqrt{m} \cdot |\mathcal{A}|$。 □

5.6 在线算法OSBPW

很多时候，在传感器网络的应用中，路由需求是一个一个产生的，并且我们需要在第一时间内处理所产生的路由需求。在这种情况下，我们就需要非干扰路径问题的在线算法。这一部分，我们将给出非干扰路径问题的一个在线算法OSBPW，该算法拥有很好的下界。

OSBPW由算法 5-2 给出。算法一个一个地处理产生的节点对。对于每个节点对$\{s_i, t_i\}$：找到一条w最短的$s_i \sim t_i$路径P_i（算法 5-2 中第2行）。如果P_i的w长度小于等于$\sqrt{m} - 1$，就用P_i来连接$\{s_i, t_i\}$并且将P_i的干扰节点集从图G中删除（算法 5-2 中第3-5行）。

Input: 图G；G中的k个节点对$\{s_1, t_1\}, \ldots, \{s_k, t_k\}$
Output: 一些连接这些节点对的非干扰路径

for $i = 1, \ldots, k$ **do**
  找到一条w最短的$s_i \sim t_i$路径P_i;
  if $l_w(P_i) \leq \sqrt{m} - 1$ **then**
    输出P_i ;
    $G = G - V_I[P_i]$;

算法 5-2 OSBPW
Algo. 5-2 OSBPW

显然，OSBPW产生的路径都是无弦路径并且这些路径是非干扰的。

定理 5.4 让$\mathcal{O}$表示非干扰路径问题的一个最优解，$\mathcal{A}'$表示OSBPW产生的非干扰路径集合。于是，我们有$|\mathcal{A}'| > \frac{|\mathcal{O}|}{\sqrt{m}} - 1$。

证明： 定义$\mathcal{O}_I \subseteq \mathcal{O}$为$\mathcal{O}_I = \{Q \mid Q \in \mathcal{O}$ 并且Q 与$\mathcal{A}'$ 中的某条路径相互干扰$\}$。定义$\mathcal{O}_U = \mathcal{O} \setminus \mathcal{O}_I$。注意对于$\mathcal{A}'$中的一条$s_i \sim t_i$路径$P$和$\mathcal{O}$中的一条$s_i \sim t_i$路径$Q$，$P$一定和$Q$互相干扰因为它们有公共节点$s_i$和$t_i$。因此，$\mathcal{O}_U$中的路径所对应的节点对都没有被$\mathcal{A}'$中的路径连接。

对于$\mathcal{O}_U$中的每条路径Q来说，我们有$l_w(Q) > \sqrt{m} - 1$，否则Q所对应的节点对就会被OSBPW连接从而使$\mathcal{A}'$进一步扩展。根据引理 5.5，

$$m \geq \sum_{Q \in \mathcal{O}_U} [l_w(Q) + 1] > \sqrt{m} \cdot |\mathcal{O}_U| \tag{5-7}$$

因此，$|\mathcal{O}_U| < \sqrt{m}$。

对于$\mathcal{O}_I$中的每条路径Q来说，我们以贪心的方式将其标记到某条与之相互干扰的路径$P \in \mathcal{A}'$上。根据引理 5.6，对于每条路径$P \in \mathcal{A}'$，至多有$l_w(P) + 1$条$\mathcal{O}_I$中的路径被标记到P上。对于每条路径$P \in \mathcal{A}'$，算法 5-2 保证$l_w(P) \leq \sqrt{m} - 1$。因此，$|\mathcal{O}_I| \leq \sqrt{m} \cdot |\mathcal{A}'|$并且

$$|\mathcal{O}| = |\mathcal{O}_I| + |\mathcal{O}_U| < \sqrt{m} \cdot |\mathcal{A}'| + \sqrt{m} = \sqrt{m} \cdot (|\mathcal{A}'| + 1) \tag{5-8}$$

证明完成。 □

除了在线的优势，OSBPW的另一个优势在于它的简单且容易实现，这使得我们可以为其设计一个适合于传感器网络的分布式版本。回忆一下G是给定网络的拓扑图。图G中存在边(u, v)当且仅当网络中有一个连接u和v的无线链接。让v代表网络中的任意节点。我们假设v知道它的邻居节点的个数$d_G(v)$并由此计

算出函数值$w(v) = d_G(v) - 1$。除此之外，对于每个需要被连接的节点对$\{s_i, t_i\}$，节点v保存了3个变量$l_w^i(v), LH_i(v)$和$NH_i(v)$。变量$l_w^i(v)$记录了所发现的从s_i到v的w最短路径的w长度。在算法开始的时候，$l_w^i(v) = +\infty$。如果v被选为$s_i \sim t_i$路径上的节点，$LH_i(v)$记录了这条路径上v的上一跳节点（距离s更近的邻居），$NH_i(v)$记录了这条路径上v的下一跳节点（距离t更近的邻居）。

```
Input: 网络中的k个节点对{s_1,t_1},...,{s_k,t_k}
Output: 一些连接这些节点对的非干扰路径
/* 为了处理节点对{s_i,t_i}                                          */
1  对于 节点s_i 的代码:
2      l_w^i(s_i) ← w(s_i);
3      广播一个包含l_w^i(s_i)的建立路径消息;
4  对于 每个节点v ≠ s_i,t_i 的代码:
5      while 接收到节点u发送的包含l_w^i(u)的建立路径消息 do
6          if l_w^i(u) + w(v) < l_w^i(v) then
7              l_w^i(v) ← l_w^i(u) + w(v);
8              LH_i(v) ← u;
9              广播一个包含l_w^i(v)的建立路径消息;
10     while 接收到节点u发送的路径确认消息 do
11         NH_i(v) ← u;
12         将这个路径确认消息转发给LH_i(v);
13         通知其邻居节点将它们从网络中删除;
14 对于 节点t_i 的代码:
15     等待路径建立过程结束;
16     if l_w^i(t_i) ≤ √m − 1 then
17         向LH_i(t_i)发送一个路径确认消息;
```

算法 5-3 OSBPW（分布式实现）

Algo. 5-3 OSBPW (Distributed Implementation)

OSBPW的分布式版本由算法 5-3 给出。这是处理每个节点对$\{s_i, t_i\}$的代码。开始的时候，节点s_i广播一个包含$l_w^i(s_i)$的建立路径消息（算法 5-3 中第3行）。通过在网络中交换建立路径消息（每个消息中包含了发送者的l_w^i），网络中的每个节点发现从s_i到其自身的w最短路径（算法 5-3 中第4-9行）。在建立路径过

程结束后，节点t_i检查从s_i到其自身的w最短路径的w长度是否小于等于$\sqrt{m}-1$。如果是，t_i向$LH_i(t_i)$发送一个路径确认消息（算法 5-3 中第14-17行）。每个接收到这个路径确认消息的节点将其转发给它的LH_i节点，并通知其邻居节点将它们从网络中删除（算法 5-3 中第10-13行）。

很明显，算法 5-3 是OSBPW的一个正确的分布式实现。

5.7 松弛非干扰路径

对于图$G=(V,E)$中的节点对$\{s_i,t_i\}$和$\{s_j,t_j\}$来说，如果$s_i=s_j$或者$(s_i,t_j)\in E$，那么任何$s_i\sim t_i$路径和$s_j\sim t_j$路径都会互相干扰。很多时候，用户不希望因为这些“与生俱来”的干扰而拒绝这两个路由请求中的一个。因此，我们提出了松弛非干扰路径的概念。相对地，定义 5.1 中所定义的非干扰路径又被称作严格非干扰路径。

给定$G=(V,E)$中的两个节点对$\{s_i,t_i\}$和$\{s_j,t_j\}$，定义边集合$\mathcal{E}_{i,j}=\{(u,v)\in E\mid u=s_i\text{或}t_i, v=s_j\text{或}t_j\}$。定义节点集合$\mathcal{N}_{i,j}=\{s_i,t_i\}\cap\{s_j,t_j\}$。

定义 5.3 在图$G=(V,E)$中，$s_i\sim t_i$路径P_i和$s_j\sim t_j$路径P_j是松弛非干扰路径如果边集合$I_{i,j}=\emptyset$，其中

$$I_{i,j}=\{(u,v)\in E\mid u\in P_1, v\in P_2, u \text{ or } v\notin \mathcal{N}_{i,j}\}\setminus\mathcal{E}_{i,j}$$

非干扰路径问题的松弛版本为（最多）松弛非干扰路径问题，其定义为：给定图$G=(V,E)$中的k个节点对$\{s_1,t_1\},\ldots,\{s_k,t_k\}$，用松弛非干扰路径来连接最多的节点对。在前面证明非干扰路径问题的难度和近似难度的时候，给定的节点对之间没有公共节点，也没有连接不同节点对的边。在这种情况下，松弛非干扰路径问题等同于非干扰路径问题。所以本章中非干扰路径问题的难度和近似难度的结论同样适用于松弛非干扰路径问题。

对于一个松弛非干扰路径问题中的两个节点对$\{s_i,t_i\}$和$\{s_j,t_j\}$：如果$s_i=s_j$，我们可以构造一个新节点s_i'，添加边来连接s_i'和s_i的所有邻居，并将给定的节点对$\{s_j,t_j\}$替换为$\{s_i',t_j\}$；如果$(s_i,t_j)\in E$，我们可以从图中删除这条边。通过这种方法，我们可以将松弛非干扰路径问题转化为非干扰路径问题，并可以用本章提出的算法来解决松弛非干扰路径问题。

5.8 实验结果

我们用一个C++编写的模拟器来验证算法的效率。这个模拟器是一个基于应用层的程序，忽略了MAC层的细节。每个数据包包含了4个字节的包头来分

别表示这个数据包的：目的节点的ID；发送者ID；数据包类型；数据包长度。每个接收数据和时钟到时事件被加上时间戳统一放到处理事件的堆中。堆中的事件按其时间戳的先后顺序一个一个进行处理。通过这种方式，我们来模拟网络中节点间的并行操作。为了获取每个实验数据，我们做了10次不同的实验（每次实验产生不同的节点对）来取它们结果的平均值。

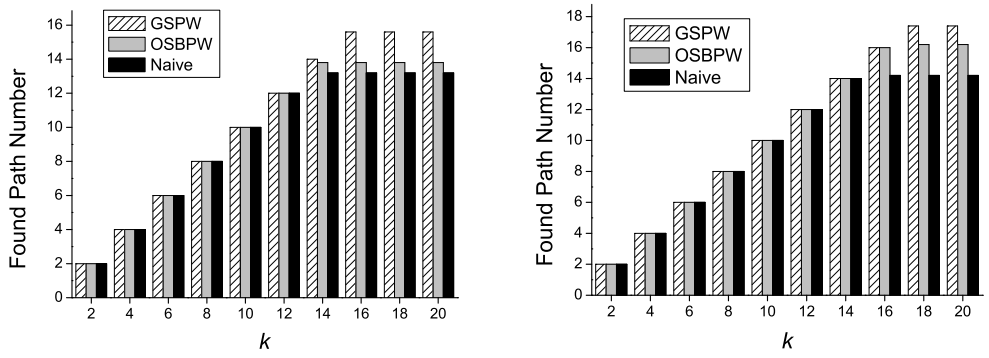
我们模拟了两个传感器网络来验证算法的效率。在第一个网络（用NLN表示）中，我们在1000m ×1000m的监测区域内布置了1100个节点。节点的位置随机产生。每个节点的通信半径被设置为50m。当两个节点的距离小于等于50m时，两个节点间存在无线链接。在第二个网络（用REN表示）中，我们还是布置了1100个节点。节点之间的链接（边）随机产生，平均每个节点有5个邻居。在模拟实验中，分别产生了$k = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20$个节点对用来表示路由需求。

算法的效率通过两个方面来衡量：算法找到的路径数量和算法的通信代价。算法找到的非干扰路径的数量就是满足的路由需求的数量。算法的通信代价由实验中总的通信量（网络中信息交换的字节数）除以节点的数量得到，也就是平均每个节点接收和发送的字节数。根据1.2.1中的描述，通信代价可以很好表示节点的能量消耗。

我们将本章中提出两个算法（GSPW和OSBPW）与一个简单的Naive算法相比较。对于$i = 1, \ldots, k$，Naive算法寻找最短的连接s_i和t_i 的路径P_i。如果存在P_i，就输出P_i并将P_i上的节点和它们的邻居节点删除。在寻找路径的时候，Naive算法采用了分布式的处理方式，通过节点间交换短小的消息建立一棵以s_i为根的最短生成树。

首先，我们来比较三个算法找到的非干扰路径数量，也就是满足的路由需求的数量。实验结果由图 5-6给出。在两个网络中，三个算法找到的路径数量都随着k的增长而增长。但是这三个算法都有个极限值，也就是它们最多能找到的非干扰路径数量。在NLN中，如图 5-6 a)所示，GSPW最多能满足大约16个路由需求，而OSBPW和Naive最多只能满足大约13 ~ 14个路由需求。在NLN中，GSPW的优势比较明显，而OSBPW的表现仅仅比Naive方法略好。在REN中，如图 5-6 b)所示，GSPW最多能满足大约17 ~ 18个路由需求，OSBPW最多能满足大约16个路由需求，而Naive方法最多只能满足大约14个路由需求。在REN中，GSPW依然具有优势，OSBPW的表现要明显好过Naive方法。

接着，我们来比较三个算法的通信代价，也就是平均每个节点接收和发送的字节数。实验结果由图 5-7给出。在两个网络中，集中式算法GSPW的通信代价一直比较稳定，因为无论k多大，GSPW都收集整个网络的拓扑信息然后做集

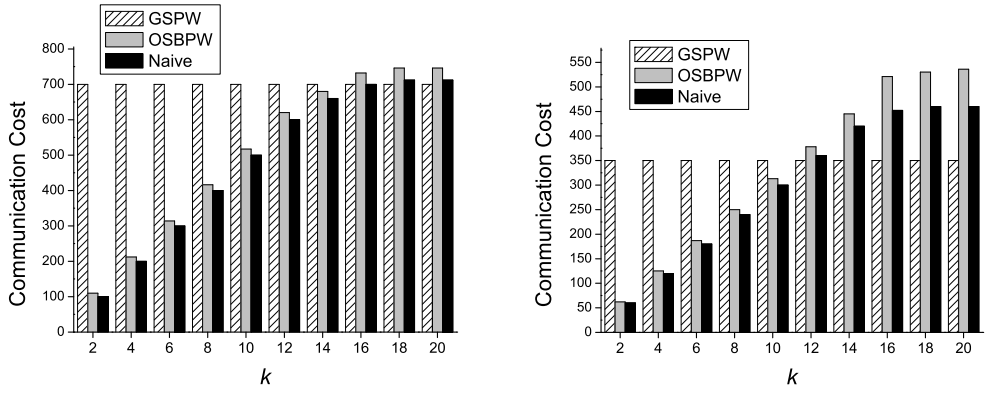


a) 在NLN中　　b) 在REN中

a) In NLN　　b) In REN

图 5-6 三个算法找到的路径数量

Fig.5-6 Number of the paths found by 3 algorithms



a) 在NLN中　　b) 在REN中

a) In NLN　　b) In REN

图 5-7 三个算法的通信代价

Fig.5-7 Communication cost of 3 algorithms

中式处理。两个分布式算法OSBPW和Naive的通信代价随着找到路径数量的增长而增长。这是因为找到更多的路径意味着这两个算法在网络中进行了更多的信息交换。如图 5-7 a)所示，在NLN中，OSBPW的通信代价仅仅比Naive 方法略高。当找到的路径数量很多时，两个分布式算法的通信代价都要超过集中式方法GSPW，在网络中执行过多次的信息交换不如一次性收集整个网络的信息。如图 5-7 b)所示，在REN中，当k较大时，OSBPW的通信代价明显比Naive方法的通信代价高。但是考虑到OSBPW可以找到更多的路径，这些高出的通信代价是完全可以接受的。当有很多路由需求时，集中式的方法GSPW更具优势。

当k较大时，无论是在找到的路径数量方面还是在通信代价方面，GSPW都比OSBPW和Naive方法的表现要好。但是OSBPW和Naive方法还是有它们的优势：在线算法和分布式处理。当路由需求一个一个地到来时，GSPW需要等待所有路由需求的产生，然后再执行算法。OSBPW和Naive方法却可以在路由需求到来时马上对其进行处理而不用考虑之后产生的路由需求。另一方面，由于其分布式的处理方式，OSBPW和Naive方法可以更好地适应网络中的拓扑变化。虽然，在通信效率方面，Naive方法比OSBPW略好，但是Naive方法并没有理论上的下界。而根据我们的证明，OSBPW可以保证几乎是最优的理论下界。

5.9 本章小结

在本章中，我们研究了传感器网络中的非干扰路径问题：给定网络中的k对节点$\{s_1,t_1\},\ldots,\{s_k,t_k\}$，用非干扰路径来连接最多的节点对。我们证明了即使当$k=2$时，这个问题也是一个NP难问题。我们还证明了非干扰路径问题的近似难度，并发现$\sqrt{m}$是可能取得的最好的近似比，其中m为网络中链接的数量。针对非干扰路径问题，我们提出了一个近似比为$\sqrt{m}$的贪心算法GSPW。此外，我们还给出了一个拥有很好理论下界的在线算法OSBPW和它的分布式实现。在今后的工作中，也许我们将考虑寻找长度受限的非干扰路径问题，或是在特殊图中，如网格和圆盘图，寻找非干扰路径的问题。

结 论

针对于无线传感器网络，本文研究了一系列在网络中寻找多条不相交或非干扰路径用于路由的问题。主要的研究成果如下：

（1）本文提出了一个高效的分布式算法EDA用于在传感器网络中寻找k条连接给定节点s和t的不相交路径。EDA采取的是增量寻找路径的方法。每次寻找新的路径，只需要网络的部分节间点交换少量的信息并从网络中收集少量信息。网络中的信息交换基于预先建立的树形结构。每个交换的数据包都具有很小的长度。用于采用这种分布式处理方式，EDA具有很高的通信效率。并且EDA 可以保证答案正确性，即如果网络中存在k条连接s和t的不相交路径，EDA一定能够输出k条不相交路径。也就是说EDA 能够找到网络中最多的不相交路径。这是第一个不用收集整个网络的拓扑信息就可以保证答案正确性的方法。实验结果表明EDA 在找到路径数量和通信代价方面都具有很高的效率。

（2）本文提出了一个高效的完全分布式方法OFDP用于在传感器网络中寻找k条连接给定节点s和t的长度之和最小的不相交路径。假设在给定的网络中，每个链接都有一个长度来表示其代价。OFDP采取逐条寻找最短路径的方法增量产生不相交路径。每次找到更多的路径之后，OFDP都对网络的拓扑图做必要的变换，然后用现有的网络模拟新的拓扑图所对应的网络。通过循环地在新的网络中寻找最短路径并将该路径添加到已发现路径上，OFDP能够找到长度之和最小的不相交路径。每次循环中，网络中的节点通过彼此交换短小的数据包来寻找最短路径。OFDP不需要收集网络中的任何数据，网络中的信息交换也不依赖于任何预先实现的结构。而且，OFDP既能够保证答案正确性又能保证结果最优性。所谓结果最优性是指OFDP输出的路径集合在所有的可能解中拥有最小的长度之和。这是第一个不用收集任何拓扑信息就可以保证答案正确性和结果最优性的方法。在不考虑路径长度的情况下，通过对OFDP进行简化，可以得到另一个用于寻找k条不相交$s \sim t$路径的完全分布式算法FDP。FDP可以保证答案正确性，而且不需要收集任何拓扑信息。实验结果表明这两个算法在找到路径的数量和质量方面，以及通信代价方面，都具有很高的效率。

（3）本文提出了一个$\sqrt{n}$近似算法GDA用来在传感器网络中寻找最多的连接给定节点s和t的不相交路径，其中每条路径的长度都小于等于用户指定的长度限制L。这里，n是网络中节点的数量。GDA采用贪心地逐条寻找最短路径的策略。本文还给出了GDA的分布式实现。GDA 是关于这个问题的第一个可

分析的近似算法。此外，针对这个问题，本文还给出了一个分布式的启发算法OPDA。OPDA仅仅通过在网络节点间进行一次信息交换就可以得到多条满足长度限制的不相交路径，而且OPDA可以有效地应对连接的长度是任意非负数的情况。实验结果表明无论是在找到的路径数量方面还是在通信代价方面，OPDA都是一个高效的算法。

（4）本文研究了（广义）非干扰路径问题：给定网络中的k对节点$\{s_1,t_1\},\ldots,\{s_k,t_k\}$，用非干扰路径来连接最多的节点对。本文证明了即使当$k=2$时，这个问题也是一个NP难问题。本文还证明了对于任意的k和$\epsilon>0$，以$m^{1/2-\epsilon}$来近似这个问题是NP难的。其中，m是网络中链接的数量。这是关于这个问题的第一个近似难度的结论。本文给出了这个问题的一个$\sqrt{m}$近似算法GSPW。这也是最优的近似算法。本文还给出了这个问题的一个在线算法OSBPW，以及OSBPW的分布式实现。OSBPW拥有接近于最优的理论下界。

下一步的研究工作将主要考虑如下问题：

（1）长度受限的不相交路径问题的近似难度。

（2）给定多对路由需求，如何在网络中优化地布置这些路由。优化的目标包括：网络拥塞最小（也就是通过每条边的路径数量最少），路径间的干扰最小（包括如何对干扰进行定量研究）等等。

（3）当网络的拓扑图为特殊图（例如，单位圆盘图、网格图）时，重新研究非干扰路径问题的难度以及算法。

参考文献

[1] 孙利民, 李建中, 陈渝, 等. 无线传感器网络[M]. 第一版., 北京: 清华大学出版社, 2005:3–10.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, et al. A Survey on Sensor Networks[J]. IEEE Communications Magazine, 2002, 40(8):102–114.

[3] C. Xueli, C. Wenyan, C. Suhua, et al. Role of Wireless Sensor Networks in Forest Fire Prevention[C]//2nd International Conference on Computer Engineering and Technology (ICCET). 2010:V4–12–V4–14.

[4] A. Mainwaring, D. Culler, J. Polastre, et al. Wireless Sensor Networks for Habitat Monitoring[C]//Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. New York, NY, USA, 2002:88–97.

[5] J. Yick, B. Mukherjee, D. Ghosal. Wireless Sensor Network Survey[J]. Computer Networks, 2008, 52(12):2292–2330.

[6] D. J. Cook, J. C. Augusto, V. R. Jakkula. Ambient Intelligence: Technologies, Applications, and Opportunities[J]. Pervasive and Mobile Computing, 2009, 5(4):277–298.

[7] M. Li, Y. Liu. Underground Coal Mine Monitoring with Wireless Sensor Networks[J]. ACM Transactions on Sensor Networks (TOSN), 2009, 5:10:1–10:29.

[8] J. Song, S. Han, A. Mok, et al. Wirelesshart: Applying Wireless Technology in Real-time Industrial Process Control[C]//Real-Time and Embedded Technology and Applications Symposium. 2008:377–386.

[9] Y. Charfi, N. Wakamiya, M. Murata. Challenging Issues in Visual Sensor Networks[J]. IEEE Wireless Communications, 2009, 16(2):44–49.

[10] C. Buratti, A. Conti, D. Dardari, et al. An Overview on Wireless Sensor Networks Technology and Evolution[J]. Sensors, 2009, 9(9):6869–6896.

[11] S. Seitinger, D. S. Perry, W. J. Mitchell. Urban Pixels: Painting the City with Light[C]//Proceedings of the 27th international conference on Human factors in computing systems. New York, NY, USA: ACM Press, 2009:839–848.

[12] S. Kim, S. Pakzad, D. Culler, et al. Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks[C]//6th International Symposium on Information Processing in Sensor Networks (IPSN). 2007:254–263.

[13] V. Gungor, G. Hancke. Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches[J]. IEEE Transactions on Industrial Electronics, 2009, 56(10):4258–4265.

[14] M. Tubaishat, P. Zhuang, Q. Qi, et al. Wireless Sensor Networks in Intelligent Transportation Systems[J]. Wireless Communications and Mobile Computing, 2009, 9(3):287–302.

[15] 高峰, 俞立, 张文安, 等. 基于无线传感器网络的作物水分状况监测系统研究与设计[J]. 农业工程学报, 2009, 25(2):107–112.

[16] 蔡义华, 刘刚, 李莉, 等. 基于无线传感器网络的农田信息采集节点设计与试验[J]. 农业工程学报, 2009, 25(4):176–178.

[17] B. Warneke, M. Last, B. Liebowitz, et al. Smart Dust: Communicating with a Cubic-millimeter Computer[J]. Computer, 2001, 34(1):44–51.

[18] W. Dargie, C. Poellabauer. Fundamentals of Wireless Sensor Networks: Theory and Practice[M]. John Wiley and Sons, 2010:168–183.

[19] K. Sohraby, D. Minoli, T. Znati. Wireless Sensor Networks: Technology, Protocols, and Applications[M]. John Wiley and Sons, 2007:203–209.

[20] 贾伯年, 俞朴, 宋爱国. 传感器技术[M]. 第三版., 南京: 东南大学出版社, 2007:10–20.

[21] 范茂军. 传感器技术:信息化武器装备的神经元[M]. 第一版., 北京: 国防工业出版社, 2008:30–40.

[22] 刘笃仁, 韩保君. 传感器原理及应用技术[M]. 第一版., 西安: 西安电子科技大学出版社, 2003:21–35.

[23] 杨建红, 张认成, 房怀英. 反射强度调制式光纤声音传感器优化设计与研究[J]. 传感器与微系统, 2008, 27(12):84–87.

[24] X.-G. Sun, X.-L. Sun, Q.-G. Yang, et al. Application of Wireless Sensor Networks in Post-disaster Road Monitoring System[C]//4th International Conference on Intelligent Networks and Intelligent Systems (ICINIS). 2011:105–108.

[25] R. McGrath. Image Sensor Technology[M]//Single-Photon Imaging. Springer Berlin / Heidelberg, 2011, vol. 160:27–47.

[26] R. Robucci, J. Gray, L. K. Chiu, et al. Compressive Sensing on a Cmos Separable-transform Image Sensor[J]. Proceedings of the IEEE, 2010, 98(6):1089–1101.

[27] J. Hill, R. Szewczyk, A. Woo, et al. Tinyos: An Operating System for Sensor Networks[M]//Ambient Intelligence. Springer-Verlag, 2005.

[28] C. chieh Han, R. Kumar, R. Shea, et al. Sos: A Dynamic Operating System for Sensor Networks[C]//Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisy). ACM Press, 2005.

[29] S. Bhatti, J. Carlson, H. Dai, et al. Mantis Os: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms[J]. Mobile Networks and Applications, 2005, 10:563–579.

[30] F. Chen, T. Talanis, R. German, et al. Real-time Enabled Ieee 802.15.4 Sensor Networks in Industrial Automation[C]//IEEE International Symposium on Industrial Embedded Systems (SIES). 2009:136–139.

[31] I. Rhee, A. Warrier, M. Aia, et al. Z-mac: A Hybrid Mac for Wireless Sensor Networks[J]. IEEE/ACM Transactions on Networking (TON), 2008, 16:511–524.

[32] 李建中, 高宏. 无线传感器网络的研究进展[J]. 计算机研究与发展, 2008, 45:1–15.

[33] 陈林星. 无线传感器网络技术与应用[M]. 第一版., 电子工业出版社, 2009:1–15.

[34] D. Maier, A. Kleiner. Improved Gps Sensor Model for Mobile Robots in Urban Terrain[C]//IEEE International Conference on Robotics and Automation (ICRA). 2010:4385–4390.

[35] O. Montenbruck, T. Ebinuma, E. Lightsey, et al. A Real-time Kinematic Gps Sensor for Spacecraft Relative Navigation[J]. Aerospace Science and Technology, 2002, 6(6):435–449.

[36] B. Hull, V. Bychkovsky, Y. Zhang, et al. Cartel: A Distributed Mobile Sensor Computing System[C]//Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys). New York, NY, USA: ACM Press, 2006:125–138.

[37] A. Howard, M. J. Mataric, G. S. Sukhatme. Mobile Sensor Network Deployment Using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem[C]//Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS). 2002:299–308.

[38] T. Liu, C. M. Sadler, P. Zhang, et al. Implementing Software on Resource-constrained Mobile Sensors: Experiences with Impala and Zebranet[C]//Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys). New York, NY, USA: ACM Press, 2004:256–269.

[39] Y. Yang, L. Wang, D. K. Noh, et al. Solarstore: Enhancing Data Reliability in Solar-powered Storage-centric Sensor Networks[C]//Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys). New York, NY, USA: ACM Press, 2009:333–346.

[40] M. Danesh, J. Long. An Autonomous Wireless Sensor Node Incorporating a Solar Cell Antenna for Energy Harvesting[J]. IEEE Transactions on Microwave Theory and Techniques, 2011, 59(12):3546–3555.

[41] D. Estrin, R. Govindan, J. Heidemann, et al. Next Century Challenges: Scalable Coordination in Sensor Networks[C]//Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom). New York, NY, USA: ACM Press, 1999:263–270.

[42] D. Estrin. Tutorial "wireless Sensor Networks" Part Iv: Sensor Network Protocols[C]//Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom). 2002.

[43] P. Levis, N. Lee, M. Welsh, et al. Tossim: Accurate and Scalable Simulation of Entire Tinyos Applications[C]//Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys). New York, NY, USA: ACM Press, 2003:126–137.

[44] Q. Cao, T. Abdelzaher, J. Stankovic, et al. The Liteos Operating System: Towards Unix-like Abstractions for Wireless Sensor Networks[C]//International Symposium on Information Processing in Sensor Networks (IPSN). 2008:233–244.

[45] H. Cha, S. Choi, I. Jung, et al. Retos: Resilient, Expandable, and Threaded Operating System for Wireless Sensor Networks[C]//International Symposium on Information Processing in Sensor Networks (IPSN). 2007:148–157.

[46] J. Gutierrez, M. Naeve, E. Callaway, et al. Ieee 802.15.4: A Developing Standard for Low-power Low-cost Wireless Personal Area Networks[J]. IEE Network, 2001, 15(5):12–19.

[47] W. Ye, J. Heidemann, D. Estrin. An Energy-efficient Mac Protocol for Wireless Sensor Networks[C]//Proceedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). 2002:1567– 1576 vol.3.

[48] T. van Dam, K. Langendoen. An Adaptive Energy-efficient Mac Protocol for Wireless Sensor Networks[C]//Proceedings of the 1st international conference on Em-

bedded networked sensor systems (SenSys). New York, NY, USA: ACM Press, 2003:171–180.

[49] M. Buettner, G. V. Yee, E. Anderson, et al. X-mac: A Short Preamble Mac Protocol for Duty-cycled Wireless Sensor Networks[C]//Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys). New York, NY, USA: ACM Press, 2006:307–320.

[50] I. Rhee, A. Warrier, M. Aia, et al. Z-mac: A Hybrid Mac for Wireless Sensor Networks[J]. IEEE/ACM Transactions on Networking (TON), 2008, 16:511–524.

[51] Y. Sun, O. Gurewitz, D. B. Johnson. Ri-mac: A Receiver-initiated Asynchronous Duty Cycle Mac Protocol for Dynamic Traffic Loads in Wireless Sensor Networks[C]//Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys). New York, NY, USA: ACM Press, 2008:1–14.

[52] S. Liu, K.-W. Fan, P. Sinha. Cmac: An Energy-efficient Mac Layer Protocol Using Convergent Packet Forwarding for Wireless Sensor Networks[J]. ACM Transactions on Sensor Networks (TOSN), 2009, 5:29:1–29:34.

[53] Y. Sun, S. Du, O. Gurewitz, et al. Dw-mac: A Low Latency, Energy Efficient Demand-wakeup Mac Protocol for Wireless Sensor Networks[C]//Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc). New York, NY, USA: ACM Press, 2008:53–62.

[54] J. Elson, L. Girod, D. Estrin. Fine-grained Network Time Synchronization Using Reference Broadcasts[J]. SIGOPS Oper. Syst. Rev., 2002, 36:147–163.

[55] S. Ganeriwal, R. Kumar, M. B. Srivastava. Timing-sync Protocol for Sensor Networks[C]//Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys). New York, NY, USA: ACM, 2003:138–149.

[56] M. Sichitiu, C. Veerarittiphan. Simple, Accurate Time Synchronization for Wireless Sensor Networks[C]//IEEE Wireless Communications and Networking (WCNC). 2003:1266–1273 vol.2.

[57] P. Sommer, R. Wattenhofer. Gradient Clock Synchronization in Wireless Sensor Networks[C]//Proceedings of International Conference on Information Processing in Sensor Networks (IPSN). Washington, DC, USA: IEEE Computer Society, 2009:37–48.

[58] T. Zhu, Z. Zhong, Y. Gu, et al. Leakage-aware Energy Synchronization for Wireless Sensor Networks[C]//Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys). New York, NY, USA: ACM, 2009:319–332.

[59] 王喆, 王福豹, 陈振华. 嵌入路由报头的无限传感器网络时间同步算法[J]. 计算机工程, 2009, 35:132–134.

[60] S. Ganeriwal, C. Pöpper, S. Čapkun, et al. Secure Time Synchronization in Sensor Networks[J]. ACM Trans. Inf. Syst. Secur., 2008, 11:23:1–23:35.

[61] G. Shen, R. Zetik, H. Yan, et al. Time of Arrival Estimation for Range-based Localization in Uwb Sensor Networks[C]//IEEE International Conference on Ultra-Wideband (ICUWB). 2010:1–4.

[62] P. Bahl, V. Padmanabhan. Radar: An In-building Rf-based User Location and Tracking System[C]//Proceedings of IEEE Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). 2000:775–784 vol.2.

[63] D. Niculescu, B. Nath. Ad Hoc Positioning System (aps) Using Aoa[C]//IEEE Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM). 2003:1734–1743 vol.3.

[64] G. Zanca, F. Zorzi, A. Zanella, et al. Experimental Comparison of Rssi-based Localization Algorithms for Indoor Wireless Sensor Networks[C]//Proceedings of the workshop on Real-world wireless sensor networks (REALWSN). New York, NY, USA: ACM, 2008:1–5.

[65] J. Blumenthal, R. Grossmann, F. Golatowski, et al. Weighted Centroid Localization in Zigbee-based Sensor Networks[C]//IEEE International Symposium on Intelligent Signal Processing (WISP). 2007:1–6.

[66] T. He, C. Huang, B. M. Blum, et al. Range-free Localization Schemes for Large Scale Sensor Networks[C]//Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom). New York, NY, USA: ACM, 2003:81–95.

[67] D. Niculescu, B. Nath. Dv Based Positioning in Ad Hoc Networks[J]. Telecommunication Systems, 2003, 22:267–280.

[68] P. Santi. Topology Control in Wireless Ad Hoc and Sensor Networks[J]. ACM Computing Surveys (CSUR), 2005, 37:164–194.

[69] Y. Wang. Topology Control for Wireless Sensor Networks[M]//Wireless Sensor Networks and Applications. Springer US, 2008:113–147.

[70] S. Lin, J. Zhang, G. Zhou, et al. Atpc: Adaptive Transmission Power Control for Wireless Sensor Networks[C]//Proceedings of the 4th international conference

on Embedded networked sensor systems (SenSys). New York, NY, USA: ACM, 2006:223–236.

[71] O. Younis, S. Fahmy. Heed: A Hybrid, Energy-efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks[J]. IEEE Transactions on Mobile Computing, 2004, 3(4):366–379.

[72] W. Heinzelman, A. Chandrakasan, H. Balakrishnan. Energy-efficient Communication Protocol for Wireless Microsensor Networks[C]//Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. 2000:10 pp. vol.2.

[73] N. Xu, A. Huang, T.-W. Hou, et al. Coverage and Connectivity Guaranteed Topology Control Algorithm for Cluster-based Wireless Sensor Networks[J]. Wireless Communications and Mobile Computing, 2012, 12(1):23–32.

[74] S. Poduri, S. Pattem, B. Krishnamachari, et al. Using Local Geometry for Tunable Topology Control in Sensor Networks[J]. IEEE Transactions on Mobile Computing, 2009, 8(2):218–230.

[75] Y. Ding, C. Wang, L. Xiao. An Adaptive Partitioning Scheme for Sleep Scheduling and Topology Control in Wireless Sensor Networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(9):1352–1365.

[76] S. R. Madden, M. J. Franklin, J. M. Hellerstein, et al. Tinydb: An Acquisitional Query Processing System for Sensor Networks[J]. ACM Transactions on Database Systems (TODS), 2005, 30:122–173.

[77] Y. Yao, J. Gehrke. The Cougar Approach to In-network Query Processing in Sensor Networks[J]. SIGMOD Rec., 2002, 31:9–18.

[78] B. Greenstein, D. Estrin, R. Govindan, et al. Difs: A Distributed Index for Features in Sensor Networks[C]//Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications. 2003:163–173.

[79] D. Ganesan, D. Estrin, J. Heidemann. Dimensions: Why Do We Need a New Data Handling Architecture for Sensor Networks[J]. ACM SIGCOMM Computer Communication Review, 2003, 33:143–148.

[80] G. Mathur, P. Desnoyers, P. Chukiu, et al. Ultra-low Power Data Storage for Sensor Networks[J]. ACM Transactions on Sensor Networks (TOSN), 2009, 5:1–34.

[81] M. Shao, S. Zhu, W. Zhang, et al. Pdcs: Security and Privacy Support for Data-centric Sensor Networks[J]. IEEE Transactions on Mobile Computing, 2009, 8(8):1023–1038.

[82] S. Madden, M. J. Franklin, J. M. Hellerstein, et al. Tag: A Tiny Aggregation Service for Ad-hoc Sensor Networks[J]. ACM SIGOPS Operating Systems Review, 2002, 36:131–146.

[83] G. Xing, R. Tan, B. Liu, et al. Data Fusion Improves the Coverage of Wireless Sensor Networks[C]//Proceedings of the 15th annual international conference on Mobile computing and networking (MobiCom). New York, NY, USA: ACM, 2009:157–168.

[84] H. Luo, H. Tao, H. Ma, et al. Data Fusion with Desired Reliability in Wireless Sensor Networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(3):501–513.

[85] R. Tan, G. Xing, B. Liu, et al. Impact of Data Fusion on Real-time Detection in Sensor Networks[C]//30th IEEE Real-Time Systems Symposium (RTSS). 2009:323–332.

[86] X. Yang, H. B. Lim, T. M. Özsu, et al. In-network Execution of Monitoring Queries in Sensor Networks[C]//Proceedings of the 2007 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, 2007:521–532.

[87] J. Gehrke, S. Madden. Query Processing in Sensor Networks[J]. IEEE Pervasive Computing, 2004, 3(1):46–55.

[88] A. Meliou, C. Guestrin, J. Hellerstein. Approximating Sensor Network Queries Using In-network Summaries[C]//International Conference on Information Processing in Sensor Networks (IPSN). 2009:229–240.

[89] K.-C. Kim, B.-J. Oh. An Energy Efficient Filtering Approach to In-network Join Processing in Sensor Network Databases[M]//Multimedia, Computer Graphics and Broadcasting. Springer Berlin Heidelberg, 2011, vol. 263:116–122.

[90] M. Ye, X. Liu, W.-C. Lee, et al. Probabilistic Top-k Query Processing in Distributed Sensor Networks[C]//IEEE 26th International Conference on Data Engineering (ICDE). 2010:585–588.

[91] M. A. Osborne, S. J. Roberts, A. Rogers, et al. Towards Real-time Information Processing of Sensor Network Data Using Computationally Efficient Multi-output Gaussian Processes[C]//Proceedings of the 7th international conference on Information processing in sensor networks (IPSN). Washington, DC, USA: IEEE Computer Society, 2008:109–120.

[92] A. Brayner, A. Lopes, D. Meira, et al. An Adaptive In-network Aggregation Operator for Query Processing in Wireless Sensor Networks[J]. Journal of Systems and Software, 2008, 81(3):328–342.

[93] J. Zhang, V. Varadharajan. Wireless Sensor Network Key Management Survey and Taxonomy[J]. Journal of Network and Computer Applications, 2010, 33(2):63–75.

[94] S. Bag, A. Saha, P. Sarkar. Highly Resilient Key Predistribution Scheme Using Transversal Designs and Reed Muller Codes for Wireless Sensor Network[M]//Advances in Network Security and Applications. Springer Berlin Heidelberg, 2011, vol. 196:344–355.

[95] X. Du, M. Guizani, Y. Xiao, et al. Transactions Papers a Routing-driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks[J]. IEEE Transactions on Wireless Communications, 2009, 8(3):1223–1229.

[96] L. B. Oliveira, D. F. Aranha, C. P. Gouvêa, et al. Tinypbc: Pairings for Authenticated Identity-based Non-interactive Key Distribution in Sensor Networks[J]. Computer Communications, 2011, 34(3):485–493.

[97] Y. Keung, B. Li, Q. Zhang. The Intrusion Detection in Mobile Sensor Network[C]//Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing (MobiHoc). New York, NY, USA: ACM, 2010:11–20.

[98] Y. Wang, X. Wang, B. Xie, et al. Intrusion Detection in Homogeneous and Heterogeneous Wireless Sensor Networks[J]. IEEE Transactions on Mobile Computing, 2008, 7(6):698–711.

[99] S. Shin, T. Kwon, G.-Y. Jo, et al. An Experimental Study of Hierarchical Intrusion Detection for Wireless Industrial Sensor Networks[J]. IEEE Transactions on Industrial Informatics, 2010, 6(4):744–757.

[100] S. Madria, J. Yin. Serwa: A Secure Routing Protocol Against Wormhole Attacks in Sensor Networks[J]. Ad Hoc Networks, 2009, 7(6):1051–1063.

[101] K. Zhang, C. Wang, C. Wang. A Secure Routing Protocol for Cluster-based Wireless Sensor Networks Using Group Key Management[C]//4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM). 2008:1–5.

[102] R. Lu, X. Lin, H. Zhu, et al. Becan: A Bandwidth-efficient Cooperative Authentication Scheme for Filtering Injected False Data in Wireless Sensor Networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(1):32–43.

[103] S. Zheng, T. Jiang, J. Baras. Robust State Estimation under False Data Injection in Distributed Sensor Networks[C]//IEEE Global Telecommunications Conference (GLOBECOM). 2010:1–5.

[104] 唐勇, 周明天, 张欣. 无线传感器网络路由协议研究进展[J]. 软件学报, 2006, 17(3):410–421.

[105] M. Maróti. Directed Flood-routing Framework for Wireless Sensor Networks[C]//Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware. New York, NY, USA: Springer-Verlag New York, Inc., 2004:99–114.

[106] Z. Haas, J. Halpern, L. Li. Gossip-based Ad Hoc Routing[C]//Proceedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). 2002, 3:1707– 1716.

[107] C. Perkins, E. Royer. Ad-hoc On-demand Distance Vector Routing[C]//Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA). 1999:90–100.

[108] D. B. Johnson, D. A. Maltz, J. Broch. Ad Hoc Networking[M]//Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001:139–172.

[109] R. Shah, J. Rabaey. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks[C]//IEEE Wireless Communications and Networking Conference (WCNC). 2002, 1:350–355.

[110] A. Manjeshwar, D. Agrawal. Teen: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks[C]//Proceedings 15th International Parallel and Distributed Processing Symposium. 2001:2009–2015.

[111] C. Intanagonwiwat, R. Govindan, D. Estrin, et al. Directed Diffusion for Wireless Sensor Networking[J]. IEEE/ACM Transactions on Networking (TON), 2003, 11:2–16.

[112] C. Schurgers, M. Srivastava. Energy Efficient Routing in Wireless Sensor Networks[C]//IEEE Military Communications Conference (MILCOM). 2001, 1:357–361.

[113] M. Chu, H. Haussecker, F. Zhao, et al. Scalable Information-driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks[J]. International Journal of High Performance Computing Applications, 2002, 16:293–313.

[114] D. Braginsky, D. Estrin. Rumor Routing Algorithm for Sensor Networks[C]//Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA). New York, NY, USA: ACM, 2002:22–31.

[115] B. Karp, H. T. Kung. Gpsr: Greedy Perimeter Stateless Routing for Wireless Networks[C]//Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom). New York, NY, USA: ACM, 2000:243–254.

[116] Y. Yu, R. Govindan, D. Estrin. Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks[R]. Tech. rep., University of California at Los Angeles, 2001.

[117] T. He, J. Stankovic, C. Lu, et al. Speed: A Stateless Protocol for Real-time Communication in Sensor Networks[C]//Proceedings of 23rd International Conference on Distributed Computing Systems. 2003:46–55.

[118] K.-I. Kawarabayashi, Y. Kobayashi. The Induced Disjoint Paths Problem[C]//Proceedings of the 13th international conference on Integer programming and combinatorial optimization (IPCO). Berlin, Heidelberg: Springer-Verlag, 2008:47–61.

[119] S. Waharte, R. Boutaba. On the Probability of Finding Non-interfering Paths in Wireless Multihop Networks[C]//Proceedings of the 7th international IFIP-TC6 networking conference on AdHoc and sensor networks, wireless networks, next generation internet. Berlin, Heidelberg: Springer-Verlag, 2008:914–921.

[120] K. Ishida, Y. Kakuda, T. Kikuno. A Routing Protocol for Finding Two Node-disjoint Paths in Computer Networks[C]//Proceedings of the 1995 International Conference on Network Protocols (ICNP). Washington, DC, USA: IEEE Computer Society, 1995:340–.

[121] J. W. Suurballe, R. E. Tarjan. A Quick Method for Finding Shortest Pairs of Disjoint Paths[J]. Networks, 1984, 14(2):325–336.

[122] R. Ogier, N. Shacham. A Distributed Algorithm for Finding Shortest Pairs of Disjoint Paths[C]//Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). 1989, 1:173–182.

[123] Y. O. Lee, A. Reddy. Disjoint Multi-path Routing and Failure Recovery[C]//IEEE International Conference on Communications (ICC). 2010:1–6.

[124] T. Hashiguchi, K. Tajima, Y. Takita, et al. Node-disjoint Paths Search in Wdm Networks with Asymmetric Nodes[C]//15th International Conference on Optical Network Design and Modeling (ONDM). 2011:1–6.

[125] M. Griffin, T. Korkmaz. Distributed Verification of Global Multiple Disjoint Paths in Mobile Wireless Networks[C]//7th International Wireless Communications and Mobile Computing Conference (IWCMC). 2011:666–670.

[126] C.-L. Li, S. McCormick, D. Simchi-Levi. The Complexity of Finding Two Disjoint Paths with Min-max Objective Function[J]. Discrete Applied Mathematics, 1990, 26(1):105–115.

[127] D. Ganesan, R. Govindan, S. Shenker, et al. Highly-resilient, Energy-efficient Multipath Routing in Wireless Sensor Networks[J]. ACM SIGMOBILE Mobile Computing and Communications Review, 2001, 5:11–25.

[128] S. Li, Z. Wu. Node-disjoint Parallel Multi-path Routing in Wireless Sensor Networks[C]//Second International Conference on Embedded Software and Systems. 2005.

[129] A. Kumar, S. Varma. Geographic Node-disjoint Path Routing for Wireless Sensor Networks[J]. IEEE Sensors Journal, 2010, 10(6):1138–1139.

[130] J. W. Baek, Y. J. Nam, D.-W. Seo. An Energy-efficient K-disjoint-path Routing Algorithm for Reliable Wireless Sensor Networks[C]//Proceedings of the 5th IFIP WG 10.2 international conference on Software technologies for embedded and ubiquitous systems. Berlin, Heidelberg: Springer-Verlag, 2007:399–408.

[131] B. Deb, S. Bhatnagar, B. Nath. Reinform: Reliable Information Forwarding Using Multiple Paths in Sensor Networks[C]//Proceedings of 28th Annual IEEE International Conference on Local Computer Networks (LCN). 2003:406–415.

[132] J. Shi. A Multipath Routing Algorithm for Wireless Sensor Networks[M]//Ubiquitous Intelligence and Computing. Springer Berlin / Heidelberg, 2006, vol. 4159:438–446.

[133] S. Omar, M. Zoulikha, B. Cousin. Energy Efficiency in Ad Hoc Wireless Networks with Node-disjoint Path Routing[C]//Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on. 2011:127–130.

[134] D. Sidhu, R. Nair, S. Abdallah. Finding Disjoint Paths in Networks[C]//Proceedings of the conference on Communications architecture & protocols (SIGCOMM). New York, NY, USA: ACM, 1991:43–51.

[135] S. Khuller, B. Schieber. Efficient Parallel Algorithms for Testing Connectivity and Finding Disjoint S-t Paths in Graphs[C]//30th Annual Symposium on Foundations of Computer Science. 1989:288–293.

[136] K. Iwama, C. Iwamoto, T. Ohsawaa. A Faster Parallel Algorithm for K-connectivity[J]. Information Processing Letters, 1997, 61(5):265–269.

[137] J. W. Suurballe. Disjoint Paths in a Network[J]. Networks, 1974, 4(2):125–145.

[138] L. R. Ford, D. R. Fulkerson. Maximal Flow Through a Network[J]. Canadian Journal of Mathematics, 1956, 8:399–404.

[139] R. Bhandari. Optimal Physical Diversity Algorithms and Survivable Networks[C]//Proceedings of the 2nd IEEE Symposium on Computers and Communications (ISCC). Washington, DC, USA: IEEE Computer Society, 1997:433–441.

[140] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs[J]. Numerische Mathematik, 1959, 1:269–271.

[141] A. Srinivas, E. Modiano. Minimum Energy Disjoint Path Routing in Wireless Ad-hoc Networks[C]//Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom). New York, NY, USA: ACM, 2003:122–133.

[142] C.-L. Li, S. Thomas McCormick, D. Simchi-Levi. Finding Disjoint Paths with Different Path-costs: Complexity and Algorithms[J]. Networks, 1992, 22(7):653–667.

[143] T. Gomes, J. Craveirinha, L. Jorge. An Effective Algorithm for Obtaining the Minimal Cost Pair of Disjoint Paths with Dual Arc Costs[J]. Computers and Operations Research, 2009, 36:1670–1682.

[144] J. Rak. K-penalty: A Novel Approach to Find K-disjoint Paths with Differentiated Path Costs[J]. IEEE Communications Letters, 2010, 14(4):354–356.

[145] R. Leepila, E. Oki, N. Kishi. Scheme to Find K Disjoint Paths in Multi-cost[C]//Proceedings of IEEE International Conference on Communications (ICC). 2011:1–5.

[146] A. Itai, Y. Perl, Y. Shiloach. The Complexity of Finding Maximum Disjoint Paths with Length Constraints[J]. Networks, 1982, 12(3):277–286.

[147] A. Bley. On the Complexity of Vertex-disjoint Length-restricted Path Problems[J]. Computational Complexity, 2004, 12:131–149.

[148] D. Ronen, Y. Perl. Heuristics for Finding a Maximum Number of Disjoint Bounded Paths[J]. Networks, 1984, 14(4):531–544.

[149] V. Guruswami, S. Khanna, R. Rajaraman, et al. Near-optimal Hardness Results and Approximation Algorithms for Edge-disjoint Paths and Related Problems[C]//Proceedings of the thirty-first annual ACM symposium on Theory of computing (STOC). New York, NY, USA: ACM, 1999:19–28.

[150] J. M. Kleinberg. Approximation Algorithms for Disjoint Paths Problems[D]. Cambridge, USA:MIT, 1996.

[151] R. M. Karp. Reducibility Among Combinatorial Problems[M]//Complexity of Computer Computations. Plenum Press, 1972:85–103.

[152] C. Chekuri, S. Khanna, F. Shepherd. Edge-disjoint Paths in Planar Graphs[C]//Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science. 2004:71–80.

[153] M. E. Kramar, J. van Leeuwen. The Complexity of Wire Routing and Finding the Minimum Area Layouts for Arbitrary Vlsi Circuits[M]//Advances in Computer Research 2: VLSI Theory. JAI Press, 1984:129–146.

[154] N. Robertson, P. Seymour. Graph Minors .xiii. the Disjoint Paths Problem[J]. Journal of Combinatorial Theory, Series B, 1995, 63(1):65–110.

[155] S. Fortune, J. E. Hopcroft, J. C. Wyllie. The Directed Subgraph Homeomorphism Problem[R]. Tech. rep., Ithaca, NY, USA, 1978.

[156] M. Andrews, J. Chuzhoy, L. Zhang. Hardness of the Undirected Edge-disjoint Paths Problem with Congestion[C]//Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS). Washington, DC, USA: IEEE Computer Society, 2005:226–244.

[157] S. G. Kolliopoulos, C. Stein. Approximating Disjoint-path Problems Using Packing Integer Programs[J]. Mathematical Programming, 2004, 99:63–87.

[158] C. Chekuri, S. Khanna, F. B. Shepherd. An $o(\sqrt{n})$ Approximation and Integrality Gap for Disjoint Paths and Unsplittable Flow[J]. Theory of Computing, 2006, 2(1):137–146.

[159] T. Nguyen. On the Disjoint Paths Problem[J]. Operations Research Letters, 2007, 35(1):10–16.

[160] D. Wagner, K. Weihe. A Linear-time Algorithm for Edge-disjoint Paths in Planar Graphs[M]//Algorithms—ESA '93. Springer Berlin / Heidelberg, 1993, vol. 726:384–395.

[161] C. Chekuri, S. Khanna, F. B. Shepherd. Edge-disjoint Paths in Planar Graphs with Constant Congestion[C]//Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (STOC). New York, NY, USA: ACM, 2006:757–766.

[162] M. R. Pearlman, Z. J. Haas, P. Sholander, et al. On the Impact of Alternate Path Routing for Load Balancing in Mobile Ad Hoc Networks[C]//Proceedings of the

1st ACM international symposium on Mobile ad hoc networking & computing (MobiHoc). Piscataway, NJ, USA: IEEE Press, 2000:3–10.

[163] D. Bienstock. On the Complexity of Testing for Odd Holes and Induced Odd Paths[J]. Discrete Mathematics, 1991, 90:85–92.

[164] S. Waharte, R. Boutaba. Totally Disjoint Multipath Routing in Multihop Wireless Networks[C]//Proceedings of IEEE International Conference on Communications (ICC). IEEE Press, 2006:5576–5581.

[165] S. Roy, D. Saha, S. Bandyopadhyay, et al. Improving End-to-end Delay Through Load Balancing with Multipath Routing in Ad Hoc Wireless Networks Using Directional Antenna[C]//Proc. IWDC 2003: 5th International Workshop, LNCS v2918. 2003:225–234.

[166] S. Roy, S. Bandyopadhyay, T. Ueda, et al. Multipath Routing in Ad Hoc Wireless Networks with Omni Directional and Directional Antenna: A Comparative Study[C]//Proceedings of the 4th International Workshop on Distributed Computing, Mobile and Wireless Computing (IWDC). London, UK, UK: Springer-Verlag, 2002:184–191.

[167] D. Saha, S. Roy, S. B. An Adaptive Framework for Multipath Routing via Maximally Zone-disjoint Shortest Paths in Ad Hoc Wireless Networks with Directional Antenna[C]//Proceedings of IEEE Global Telecommunications Conference. 2003:226–230.

[168] J.-Y. Teo, Y. Ha, C.-K. Tham. Interference-minimized Multipath Routing with Congestion Control in Wireless Sensor Network for High-rate Streaming[J]. IEEE Transactions on Mobile Computing, 2008, 7(9):1124–1137.

[169] B. Fu, R. F. Li, X. Xiao, et al. Non-interfering Multipath Geographic Routing for Wireless Multimedia Sensor Networks[C]//Proceedings of International Conference on Multimedia Information Networking and Security (MINES). 2009, 1:254–258.

[170] T. V. Adam, A. Dunkels, T. Braun. On-demand Construction of Non-interfering Multiple Paths in Wireless Sensor Networks[C]//Paths in Wireless Sensor Networks 2nd Workshop on Sensor Networks (Informatik). 2005.

[171] Y. Kobayashi. Induced Disjoint Paths Problem in a Planar Digraph[J]. Discrete Applied Mathematics, 2009, 157:3231–3238.

[172] M. Albano, S. Chessa, F. Nidito, et al. Q-night: Adding Qos to Data Centric Storage in Non-uniform Sensor Networks[C]//International Conference on Mobile Data Management. 2007:166–173.

[173] J. Edmonds. Paths, Trees and Flowers[J]. Canadian Journal of mathematics, 1965, 17:449–467.

攻读博士学位期间发表的论文及其他成果

发表的学术论文

[1] Kejia Zhang, Hong Gao, Jianzhong Li. Finding Multiple Induced Disjoint Paths in General Graphs[J]. Information Processing Letters, 2011, 111(20): 1022-1026.（SCI 收录，IDS号为 831ST，IF=0.774）

[2] 张可佳, 高宏. 在传感器网络中寻找不相交路径的增量算法[J]. 通信学报, 2010, 31(9A): 59-67.（EI 收录号：20104613391340）

[3] Kejia Zhang, Shengfei Shi, Hong Gao, Jianzhong Li. Unsupervised Outlier Detection in Sensor Networks Using Aggregation Tree[C]. 3rd International Conference on Advanced Data Mining and Applications, Harbin, 2007: 158-169.（EI 收录号：20080311023327）

[4] Kejia Zhang, Hong Gao. Finding Multiple Length-Bounded Disjoint Paths in Wireless Sensor Networks[J]. Wireless Sensor Network, 2011, 3(12): 384-390.

[5] Kejia Zhang，Hong Gao. Efficient Distributed Algorithm for Correctly Finding Disjoint Paths in Wireless Sensor Networks[J]. International Journal of Sensor Networks.（已录用）

哈尔滨工业大学学位论文原创性声明及使用授权说明

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《无线传感器网络中多路径路由的研究》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果。据本人所知，论文中除已注明部分外不包含他人已发表或撰写过的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。本声明的法律结果将完全由本人承担。

作者签名：  日期：　　年　　月　　日

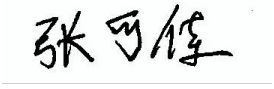
学位论文使用授权说明

本人完全了解哈尔滨工业大学关于保存、使用学位论文的规定，即：

（1）已获学位的研究生必须按学校规定提交学位论文；（2）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（3）为教学和科研目的，学校可以将学位论文作为资料在图书馆及校园网上提供目录检索与阅览服务；（4）根据相关要求，向国家图书馆报送学位论文。

保密论文在解密后遵守此规定。

本人保证遵守上述规定。

作者签名：  日期：　　年　　月　　日

导师签名：  日期：　　年　　月　　日

致 谢

衷心感谢导师 高宏 教授对我的精心指导。她的理解和关怀将会永远被我铭记在心。感谢李建中教授对我的指导和教诲。他对待科学的态度值得我永远的学习。感谢姜守旭教授对我的指导和帮助。

感谢实验室的洛吉洲老师、石胜飞老师、张炜老师、王宏志老师、邹兆年老师，以及实验室同窗们的热情帮助和支持!

…

个人简历

1981 年 08 月 11 日出生于 黑龙江省哈尔滨市。

2000 年 09 月考入 北京师范 大学 管理 学院（系）信息管理与信息系统 专业，2004 年 07 月本科毕业并获得 管理 学学士学位。

2005 年 09 月——2007 年 07 月在 哈尔滨工业 大学 计算机科学与技术 学院（系）计算机软件与理论 学科学习并获得 工 学硕士学位。

2007 年 09 月——至今在 哈尔滨工业 大学 计算机科学与技术 学院（系）计算机软件与理论 学科攻读博士学位。