

硕 士 学 位 论 文

**(2 0 1 8** 届**)**

广义超立方体的条件连通度及容错路由研 究

**Research on Conditional Connectivity and Fault-Tolerant Routing of Generalized Hypercubes**

## 研 究 生 姓 名 郭莉莉

指导教师姓名 樊建席，林政宽

专 业 名 称 计算机技术

研 究 方 向 并行与分布式系统

论文提交日期 2018 年 5 月

苏苏州州州大大大学学学学学学位位位论论论文文文独独独创创创性性性声声声明明

### 本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进 行研究工作所取得的成果。 除文中已经注明引用的内容外，本论文不含 其他个人或集体已经发表或撰写过的研究成果，也不含为获得苏州大学 或其它教育机构的学位证书而使用过的材料。 对本文的研究作出重要贡 献的个人和集体，均已在文中以明确方式标明。 本人承担本声明的法律 责任。

论文作者签名： 日 期：

苏苏州州州大大大学学学学学学位位位论论论文文文使使使用用用授授授权权权声声声明明

本人完全了解苏州大学关于收集、 保存和使用学位论文的规定，即：学位论文 著作权归属苏州大学。 本学位论文电子文档的内容和纸质论文的内容相一致。 苏州 大学有权向国家图书馆、 中国社科院文献信息情报中心、 中国科学技术信息研究所

（含万方数据电子出版社）、 中国学术期刊（光盘版）电子杂志社送交本学位论文的 复印件和电子文档，允许论文被查阅和借阅，可以采用影印、 缩印或其他复制手段 保存和汇编学位论文，可以将学位论文的全部或部分内容编入有关数据库进行检 索。

### 涉密论文口

本学位论文属 在 年 月解密后使用本规定。 非涉密论文口

论文作者签名： 日 期：

导 师 签 名： 日 期：

# 广义超立方体的条件连通度及容错路由研究

## 摘 要

随着信息化社会的飞速发展, 高性能计算已经成为继理论科学和实验科学之后科 学研究的第三大支柱。 从战略高度方面讲, 高性能计算技术是一个国家综合国力的表 现，在国防安全、 高科技发展和国民经济建设等各个领域都占有不可或缺的重要地 位。

在高性能计算的研究中，提升运行效率一直是发展的首要目标。 其中并行计 算(Parallel Computing)是提高计算机系统计算速度和处理能力的一种有效手段，它指 的是同时使用多种计算资源共同解决计算问题的方法，具体来讲，并行计算系统通 过多个处理单元之间的通信和协作来完成计算任务。 并行计算系统中的处理单元之 间的连接方式可视为一个网络，我们将该连接方式称为互连网络。

互 连 网 络 一 般 可 以 抽 象 为 一 个 简 单 图*G* = (*V*(*G*), *E*(*G*))， 其 中*G*中 顶 点 集 合*V*(*G*)表示互连网络中的处理器集合，*G*中边集合*E*(*G*)表示处理器之间的链路 集合。

随着系统计算需求的不断增加，互连网络规模不断扩大，网络中处理器数目逐 渐增多，继而导致处理器发生故障的概率增加。 当网络中处理器发生故障时，网络 能否继续保持正常运作取决于该网络的容错性，它是衡量互连网络优劣的一项关键 指标。 其中，连通度是衡量网络容错性的一个重要参数。 然而在实际情况中，条件 连通度中的限制连通度和额外连通度相较于传统的连通度能够更准确地衡量一个网 络的容错性。 同时，在某些处理器发生故障后仍然保持连通的网络中如何进行处理 器之间的路由是网络的容错性需要研究的重要课题。

广义超立方体是多处理器系统中一种常用的互连网络，具有正则性、 对称性、 良好的嵌入性和可扩展性等许多优良特性。 目前，广义超立方体已经被应用在多种 大型多处理器并行系统、数据中心网络和光纤通讯网络的构建中。

本文研究了广义超立方体的限制连通度和额外连通度，并给出相应条件下的容 错路由算法。 主要研究内容如下：

(1) 基于限制连通度的定义，本文给出了*r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)的1- 限 制 连 通 度， 并 给 出 了 其 详 细 证 明。 同 时， 本 文 给 出 了*G*(*mr*, *mr*−1, . . . , *m*1)中 的 总 故 障 顶 点 个 数 小 于1-限 制 连 通 度， 且 满 足1-限 制 连 通 度 条 件 的 容 错 路 由 算

法，该算法的时间复杂度为*O*(κ(*G*(*mr*, *mr* 1, . . . , *m* ))3)，其中κ(*G*(*m* , *m* , . . . , *m* ))表

− 1 *r r*−1 1

I

示*G*(*mr*, *mr*−1, . . . , *m*1)的连通度。

(2) 基于额外连通度的定义，本文给出了*r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)

的1-额外连通度和2-额外连通度，并给出了其详细证明。 同时，本文给出了*G*(*mr*,

*mr*−1, . . . , *m*1)中的总故障顶点个数小于2-额外连通度，且满足2-额外连通度条件 的容错路由算法，该算法的时间复杂度为*O*(*N* × κ(*G*(*mr*, *mr*−1, . . . , *m*1)))，其中*N*表

示*G*(*mr*, *mr*−1, . . . , *m*1)中的总顶点个数。

关键词： 高性能计算，互连网络，广义超立方体，限制连通度，额外连通度，容错 路由算法

作 者： 郭莉莉

指导教师： 樊建席，林政宽

II

# Research on Conditional Connectivity and Fault-Tolerant Routing of Generalized Hypercubes

## Abstract

With the rapid development of information society, high performance computing has been the third pillar of scientific research following theoretical sciences and experimental sciences. Speaking strategically, high performance computing technology is a manifestation of a country’s comprehensive national strength, and plays an indispensable role in the field of national defense security, high-tech development and national economic construction.

In the research of high performance computing, improving operation efficiency has al- ways been the primary goal of its development. Parallel computing is an effective means to improve the speed and computing capacity of the systems. It refers to the method to solve the calculation problem jointly with multiple processing units. Specifically, the parallel computing system completes the computing task through communication and cooperation among a plurality of processing units. In a parallel computing system, the connection be- tween processing units can be viewed as a network, which is named as the interconnection network.

Interconnection network can be abstracted as a simple graph *G* = (*V*(*G*), *E*(*G*)), where *V*(*G*) is the set of vertices in *G*, representing the set of processors in the interconnection network, and *E*(*G*) is the set of edges in *G*, representing the set of links between the various processors.

With the increasing demand of computing, the scale of the interconnection network is continuously expanding, and then the number of processors is gradually increasing, which in turn leads to an increase in the probability of processor failure. Whether the network with faulty processors can continue to operate normally or not depends on the fault-tolerance of this network. It is a key index to measure the performance of the interconnection net- work. And connectivity is an important parameter to measure the fault-tolerance of a net- work. However, restricted-connectivity and extra-connectivity of conditional connectivity can more accurately measure the fault-tolerance than traditional connectivity in actual sit-

remains connected with some faulty processors is a problem that needs to be considered when studying the fault-tolerance of this network.

Generalized hypercubes is a common interconnection network in multiprocessor sys- tem, which has many excellent properties such as regularity, symmetry, good embeddability and extensibility. At present, the generalized hypercubes has been used in many large multi- processor parallel systems, the structure of data center networks and optical fiber communi- cation networks.

In this paper, the restricted-connectivity and extra-connectivity of the *r*-dimensional generalized hypercubes *G*(*mr*, *mr*−1, . . . , *m*1) are studied, and two fault-tolerant routing algo- rithm are given. The major contributions are listed as follows.

1. Based on the definition of restricted connectivity, this paper gives the 1-restricted

connectivity of *G*(*mr*, *mr*−1, . . . , *m*1), and its detailed proof is given. At the same time, this paper presents a fault-tolerant routing algorithm in *G*(*mr*, *mr*−1, . . . , *m*1) with faulty vertices

under the conditions that the number of faulty vertices is less than 1-restricted connectivity and the every fault-free vertices has at least one fault-free neighbor. The time complexity of this algorithm is *O*(κ(*G*(*mr*, *mr* 1, . . . , *m* ))3), where κ(*G*(*m* , *m* , . . . , *m* )) represents the

− 1 *r r*−1 1

restricted-connectivity of *G*(*mr*, *mr*−1, . . . , *m*1).

1. Based on the definition of external connectivity, this paper presents the 1-extra con-

nectivity and 2-extra connectivity of *G*(*mr*, *mr*−1, . . . , *m*1), and the detailed proof are given. At the same time, this paper presents a fault-tolerant routing algorithm in *G*(*mr*, *mr*−1, . . . ,

*m*1) with fault vertices under the conditions that the number of faulty vertices is less than 2- extra connectivity and each component has at least 3 fault-free vertices. the time complexity

of the algorithm is *O*(*N* × κ(*G*(*mr*, *mr* 1, . . . , *m* ))3), where *N* presents the number of vertices

− 1

in *G*(*mr*, *mr*−1, . . . , *m*1).

**Keywords:** High performance computing, Interconnection network, Generalized hyper- cubes, Restricted connectivity, Extra-connectivity, Fault-tolerant routing algorithm

**Written by** Lili Guo

**Supervised by** Janxi Fan, Cheng-Kuan Lin

|  |  |  |
| --- | --- | --- |
| [第一章](#_bookmark0) | 目 录  [绪论](#_bookmark0)· · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | **1** |
| [1.1](#_bookmark1) | [研究背景](#_bookmark1) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 1 |
|  | [1.1.1 并行计算](#_bookmark2) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 2 |
|  | [1.1.2 互连网络](#_bookmark3) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 2 |
| [1.2](#_bookmark4) | [研究现状及研究意义](#_bookmark4) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 3 |
|  | [1.2.1 条件连通度](#_bookmark5) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 3 |
|  | [1.2.2 容错路由](#_bookmark6) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 5 |
| [1.3](#_bookmark7) | [研究内容](#_bookmark7) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 6 |
| [1.4](#_bookmark8) | [文章组织结构](#_bookmark8) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 6 |
| [第二章](#_bookmark9) | [相关知识](#_bookmark9) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | **8** |
| [2.1](#_bookmark10) | [基本定义和符号表示](#_bookmark10) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 8 |
| [2.2](#_bookmark11) | [广义超立方体及其性质](#_bookmark11) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 10 |
|  | [2.2.1 广义超立方体的定义](#_bookmark12) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 10 |
|  | [2.2.2 广义超立方体的性质](#_bookmark16) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 11 |
| [2.3](#_bookmark17) | [本章小结](#_bookmark17) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 12 |
| [第三章](#_bookmark18) | [广义超立方体的限制连通度及容错路由](#_bookmark18)· · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | **13** |
| [3.1](#_bookmark19) | [广义超立方体的1-限制连通度](#_bookmark19)· · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 13 |
| [3.2](#_bookmark28) | [1-限制连通度条件下的容错路由](#_bookmark28) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 20 |
|  | [3.2.1 1-限制连通度条件下的路由算法](#_bookmark29)· · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 20 |
|  | [3.2.2 容错路由算法分析](#_bookmark31) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 23 |
|  | [3.2.3 模拟实验](#_bookmark32) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 25 |
| [3.3](#_bookmark39) | [本章小结](#_bookmark39) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 29 |
| [第四章](#_bookmark40) | [广义超立方体的额外连通度及容错路由](#_bookmark40)· · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | **30** |
| [4.1](#_bookmark41) | [广义超立方体的1-额外连通度和2-额外连通度](#_bookmark41) · · · · · · · · · · · · · · · · · · · · · · · · · · · | 30 |
|  | [4.1.1 广义超立方体的1-额外连通度](#_bookmark42) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 30 |
|  | [4.1.2 广义超立方体上2-额外连通度下的基本性质](#_bookmark43)· · · · · · · · · · · · · · · · · · · · · · | 30 |
|  | [4.1.3 广义超立方体的2-额外连通度](#_bookmark52) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 37 |
| [4.2](#_bookmark61) | [广义超立方体的2-额外连通度下的容错路由](#_bookmark61) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 44 |
|  | [4.2.1 2-额外连通度条件下的容错路由算法](#_bookmark62) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 44 |

[4.2.2 容错路由算法分析](#_bookmark63) 46

[4.2.3 模拟实验](#_bookmark64) 49

[4.3 本章小结](#_bookmark71) 52

[第五章 总结与展望](#_bookmark72) **54**

[5.1 总结](#_bookmark73) 54

[5.2 展望](#_bookmark74) 55

[参考文献](#_bookmark75) 56

[攻读硕士学位期间发表的论文和参与的科研项目](#_bookmark129) **61**

[致谢](#_bookmark130) 62

# 第一章 绪论

### 1.1 研究背景

高性能计算(High Performance Computing)是计算机科学的一个分支，它是继理 论科学和实验科学之后的人类科学研究的第三大科学，致力于开发高性能计算机和 运行在高性能计算机上的应用软件 [[1](#_bookmark76)]。 高性能计算利用大量处理单元的聚合计算能 力在很短的时间周期内完成复杂的计算任务，更直观地讲，它是指快速、 量大并且 性能高的一类计算，典型的有向量计算、并行与分布式计算和网格计算等 [[2](#_bookmark77)]。

高性能计算作为世界高技术领域的战略制高点，已经成为科技进步的重要标志 之一，同时也是一个国家科技综合实力的集中体现 [[2](#_bookmark77)]，是支撑国家实力持续发展的 关键技术之一。 现在高性能计算已经渗透到传统应用和新兴应用的各个学科领域。 在传统应用领域中，高性能计算已经被广泛应用在高能物理研究、 核武器设计、 航 天航空飞行器设计、 能源勘探、 中长期天气预报、 卫星图像处理、 互联网服务和工 业仿真等领域，对国民经济发展和国防建设具有重要的价值 [[1](#_bookmark76)]。 在新兴应用领域中， 高性能计算在生命科学、 网络社交、 人工智能、 大数据处理和云计算等方面都有广 泛的应用 [[1](#_bookmark76), [2](#_bookmark77)]。 微软公司的Dan Reed认为，高性能计算和云计算是出生时被分开的 双胞胎，并且他认为高性能并行计算技术是隐藏在云计算背后的核心技术 [[3](#_bookmark78)]。 人工 智能和机器学习的顶级学者Andrew Ng和分布式系统顶级专家Jeff Dean所共同执行 的Google Brain项目，用了包含16000个CPU核的高性能并行计算平台训练超过10亿 个神经元的深度神经网络，在语音识别和图像识别等领域取得了突破性的进展 [[1](#_bookmark76)]。 目前，在科研、 国防、 工业、 金融、 服务和生活等领域中，高性能计算都发挥着重 要的作用。

根据Top 500历史数据进行预测，高性能计算机将在2020年左右进入E级时代 [[1](#_bookmark76)]。 在中国发布的《国家中长期科学和技术发展规划纲要(2006―2020年)》中，提出了要 全面提升我国的自主创新能力，明确指出要加速发展高性能计算, 并启动了“十三 五”期间E级计算机系统(计算峰值达到百亿亿次的高性能计算机)的研制计划，在能 耗、可靠性、应用效率等方面都将面临一系列挑战。

近年来，我国高性能计算机的发展备受世界瞩目，其中代表性的国产高性能计 算机有神威蓝光、 天河一号、 天河二号、 联想6800 和曙光600(星云)等 [[2](#_bookmark77)]。 其中，天

河一号和曙光星云系统在2010年先后获得了世界高性能计算机Top500榜单的第一名 和第二名。 自2013年起, 天河二号在世界超级计算机排行榜TOP500中连续七次占据 第一，在国际中引起了高度的关注 [[4](#_bookmark79)]。 在2017年最新发布的第50届TOP500榜单中， 神威太湖之光和天河二号再次蝉联冠亚军，它们的浮点运算速度分别达到每秒9.3亿 亿次和每秒3.39亿亿次。

**1.1.1** 并并行行行计计计算算

在高性能计算的研究中，提升运算效率一直是其发展的首要目标。 基于高性能 计算机的速度峰值性能不等同于单个应用软件运行时的实际性能，它们之间往往差 距较大。 要发挥高性能计算机的高速硬件优势，必须要有适用的算法和调优的应用 程序来实现数百万核之间的并行运算。 因此，创建和应用并行计算成为解决多处理 单元速度瓶颈的最好方法之一。

并行计算(Parallel Computing)指的是同时使用多种计算资源共同解决计算问题的 方法，是提高计算机系统计算速度和处理能力的一种有效手段。 具体来讲，它的基 本思想是将大规模的计算任务分解成若干个部分，各部分均由不同的处理单元来 进行计算，然后通过多个处理单元之间的通信和协作来完成计算任务，进而有效地 提升整个系统的运算速度和运算能力 [[2](#_bookmark77)]。 并行计算由以下几个部分构成：并行计算 机(并行计算的硬件平台)、 并行算法(并行计算的理论基础)、 并行程序设计(并行计 算的软件支撑)和并行应用(并行计算的发展动力) [[5](#_bookmark80)]。

并行计算系统中的处理单元按照某种特定的方式进行连接，处理单元之间的连 接方式可视为一个网络，我们将该连接方式称为互连网络。 直观地讲，互连网络的 拓扑结构在很大程度上影响着并行计算机系统的带宽、 延迟、 可靠性等通信性能， 进而直接影响其运算能力 [[6](#_bookmark81)]。 因此，与互连网络拓扑结构的相关研究成为研究并行 计算机系统的一个重要课题。

**1.1.2** 互互连连连网网网络络

根据互连网络的拓扑结构在程序运行过程中是否可变，互连网络分为静态 网络(Static Network)和动态网络(Dynamic Network) [[2](#_bookmark77), [7](#_bookmark82)]。 其中静态网络是指网络中 顶点之间存在固定的物理连接，且在程序的执行过程中顶点之间的连接方式不 变。常见的静态网络有广义超立方体(Generalized Hypercube) [[8](#_bookmark83)–[10](#_bookmark85)]、网格(Mesh) [[11](#_bookmark86)]、环

绕(Torus) [[12](#_bookmark87)]、 星图(Star) [[13](#_bookmark88)]、 超立方体(Hypercube) [[14](#_bookmark89)]、 交叉立方体(Crossed Cube) [[15](#_bookmark90), [16](#_bookmark91)] 和莫比乌斯立方体(Mo¨bius Cube) [[17](#_bookmark92), [18](#_bookmark93)]等。 另外，动态网络是指顶点之间没有固定的 物理连接，而是在连接路径的交叉点处用电子开关、 路由器或仲裁器等提供动态连 接。 典型的动态网络有多级互连网络(Multistage Interconnection Network) [[19](#_bookmark94)]、 交叉开 关(Cross-Switching) [[20](#_bookmark95)]、总线(Bus) [[21](#_bookmark96)]等。

传统静态网络中的超立方体是很早就被提出并展开研究的一类网络。 本文主要 研究静态网络中超立方体的一个主要的变型网络结构：广义超立方体。 它是目前 重要的、 有吸引力的网络之一，具有良好的正则性、 低直径、 对称性、 可嵌入性、 良好的通信性能 [[22](#_bookmark97)–[24](#_bookmark98)] 和可扩展性等优点 [[9](#_bookmark84)]。 目前，它不仅是某些大型多处理器并行 计算机系统所采用的拓扑结构 [[25](#_bookmark99), [26](#_bookmark100)]，而且还被用于光纤通讯网络(Optical Fibers) [[27](#_bookmark101)]的 构建中。 同时，广义超立方体还被用于多种数据中心网络的构建中，如BCube [[28](#_bookmark102)]、 SWCube [[29](#_bookmark103), [30](#_bookmark104)]、FBFLY(Flattened Butterfly) [[31](#_bookmark105)] 和HyperX [[32](#_bookmark106)]。

### 1.2 研究现状及研究意义

在实际应用中，一个大型的高性能互连网络具有大量强大计算能力的处理单元， 要求该系统在很短的时间周期内完成给定的计算任务。 实际应用中一般通过不断增 加处理器数目来提升整个系统的计算能力(Capability)，即计算速度 [[2](#_bookmark77)]。 随着网络中处 理器数目逐渐增多，处理器发生故障的概率逐渐增加。 当网络中处理器发生故障时， 网络中某些特性能否继续保持取决于网络的容错性，它是衡量互连网络优劣的一项 关键指标。 近年来，互连网络的容错问题一直是学者们研究的热点 [[33](#_bookmark107)–[35](#_bookmark108)]。 其中，顶 点连通度是衡量网络容错性的一个重要参数。 而且，在某些处理器发生故障后仍然 保持连通的网络中，无故障的处理器之间如何进行路由是研究该网络的容错性需要 考虑的一个重要问题，该问题被称为容错路由问题 [[6](#_bookmark81)]。 接下来，我们分别介绍连通 度和容错路由两个问题相关的研究现状。

**1.2.1** 条条件件件连连连通通通度度

互连网络中的顶点连通度越大，表示该网络的容错性越 好。 我们以*r*-维广

义 超 立 方 体*G*(*mr*, *mr*−1, . . . , *m*1)为 例， 它 的 顶 点 连 通 度 为∑*i*=1(*mi* − 1)。 也 就 是 说， 在*G*(*mr*, *mr*−1, . . . , *m*1)中，当总故障顶点数小于∑*i*=1(*mi* − 1)时，该网络仍是连通的。然

*r*

*r*

而，仅仅用顶点连通度来衡量一个网络的容错性仍存在不足之处。 对于一些常见的

网络结构，它们的顶点连通度至多为该网络中所有顶点的邻接点个数的最小值。 我

们可以计算得到当*G*(*mr*, *mr* 1, . . . , *m* )的总故障顶点数为∑*r* (*mi* − 1)时，某个顶点的

− 1 *i*=1

∏*r*

所有邻接点同时故障的概率为*P* = *i*=1 *mi* 。 在这种情况下，某个顶点的所有邻接点

∏*r*

(∑*r*

*i*=1 *mi* )

*i*=1(*mi*−1)

同时故障的概率*P*是极小的，特别是当*G*(*mr*, *mr*−1, . . . , *m*1)的规模足够大时，这种情况

发生的概率趋近于0。 因此，传统的顶点连通度无法很好地衡量实际的互连网络容错

性的优劣。 为了解决这个问题，Harary [[36](#_bookmark109)]提出了条件连通度的概念。 令图*G*是一个连 通的无向图，ρ表示一个图的某种性质，条件连通度指的是使得*G*变得不连通且剩余 的分支仍然保留着性质ρ时，所需去掉的最少顶点数。 在此基础上，Esfahanian [[37](#_bookmark110)]进 一步引入了限制连通度的概念，限制连通度要求当某些顶点发生故障时，每个顶点 至少要有一个无故障的邻接点。 之后，Latifi等人 [[38](#_bookmark111)]在限制连通度的基础上，进一步 提出了*g*-限制连通度，即要求当发生顶点故障时，每个顶点至少有*g*个无故障邻居。

近年来，*g*-限制连通度已成为衡量网络容错性的重要标准之一，在超立方体及 其部分变型立方体中，已取得了许多相关的研究成果。 1998年，Wu和Guo研究了超 立方体上的*g*-限制连通度 [[39](#_bookmark112)]；2013年，Li和Xu研究了交换超立方体上的*g*-限制点连 通度和*g*-限制边连通度 [[40](#_bookmark113)]；2016年，Ye和Liang进一步研究了类超立方体上的*g*-限制 连通度，得到了交叉立方体、莫比乌斯立方体、局部扭立方体(Local Twisted Cube)等 超立方体的变型立方体的*g*-限制连通度 [[41](#_bookmark114)]；2017年，Hsieh等人研究了局部扭立方体 上的2-限制连通度和3-限制连通度 [[42](#_bookmark115)]。

另外，Fa`brega和Fiol在限制连通度的基础上定义了*g*-额外连通度 [[43](#_bookmark116)]。 要求除去 网络中的故障顶点之后，剩余的每个分支至少包含*g* + 1个无故障顶点。 目前，*g*-额 外连通度在超立方体及其部分变型立方体中，已得到许多相关的研究成果。 2009年， Yang和Meng研 究了超立方体上的*g*-额外连通度 [[44](#_bookmark117)]；2013年，Chang和Hsieh将*g*-额 外连通度应用到类超立方体中，研究了它们的2-额外连通度和3-额外连通度 [[45](#_bookmark118)]； 2013年，Li和Yang研究了超立方体上的*g*-外边连通度 [[46](#_bookmark119)]；2014年，Yang和Lin进一步 研究了类超立方体上的*g*-额外连通度，得到了包括交叉立方体、 莫比乌斯立方体、 局部扭立方体等超立方体的变型立方体的*g*-额外连通度 [[47](#_bookmark120)]。

广义超立方体作为超立方体的一个重要变型，具有更好的容错性和可扩展性。 在1984年Bhuyan和Agrawal定义了广义超立方体 [[8](#_bookmark83)]。 之后，在1996年Dyi-Rong等人研 究了广义超立方体传统连通度 [[9](#_bookmark84)]。 广义超立方体的限制连通度和额外连通度的相关

研究却仍有不足。 因此，我们有必要对这两个参数进行研究。

**1.2.2** 容容错错错路路路由由

根据网络中故障信息的情况，容错路由算法可以分为三类：基于全局信息的算 法、 基于局部信息的算法以及介于两者之间的算法。 基于全局信息的算法，又称为 非分布式(Non-distributed)算法或离线(Off-line)算法，这类算法要求网络中每个顶点 了解网络中所有的故障信息，一般可以得到源顶点和目的顶点之间的最短通信路 径，但需要使用大量的存储空间以及通信开销。 Gu和Peng人提出的算法是其典型代 表 [[48](#_bookmark121)]。 基于局部信息的算法，又称分布式(Distributed)算法，这类算法要求每个顶点 了解其邻接点的故障信息，因此只需要较少的存储空间和通信开销，相应的，它不 能保证得到的通信路径是最优的。 Li等人提出的算法就属于这一类 [[49](#_bookmark122)]。 还有一些算 法介于两者之间，即每个顶点只了解自己附近顶点的故障信息，这类算法需要的存 储空间和通信开销也介于上述两类算法之间。 其典型的算法是Wu提出的算法 [[50](#_bookmark123), [51](#_bookmark124)]。

很早就有学者开始研究广义超立方体上的容错路由。 在1998年，董明生等人就 研究了广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)上的容错路由算法，在该网络中，对于任意

一个*i* ∈ ⟨*r*⟩ \ {0}，都有*mi* = *n*。 对*G*(*mr*, *mr*−1, . . . , *m*1)中给定的任意两个顶点，用*h*表示

这两个顶点的汉明距离。董明生等人提出的算法在故障顶点的个数小于*mr* − *r*时，可

以构造出连接这两个顶点的长度不大于*h* + 2的无故障路径；在故障顶点个数大于等

于*mr* − *r*且小于*m*2*r* − *m*2 − *mh* + *m*时，可以构造出长度不大于*h* + 4*m* − 2的无故障路 径 [[52](#_bookmark125)]。 该算法在故障顶点较少时表现良好；当故障顶点较多时，获取的路径长度较

长。 之后，在2006年，刘红美基于广义超立方体的局部连通性定义，研究了故障顶 点分布比较均匀且具有局部连通性的广义超立方体中的单播容错路由算法 [[53](#_bookmark126)]。 继 而，在2009年，针对*r*-维BCube数据中心网络(该网络的逻辑结构即为广义超立方体)， Guo等人提出的容错路由算法BuildPathSet [[28](#_bookmark102)]在网络中的总故障顶点数少于该网络的 连通度κ(*G*)时，可对任意两个无故障顶点构建κ(*G*)条无故障路径，这些路径的长度 最长为*h* + 2且该算法的时间复杂度为*O*(κ(*G*)2)。

综上所述，目前尚缺乏广义超立方体在限制连通度与额外连通度条件下的容错 路由的研究成果。 因此，我们将针对这个问题开展研究。

1.3 研究内容 本文的研究内容分为以下两个部分： 一、广义超立方体的限制连通度及容错路由问题

在第三章中，基于1-限制连通度的定义，我们求出了广义超立方体的1-限制连 通度，并给出其相关证明。 同时，我们给出了当广义超立方体的总故障顶点数小于 其1-限制连通度时，在每个无故障顶点都至少有一个无故障邻居条件下的容错路由 算法。 本文取得如下研究成果：

1. *r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)的1-限制连通度是2κ(*G*(*mr*, *mr*−1, . . . , *m*1)) − *n*，其中*n* = max{*m*1, *m*2, . . . , *mr*}。

2. 给出一个满足1-限制连通度条件的容错路由算法，该算法的时间复杂度 为*O*(κ(*G*(*mr*, *mr* 1, . . . , *m* ))3)。

− 1

二、广义超立方体的额外连通度及容错路由问题 在第四章中，基于额外连通度的定义，我们求出了广义超立方体的1-额外连通

度和2-额外连通度，并给出其相关证明。 同时，我们给出当广义超立方体的总故障

顶点数小于其2-额外连通度时，在除去故障顶点集合后的每个分支中至少包含3个无 故障顶点条件下的容错路由算法。 本文取得如下研究成果：

1. *r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)的1-额外连通度是2κ(*G*(*mr*, *mr*−1, . . . , *m*1)) − *n*，其中*n* = max{*m*1, *m*2, . . . , *mr*}。

2. 对 于 广 义 超 立 方 体*G*(*mr*, *mr*−1, . . . , *m*1)， 其 中*r* ≥ 5且 对 于 任 意 的*i* ∈ ⟨*r*⟩ \

{0} 都 有*mi* ≥ 3 且*mi* ≥ *mi*−1。 若*mr* > *mr*−1， 则*G*(*mr*, *mr*−1, . . . , *m*1)的2-额 外 连 通 度 为3κ(*G*(*mr*, *mr*−1, . . . , *m*1)) − 2*mr*；若*mr* = *mr*−1，则*G*(*mr*, *mr*−1, . . . , *m*1)的2-额外连通度 为3κ(*G*(*mr*, *mr*−1 , . . . , *m*1)) −2*mr* − 1。

3. 给出一个满足2-额外连通度条件的容错路由算法，该算法的时间复杂度

为*O*(*N* × κ(*G*(*mr*, *mr*−1, . . . , *m*1)))，其中*N*为*G*(*mr*, *mr*−1, . . . , *m*1)中的总顶点个数。

### 1.4 文章组织结构

全文的组织结构如下： 第一章介绍了本文的研究背景、研究意义、研究内容以及文章的组织结构。

第二章首先介绍了与本文相关的图论中的基本定义和符号表示，然后介绍了广 义超立方体的定义以及一些主要的性质。

第三章给出广义超体方体的1-限制连通度，并且给出广义超立方体中满足1-限 制连通度条件下的容错路由算法。

第四章给出广义超体方体的1-额外连通度和2-额外连通度，并且给出广义超立 方体中满足2-额外连通度条件下的容错路由算法。

第五章对本论文所做的工作进行了总结，并对未来的研究工作进行了展望。

# 第二章 相关知识

本章首先介绍本文所需要用到的一些图论的基本定义和符号表示，然后介绍广 义超立方体的定义及其主要性质。

### 2.1 基本定义和符号表示

互连网络的拓扑结构可以抽象成图来表示。 图中的顶点表示互连网络中的处理 器，图中的边表示处理器之间的链路。 因此，研究互连网络拓扑结构的性质问题可 以归结为研究图结构及其性质的问题。 因为图结构可以看作是互连网络结构的数学 模型，所以对应的图结构及其性质可以用来衡量互连网络的性能。 其中，图论是目 前研究网络最常使用的数学工具之一。

在不同的文献中，对图论中的一些定义的描述和符号的表示存在着些许差 异。 为了不产生混淆，本文中我们主要参考孙惠泉所著的“图论及其应用” [[54](#_bookmark127)]，以 及Hsu和Lin所著的“Graph Theory and Interconnection Networks” [[55](#_bookmark128)]，给出本文用到 的一些图论的基本定义和符号表示。

我们将图(Graph)*G*定义为一个有序的二元组*G*(*V*(*G*), *E*(*G*))，可以简写为*G*(*V*, *E*)。 其中，*V*(*G*)是*G*的非空顶点集合(Vertex Set)且*E*(*G*)是*G*的边集合(Edge Set)。 *V*(*G*)中

的元素称为*G*的顶点(Vertex)，*E*(*G*)中的元素称为*G*的边(Edge)，并且*E*(*G*) ⊆ *V*(*G*) ×

*V*(*G*)。 若*V*(*G*) × *V*(*G*)是无序顶点对集合，则称*G*为无向图。 若*G*中任意两个顶点之

间存在至多一条边且每个顶点都没有自己连接到自己的边，则称该图为简单图。

本文中所研究的图是无向简单图。 若*G*中仅有一个顶点，则称该图为平凡图(Trivial Graph)。

若两个顶点*u*和*v*在*G*中有边相连，则我们称这两个顶点是相邻的，并称*u*是*v*的

一个邻接点或邻居。 顶点*u*的所有邻居集合用*NG*(*u*) = {*v* | (*u*, *v*) ∈ *E*(*G*)且*v* ∈ *V*(*G*)}来 表示。 集合*NG*(*u*)的大小为顶点*u*的度(Degree)，用*degG*(*u*)表示。 *G*中所有顶点的度

的最大值和最小值分别记为∆(*G*)和δ(*G*)。 若*G*中每个顶点的度都为*k*，则称*G*为*k*-正 则图。 若*V*′是*G*中顶点集合*V*(*G*)的一个非空子集，则集合*V*′的邻居集合为*NG*(*V*′) = (∪*x*∈*V*′ *NG*(*x*)) \ *V*′。

令*F*为*G*的一个故障顶点集合，即*F* ⊂ *V*(*G*)。若顶点*u*满足*u* ∈ *F*，则称*u*是故障顶 点；否则，称*u*是无故障顶点。 若*x* ∈ *NG* (*u*)且*x* <! *F*, 则称*x*是*u*的一个无故障邻居。

对于两个图*G*1 = (*V*(*G*1), *E*(*G*1))和*G*2 = (*V*(*G*2), *E*(*G*2))，若*V*(*G*1) ⊆ *V*(*G*2)且*E*(*G*1) ⊆

*E*(*G*2)，则称*G*1是*G*2的子图。 设*V*′ 是*G*的顶点集合*V*的一个非空子集。 以*V*′ 作为顶点

集合并以*G*中两端点都在*V*′ 中的所有边作为边集合的子图，称为*G*中由*V*′ 所导出的子 图，记为*G*[*V*′ ]。

令*W* = (*v*1, *v*2, . . . , *vn*)为*G*中 的 一 个 非 空 有 限 的 顶 点 交 替 序 列， 且 对 任 意 一

个1 ≤ *j* < *n*， 都 有*vj*和*vj*+1 是 相 邻 的， 则 称*W*是 一 条 从*v*1到*vn*的 途 径(Walk)， 其 中*v*1和*vn*分 别 为 该 途 径 的 起 点 和 终 点。 途 径 中 所 经 过 的 边 的 数 目 称 为*W*的 长

度。 如果*W*中不存在重复的顶点，那么称该途径为一条从*u*1到*un*的路或路径。 同 时，我们也用(*u*1, *u*2, . . . , *ui*, *Q*, *uj*, . . . , *un*) 表示路径*P* = (*u*1, *u*2, . . . , *un*)，其中*Q*表示路

径(*ui*+1, *ui*+2, . . . , *uj* 1)。 令*P*−1表示路径(*u* , *u* , . . . , *u* , *u* )，即路径*P*中顶点的逆序排

− *n n*−1 2 1

列。 同时，我们使用*Path*(*P*, *u*1, *ui*) 来表示沿着*P*从*u*1到*ui*的路径(*u*1, *u*2, . . . , *ui*)。 简单

地，我们用*Pn*表示一个包含*n*个顶点的路径。 设*P*和*Q*是顶点*u*与*v*之间的两条不同的

路径，若*V*(*P*) ∩ *V*(*Q*) = {*u*, *v*}，则称*P*和*Q*是顶点不相交(Vertex-Disjoint)路径，简称不 相交路径或不交路。 另外，若*P*中不包含故障顶点，则我们称*P*为无故障路径。

*G*中 任 意 两 个 顶 点*u*和*v*之 间 所 有 路 径 的 长 度 的 最 小 值 称 为 它 们 之 间 的 距 离(Distance)，记作*dist*(*G*, *u*, *v*)，简写为*dist*(*u*, *v*)。 *G*的直径(Diameter)为*D*(*G*) = max

{*dist*(*G*, *u*, *v*) | *u*, *v* ∈ *V*(*G*)且*u* 芋 *v*}。 可以看出，直径是图中任何两个顶点之间的距离

的最大值，因此图的直径可以用来度量网络通信性能的最坏情形。

当*G*中存在一条以*u*为起点且以*v*为终点的路径(或途径)，称*G*中顶点*u*和*v*为连通 的(Connected)。 显然，两个顶点间的连通关系是*V*(*G*)上的等价关系(即满足自反性、 对称性及传递性的二元关系)。 它将*V*(*G*)划分成等价类*V*1, . . . , *V*ω，其中每个*Vi*中的任 意两个顶点*u*和*v*都连通，且不同*Vi*与*Vj*之间的任意两个顶点都不连通。 我们称每个

导出子图*G*[*Vi*] 为*G*的一个分支(Component)，其中1 ≤ *i* ≤ ω。 若*G*中任意两个顶点间

都有一条路，则称*G*是连通的，否则称*G*是不连通的(Disconnected)。 若*G*1是图*G*的一

个子图，*u* ∈ *G* − *V*(*G*1) 且*u*和子图*G*1中某个顶点之间有一条路，则称*u*和子图*G*1连 通。

*G*的顶点连通度(Vertex Connectivity)就是使*G*变成不连通的或平凡图，所需去 掉的最少的顶点数，用κ(*G*)表示。 顶点连通度简称为连通度。 *G*的*g*-限制连通度是 使*G*不连通，且使*G*中每个无故障顶点都至少有*g*个无故障邻居，所需要去掉的最少

顶点数，用κ*g*(*G*)表示。 另外，*G*的*g*-额外连通度是使该图不连通，且使除去故障顶 点后的每个分支中至少有*g* + 1个无故障顶点，所需要去掉的最少顶点数，用κ*g*(*G*)表 示。

对 于 两 个 图*G*1和*G*2， 若 存 在 两 个 一 一 映 射： Ψ : *V*(*G*1) → *V*(*G*2)以 及Φ :

*E*(*G*1) → *E*(*G*2)，使得对于任意*e* = (*u*, *v*) ∈ *E*(*G*1)，都有(Ψ(*u*), Ψ(*v*)) ∈ *E*(*H*)且Φ(*e*) =

(Ψ(*u*), Ψ(*v*))，则称*G*1和*G*2同构(Isomorphism)，记为*G*1 三 *G*2。

若*G*中的任意两个顶点相邻，我们称*G*为完全图(Complete Graph)。 我们把*n*个顶

点的完全图记为*Kn*。 此外，对于一个整数*r*，我们令⟨*r*⟩ = {0, 1, 2, ..., *r*}。

### 2.2 广义超立方体及其性质

广义超立方体具有良好的正则性、 低直径、 对称性、 可嵌入性和可扩展性等优 点。 一直以来广义超立方体都受到学者们的广泛关注和研究。 本小节中首先介绍广 义超立方体的定义，然后介绍其与本文相关的符号定义和表示。

**2.2.1** 广广义义义超超超立立立方方方体体体的的的定定定义义

*r*-维广义超立方体可以用如下方式来定义 [[8](#_bookmark83)]：

定义 **2.1.** 令*r* ≥ 1，且对于任意一个1 ≤ *i* ≤ *r*都有*mi* ≥ 2。*r*-维广义超立方体*G*(*mr*, *mr*−1,

. . . , *m*1)中的顶点可以用长度为*r*的字符串*urur*−1 . . . *u*1来表示，其中对于所有的1 ≤ *j* ≤

*r*，有*uj* ∈ ⟨*mi* − 1⟩。 图中的任两个顶点相邻当且仅当它们只有一位不同。

图 [2-1](#_bookmark14)是1-维广义超立方体*G*(2)和2-维广义超立方体*G*(3, 2)与*G*(2, 3)；图 [2-2](#_bookmark15)是3- 维 广 义 超 立 方 体*G*(4, 3, 2)。 根 据 定 义[2.1](#_bookmark13)可 知 任 意 调 整*mr*, *mr*−1, . . . , *m*1的 次 序 都 不 影 响 广 义 超 立 方 体 的 结 构。 例 如*G*(3, 2)同 构 于*G*(2, 3)(如 图[2-1](#_bookmark14))且*G*(4, 3, 2)同 构

于*G*(3, 4, 2)。

特别地，为了简化符号的表示，当确定叙述中所讨论的图为*G*(*mr*, *mr*−1, . . . , *m*1)

时，我们将用*G*来表示*G*(*mr*, *mr*−1, . . . , *m*1)。

接下来，我们给出广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)相关的定义和符号表示。 对于广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)，*mi*为*G*(*mr*, *mr*−1, . . . , *m*1)中第*i*维的点的个

数，我们称*mi*为第*i*维的基数(Radix)，其中*i* ∈ ⟨*r*⟩ \ {0}。

20 21

02 12

10

0 1

**G(2)** 00

11 01 11

01 00 10

**G(3,2)**

**G(2,3)**

图 2-1 广义超立方体*G*(2)，*G*(3, 2)和*G*(2, 3)

021 121

020 120

220

221

320

321

011

010

110

111

210

211

310

311

001 101

201

301

000 100 200 300

图 2-2 广义超立方体*G*(4, 3, 2)

令*u* = *urur*−1 . . . *u*1，*v* = *vrvr*−1 . . . *v*1 且(*u*, *v*) ∈ *E*(*G*(*mr*, *mr*−1, . . . , *m*1))。 若*uj* 芋 *v j*， 且对于任意一个*i* ∈ ⟨*r*⟩ \ { *j*, 0}都有*ui* = *vi*，我们称边(*u*, *v*)为第 *j*维的边。

在*G*(*mr*, *mr* 1, . . . , *m* )中，对于任意的一个 *j* ∈ ⟨*m* − 1⟩，我们用*Gi*[ *j*]表示由顶点集 合{*ur* . . . *ui*+1 *jui* 1 . . . *u u* | *u* ∈ ⟨*m* − 1⟩}所导出的子图，其中*i* ∈ ⟨*r*⟩ \ {0}。子图*Gr*[ *j*]中共

− 1 *i*

− 2 1 *i i*

有∏*r*

*i*=1 *mi*

*mj*

个顶点，且κ(*Gi*[ *j*]) = κ(*G*) −

*mi* + 1。此外，我们令子图*Gr*[*J*] 为∪

*j*∈*J*

*V*(*Gr*[ *j*])所

导出的子图，其中*J* ⊆ ⟨*mr* − 1⟩。 对于任意一个顶点集合*F* ⊂ *V*(*G*(*mr*, *mr*−1, . . . , *m*1))，

我们令*Fj* = *F* ∩ *V*(*Gi*[ *j*])，其中 *j* ∈ ⟨*mi* − 1⟩。 我们令*FJ* = ∪ *j J F* ，其中*J* ⊆ ⟨*m*

− 1⟩。

**2.2.2** 广广义义义超超超立立立方方方体体体的的的性性性质质

∈ *j* *i*

*r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)中与本文相关的主要性质如下

[[9](#_bookmark84)]：

(1)共有∏*r*

*i*=1

*mi*个顶点；

*r r*

(2)共有∏ 条边；

∑

*i*=1 *mi*×

*i*=1(*mi*−1)

2

(3)直径为*r*；

(4)是∑*r*

*i*=1

(*mi* − 1)−正则的；

(5)顶点连通度为∑*r*

*i*=1

(*mi* − 1)；

(6)任意两个不同的顶点之间存在∑*r*

*i*=1

(*mi* − 1)条不相交路。

### 2.3 本章小结

本章介绍了与本文密切相关的知识。 首先，我们介绍了本文用到的图论相关的 一些基本定义和符号表示。 然后，我们介绍了广义超立方体相关的定义和符号表示 以及它的一些主要性质。

# 第三章 广义超立方体的限制连通度及容错路由

连通度是衡量网络容错性的重要标准之一。 而限制条件连通度可以更好地衡量 一个网络的容错性。 本章中，首先给出广义超体方体的1-限制连通度，并给出其证 明，然后给出广义超立方体的1-限制连通度条件下的容错路由算法GHCFP。

本章研究了广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)的1-限制连通度，得出的主要结 论如下：(1)*r*−维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)的1- 限制连通度是2κ(*G*) − *n*，其 中*n* = max{*m*1, *m*2, . . . , *mr*}。 (2)给出一个1−限制连通度下的容错路由算法，其时间复

杂度为*O*(κ(*G*)3)。

### 3.1 广义超立方体的1-限制连通度

本节中，在给出广义超立方体的1-限制连通度及其具体证明之前，先给出证明 相关的引理[3.1](#_bookmark20)、[3.2](#_bookmark21)和定理[3.1](#_bookmark22)。

引理 **3.1.** 若*u*和*x*在*G*(*mr*, *mr*−1, . . . , *m*1)中相邻，则|*NG* (*u*) ∩ *NG*(*x*)| ≤ *n* − 2，其中*n* =

max{*m*1, *m*2, . . . , *mr*}。

证明**.** 令*u* = *urur*−1 . . . *u*1且*x* = *xr xr*−1 . . . *x*1。 根据定义[2.1](#_bookmark13)可知，存在唯一一个整数*i*使 得*ui* 芋 *xi*，其中*i* ∈ ⟨*r*⟩ \ {0}。 因此，|*NG*(*u*) ∩ *NG*(*x*) = {*yryr*−1 . . . *y*1 | *yi* ∈ ⟨*mi* − 1⟩ \ {*ui*, *xi*} 且对于所有的 *j* 芋 *i*，有*yj* = *uj* = *x j*}| = *mi* − 2 ≤ *n* − 2。 口

引理 **3.2.** 若*F*为*G*(*mr*, *mr*−1, . . . , *m*1)中的一个顶点集合，且对于⟨*mr* − 1⟩中的任意的一

−

个元素*i*，都有|*F* ∩ *V*(*G* − *V*(*Gr*[*i*]))| ≤ κ(*G*) − 2，则*G*(*mr*, *mr*

连通的。

1, . . . , *m*1) − (*V*(*Gr*[*i*]) ∪ *F*)是

证明**.** 因为当*mr* = 2时，*G*(*mr*, *mr* 1, . . . , *m* ) − *V*(*Gr*[*i*])和*G*(*m* , . . . , *m* )同构。 当*m* ≥

− 1 *r*−1 1 *r*

3时，*G*(*mr*, *mr* 1, . . . , *m* ) − *V*(*Gr*[*i*])和*G*(*m* − 1, *m* , . . . , *m* )同构。 根据定义[2.1](#_bookmark13)可知，

− 1 *r r*−1 1



*r*  κ(*G*(*mr*−1, . . . , *m*1)) = κ(*G*) − 1，若*mr* = 2

κ(*G*(*mr*, *mr*−1, . . . , *m*1) − *V*(*G* [*i*])) =

 κ(*G*(*mr* −

1, *m*

*r*−1

, . . . , *m*1)) = κ(*G*) −

1，若*mr* ≥ 3

因为|*F* ∩ *V*(*G* − *V*(*Gr*[*i*]))| ≤ κ(*G*) − 2且*G*(*mr*, *mr*

−

*r*

1, . . . , *m*1) − *V*(*Gr*[*i*])的连通度为κ(*G*) − 1，

所以*G*(*mr*, *mr*−1, . . . , *m*1) − (*V*(*G* [*i*]) ∪ *F*)是连通的。 口

定理 **3.1.** *G*(*mr*, *mr* 1, . . . , *m* )的1−限制连通度κ1(*G*) ≥ 2κ(*G*) − *n*， 其中*r* ≥ 2 且*n* =

− 1

max{*m*1, *m*2, . . . , *mr*}。

证明**.** 令*F*为*G*(*mr*, *mr*−1, . . . , *m*1)的一个顶点集合。要证明本定理等价于证明下列叙述： 当|*F*| ≤ 2κ(*G*) − *n* − 1 且*G* − *F*中任意一个顶点都至少有一个无故障邻居时，那么， *G* − *F*中对于任意两个顶点*u*和*v* 都存在一条从*u*到*v*的无故障路径。

若*u*和*v*相 邻， 则 本 定 理 得 证。 因 此， 下 面 我 们 讨 论*u*和*v*不 相 邻 的 情 况。 假

设*u* ∈ *Gr*[α]且*v* ∈ *Gr*[β]，其中α, β ∈ ⟨*mr* −1⟩。令*F*α = *V*(*Gr*[α])∩ *F* 且*F*β = *V*(*Gr*[β])∩ *F*。 接下来，我们讨论以下两种情形进行。

情形1. α 芋 β。

情形1.1. |*F*α| ≥ κ(*Gr*[α])或|*F*β| ≥ κ(*Gr*[β])。

不失一般性，假设|*F*α| ≥ κ(*Gr*[α])。令*J* = ⟨*mr* −1⟩\{α}且*FJ* = ∪ *j J F* 。则有*Gr*[*J*] =

∈ *j*

*G* − *V*(*Gr*[α])。 显然|*F* ∩ *V*(*G* − *V*(*Gr*[α]))| = |*F* \ *F*α| ≤ 2κ(*G*) − *n* − 1 − (κ(*G*) − *mr* + 1) =

κ(*G*) − *n* + *mr* − 2 ≤ κ(*G*) − 2。 根据引理[3.2](#_bookmark21) 可知*Gr*[*J*] − *FJ* 是连通的。 接下来，我们讨

论以下两种情形。

情形1.1.1. *NG* (*u*) ∩ *V*(*Gr*[*J*]) cJ *F*。

显 然， 在*NG*(*u*) ∩ *V*(*Gr*[*J*])中， 必 定 存 在 一 个 无 故 障 顶 点*x*， 其 中*x* 芋 *v*。 因 为*Gr*[*J*] − *FJ* 是连通的，所以*Gr*[*J*] − *FJ* 中必定存在一条从*u*到*v*的无故障路径*P*。因此， (*u*, *P*)是*G* − *F*中的一条从*u*到*v*的无故障路径。 如图[3-2](#_bookmark24)(a)所示。

情形1.1.2. *NG* (*u*) ∩ *Gr*[*J*] ⊆ *F*。

因为*u*在*G*中至少有一个无故障邻居且*u*在子图*Gr*[*J*] 中的邻居都是故障的，所

以*u*在子图*Gr*[α]中一定存在无故障邻居*x*。 令*u*1, *u*2, . . . , *u*κ(*G*)

−

*m* 为*u*在子图*Gr*[α]中除

了*x* 之外的所有邻居，且令*u*′ , *u*′ , . . . , *u*′

*r*

分别为*u*1, *u*2, . . . , *u*κ(*G*) *m* 在子图*Gr*[*J*]中的

1 2 κ(*G*)−*mr*

− *r*

*r*

邻居。同样的，令*x*1, *x*2, . . . , *x*κ(*G*)−*mr* 为*x*在子图*G* [α]中除了*u*之外的所有邻居。根据引 理[3.1](#_bookmark20)可知，|{*u*1, *u*2, . . . , *u*κ(*G*)−*mr* } ∩{*x*1, *x*2, . . . , *x*κ(*G*)−*mr* }| ≤ *n* − 2，其中*n* = max {*m*1, *m*2, . . . ,

*mr*}。 令{*u*1, *u*2, . . . , *u*κ(*G*)−*mr* } ∩ {*x*1, *x*2, . . . , *x*κ(*G*)−*mr* } = {*u*1, *u*2, . . . , *ul*}，其中*l* ≤ *n* − 2。 不失

一般性，假设{*u*1, *u*2, . . . , *u*κ(*G*)−*mr* } ∩ {*x*1, *x*2, . . . , *x*κ(*G*)−*mr* −*l*} = ∅。 令*x*′ , *x*′ , . . . , *x*′ 分

1 2 κ(*G*)−*mr* −*l*

*r r*

别为*x*1, *x*2, . . . , *x*κ(*G*)−*mr* −*l*在*G* [*J*]中的邻居且*W* = {*wi* | *wi* ∈ *NG*(*x*) ∩ *V*(*G* [*J*])}。 根据定

义[2.1](#_bookmark13)可知，{*u*′ , *u*′ , . . . , *u*′

} ∩ {*x*′ , *x*′ , . . . , *x*′

} = ∅ 且*W* ∩ ({*u*′ , *u*′ , . . . , *u*′ } ∪

1 2 κ(*G*)−*mr* 1 2

κ(*G*)−*mr* −*l*

1 2 κ(*G*)−*mr*

{*x*′ , *x*′ , . . . , *x*′

}) = ∅。 因此，必定存在2κ(*G*) − 2*mr* − *l* + |*J*|条以*u*为起点且以子

1 2 κ(*G*)−*mr* −*l*

图*Gr*[*J*] 中顶点为终点的路径，如图[3-1](#_bookmark23)所示。 具体路径列举如下：

*Pi* = (*u*, *ui*, *u*′)， 其中1 ≤ *i* ≤ κ(*G*) − *mr*

*i*

*Qi* = (*u*, *x*, *xi*, *x*′)，其中1 ≤ *i* ≤ κ(*G*) − *mr* − *l*

*i*

*Qi* = (*u*, *x*, *wi*)， 其中κ(*G*) − *mr* − *l* + 1 ≤ *i* ≤ κ(*G*) − *mr* − *l* + |*J*|

因为*l* ≤ *n* − 2，所以2κ(*G*) − 2*mr* − *l* + |*J*| ≥ 2κ(*G*) − 2*mr* − *n* + 2 + |*J*|。 因为|*J*| =

|⟨*mr* − 1⟩ \ {α}| = *mr* − 1， 所以2κ(*G*) − 2*mr* + |*J*| − *n* + 2 = 2κ(*G*) − *mr* − *n* + 1。 因 此，至少有2κ(*G*) − *mr* − *n* + 1条以*Gr*[α]中的顶点*u*为起点且以*Gr*[*J*]的顶点为终点 的路 径。 目前在*NG*(*u*) ∩ *V*(*Gr*[*J*])中的故障的顶点数为|*F* − (*NG*(*u*) ∩ *V*(*Gr*[*J*]))|，既 然|*F* − (*NG* (*u*) ∩ *V*(*Gr*[*J*]))| ≤ (2κ(*G*) − *n* − 1) − |*J*| = 2κ(*G*) − *mr* − *n*， 那 么 在 上 述 的2κ(*G*) − *mr* − *n* + 1条路径中至少存在一条无故障路径*P*。假设*z*为*P*的终点。若*z* = *v*， 则*P*即为*u*和*v*在*G* − *F*中的一条无故障路径。若*z* 芋 *v*，因为*Gr*[*J*]− *FJ* 是连通的且*z*和*v*都 在子图*Gr*[*J*]中，所以子图*Gr*[*J*]中必定存在一条从*z*到*v*的无故障路径*P*′。 因此，路

径(*P*, *P*′)为*G*中一条从*u*到*v*的无故障路径。 情形1.2. |*F*α| < κ(*Gr*[α])且|*F*β| < κ(*Gr*[β])。

因为|*F*β| < κ(*Gr*[β])且*Gr*[β]的连通度为κ(*Gr*[β])，所以子图*Gr*[β] − *F*β是连通的。 令*x*是*u*在子图*Gr*[β]中的邻居。 若*x* <! *F*，因为*Gr*[β] − *F*β是连通的，所以子图*Gr*[β] − *F*β中必定存在一条从*x*到*v*的无故障路径*P*。 因此，路径(*u*, *P*)即为在*G* − *F*中一条 从*u*到*v*的无故障路径。 若*x* ∈ *F*，我们分以下两种情况讨论。

情形1.2.1. |*F*α ∪ *F*β| ≤ 2κ(*G*) − 2*mr* − *n* + 3。

既然|*F*α| < κ(*Gr*[α])，那么在*Gr*[α]中*u*至少有一个无故障邻居*y*。 令*J* = {β}。 类似 于情形1.1.2的讨论(如图[3-1](#_bookmark23))，*G*中存在2κ(*G*) − 2*mr* + |*J*| − *n* + 2 = 2κ − 2*mr* − *n* + 3 条 以子图*Gr*[α]中的*u*为起点并以*Gr*[β]的顶点为终点的路径。 既然2κ(*G*) − 2*mr* − *n* + 3 −

|(*F*α ∪ *F*β) \ {*x*}| ≥ 1，那么在上述的路径中至少存在一条无故障路径*P*。假设*z*为*P*的终 点。 若*z* = *v*，则*P*即为*u*和*v*在*G* − *F*中的一条无故障路径。 若*z* 芋 *v*，因为*Gr*[*J*] − *FJ* 是 连通的，且*z*和*v*都在子图*Gr*[β] 中，所以在子图*Gr*[*J*] − *FJ* 中必定存在一条从*z*到*v*的无 故障路径*P*′。 因此，路径(*P*, *P*′)为*G* − *F*中的一条从*x*到*v*的无故障路径(如图[3-2](#_bookmark24)(b)所

示)。

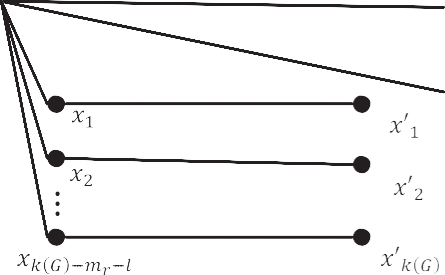
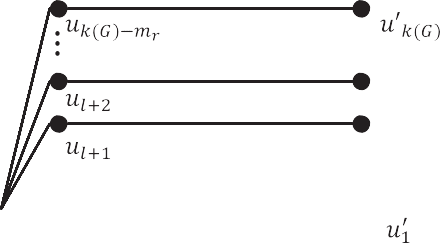


图 3-1 *u*连通到*Gr* [*J*]的2κ(*G*) − 2*mr* − *n* + 2 + |*J*|条路径

情形1.2.2. |*F*α ∪ *F*β| > 2κ(*G*) − 2*mr* − *n* + 3。

首先声明，由于|*F*α ∪ *F*β| ≤ κ(*Gr*[α]) − 1 + κ(*Gr*[β]) − 1 = 2κ(*G*) − 2*mr*，该情形 在*r* = 2时是不存在的。 显然|*F* \ (*F*α ∪ *F*β)| ≤ (2κ(*G*) − *n* − 1) − (2κ(*G*) − 2*mr* − *n* + 4) = 2*mr* − 5，因为*G* \ (*Gr*[α] ∪ *Gr*[β])中的每个子图为*Gr*[*i*]，其中*i* ∈ ⟨*mk*⟩ \ {0, α, β}，那么共 有|⟨*mk*⟩ \ {0, α, β}| = *mk* − 2个子图。 假设每个子图中的总故障顶点数都大于等于2，则 有|*F* \ (*F*α ∪ *F*β)| ≥ 2(*mr* − 2) = 2*mr* − 4 > 2*mr* − 5，该结论与|*F* \ (*F*α ∪ *F*β)| ≤ 2*mr* − 5矛

盾。 因此该假设不成立，即一定存在一个子图的总故障顶点数小于等于1。 不妨假设

子图*Gr*[γ]的总故障顶点数小于等于1。 令*y*与*z*分别为*u*与*v*在子图*Gr*[γ]中的邻居。 因

为|*F*γ| ≤ 1且*y* 芋 *z*，则*y*和*z*中至少有一个无故障顶点*z*。若*y* <! *F*γ，则*y*和*z*在*Gr*[γ]−*F*γ中 必定存在1条无故障路径*P*。 则路径(*u*, *P*, *v*)即是*G* − *F*中一条从*u*到*v*的无故障路径。 若*y* ∈ *F*γ，既然*u*在图*G*中至少存在一个无故障邻居，则*u*在*G* − *Gr*[γ]中存在一个无故 障邻居*u*′。 显然*u*′在子图*Gr*[γ]中的邻居*u*′′是无故障的。 同理可得，顶点*Gr*[γ] − *F*γ中 必定存在一条从*u*′′到*z*无故障路径*P*。因此，路径(*u*, *u*′, *P*, *v*)是*G* − *F*中一条从*u*到*v*的无

故障路径(如图[3-2](#_bookmark24)(c))。

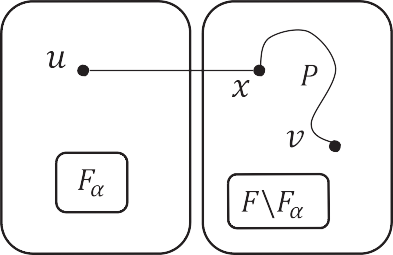
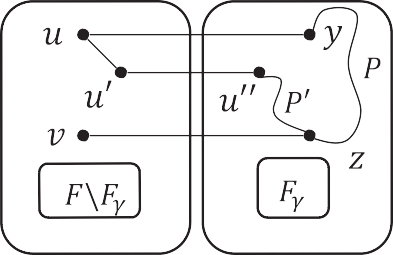
情形2. α = β。

若|*F*α| < κ(*Gr*[α])，则*Gr*[α] − *F*α是连通的，因此本引理得 证。 接下来我们考 虑|*F*α| ≥ κ(*Gr*[α])的情况。令*J* = ⟨*mr* − 1⟩ \ {α}，则*Gr*[*J*] = *G* − *Gr*[α]。类似于情形1.1的 讨论，可以推出在*Gr*[*J*] − *FJ* 中必定存在一条从*u*到*x*的无故障路径*P*1，同时也必定 存在一条从*v*到*y*的无故障路径*P*2。 假设*V*(*P*1) ∩ *V*(*P*2) 芋 ∅。 不妨令*z*为*P*1和*P*2的首个 公共顶点，则路径(Path(*P*1, *u*, *z*), Path(*P*−1, *z*, *v*))即为*u*和*v*在*G* − *F*中的一条无故障路 径，因此本引理得证。 假设*V*(*P*1) ∩ *V*(*P*2) = ∅，因为|*F* ∩ (*G* − *V*(*Gr*[α]))| = |*F* \ *F*α| ≤ (2κ(*G*) − *n* − 1) − (κ(*G*) − *mr* + 1) = κ(*G*) − *n* + *mr* − 2 ≤ κ(*G*) − 2，根据引理[3.2](#_bookmark21) 可 知*Gr*[*J*] − *FJ* = *G* − (*V*(*Gr*[α]) ∪ *F*)是 连 通 的， 所 以 在*Gr*[*J*] − *FJ* 中 必 定 存 在 一 条 从*x*到*y*的无故障路径*P*3。 因此，路径(*P*1, *P*3, *P*−1)则是*G* − *F*中一条从*u*到*v*的无故障路

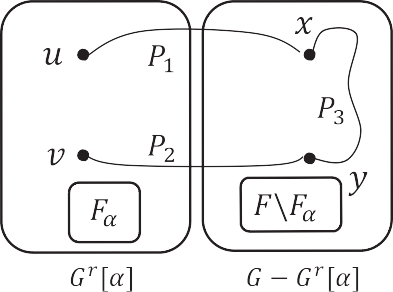
2

2

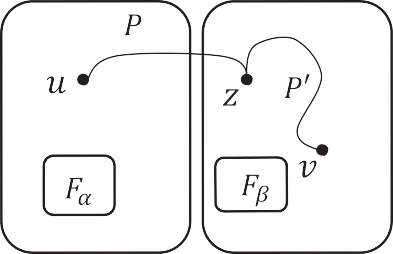
径(如图[3-2](#_bookmark24)(d))。

#### (c)



(d)



#### (b)

图 3-2 *G*中从*u*到*v*的无故障路径

综上所述，对于*G* − *F*中任意两个无故障顶点*u*和*v*，都有一条从*x*到*v*的无故障路 径，也就是说*G* − *F*是连通的。 因此，κ1(*G*) ≥ 2κ(*G*) − *n*，本定理得证。 口

接下来，在给出定理[3.2](#_bookmark26)之前，我们先给出引理[3.3](#_bookmark25)。

引理 **3.3.** 若*u*和*x*是*G*(*mr*, *mr*−1, . . . , *m*1)中任意两个相邻的顶点，*y*是图中的另外一个顶 点，且*y* <! *NG*({*u*, *x*})，则|*NG*(*y*) ∩ *NG*({*u*, *x*})| ∈ {0, 2, 3}。 特别地，若*G*同构于超立方体， 则|*NG*(*y*) ∩ *NG*({*u*, *x*})| ∈ {0, 2}。

证明**.** 不失一般性，假设*dist*(*u*, *y*) ≥ *dist*(*x*, *y*)。因为*y* <! *NG* ({*u*, *x*})，可以推出*dist*(*u*, *y*) ≥

*dist*(*x*, *y*) ≥ 2。 因此，我们讨论以下两种情形。 情形1. *dist*(*x*, *y*) ≥ 3。

根据定义[2.1](#_bookmark13)可知*NG*(*x*) ∩ *NG*(*y*) = ∅。 因为*dist*(*u*, *y*) ≥ *dist*(*x*, *y*) ≥ 3，所以*NG*(*u*) ∩

*NG*(*y*) = ∅。 因此|*NG*(*y*) ∩ *NG*({*u*, *x*})| = |(*NG*(*x*) ∩ *NG*(*y*)) ∪ (*NG* (*u*) ∩ *NG*(*y*))| = 0。

情形2. *dist*(*x*, *y*) = 2。

令*u* = *urur*−1 . . . *u*1， *x* = *xr xr*−1 . . . *x*1 且*y* = *yryr*−1 . . . *y*1。 若α, β ∈ *V*(*G*)， 令 顶

点*zj*,α,β = *zrzr*

1 . . . *z* ，其中*i* ∈ ⟨*r*⟩ \ {0, *j*}，*z*

= β 且*z*

= α 。 因为*dist*(*x*, *y*) = 2，所以存

− 1 *i i j j*

在两个非负整数*w*1和*w*2，满足*xw*1 芋 *yw*1 ，*xw*2 芋 *yw*2 和*xi* = *yi*，其中*i* ∈ ⟨*r*⟩ \ {0, *w*1, *w*2}。

根据定义[2.1](#_bookmark13)可知*NG*(*x*) ∩ *NG*(*y*) = {*zw*1,*x*,*y*, *zw*2,*x*,*y*}。 接下来，我们讨论以下两种情形。 情形2.1. *dist*(*u*, *y*) ≥ 3。

根据定义[2.1](#_bookmark13)可知*NG*(*y*) ∩ *NG*(*u*) = ∅。 因此|*NG*(*y*) ∩ *NG*({*u*, *x*})| = |{*zw*1,*x*, *zw*2,*x*}| = 2。

情形2.2. *dist*(*u*, *y*) = 2。

因为*dist*(*u*, *y*) = 2， 所以存在两个非负整数*s*1和*s*2， 满足*us*1 芋 *ys*1 ， *us*2 芋 *ys*2

和*ui* = *yi*，其中*i* ∈ ⟨*r*⟩ \ {0, *s*1, *s*2}。 由(*u*, *x*) ∈ *E*(*G*)和定义[2.1](#_bookmark13)可知，必存在一个非负整 数*t*，满足*ut* 芋 *yt*且*ui* 芋 *yi*，其中*i* ∈ ⟨*r*⟩ \ {0, *t*}。 假设*t* <! {*s*1, *s*2}，则*ys*1 芋 *us*1 = *xs*1 ，*ys*2 芋 *us*2 = *xs*2 且*yt* = *ut* 芋 *xt*。 显然，*dist*(*u*, *x*) = 3，与上述的假设矛盾。 因此，我们考虑*t* ∈

{*s*1, *s*2}。 类似地，可以推出*t* ∈ {*w*1, *w*2}。 不失一般性，假设*t* = *s*1 = *w*1。 因为*us*1 芋 *ys*1 ，

*ut* 芋 *xt*和*xw*1 芋 *yw*1 ，所以*ut* 芋 *xt* 芋 *yt*，且*mt* ≥ 3。 因此*NG*(*u*) ∩ *NG*(*y*) = {*z*

*s*1,*u*,*y*

, *zs*2,*u*,*y*

}。 因

为*us*2 = *xw*2 ，所以*z*

*s*2,*u*,*y*

= *zw*2,*x*,*y*。 从而，|*NG*(*y*) ∩ *NG*({*u*, *x*})| = |{*z*

*w*1,*x*,*y*

, *zw*2,*x*,*y*

, *zs*1,*u*,*y*

}| = 3。

特别地，当该图同构于超立方体时，不存在*dist*(*x*, *y*) = 2且*dist*(*u*, *y*) = 2这种情况。 综 上 所 述，|*NG* (*y*) ∩ *NG*({*u*, *x*})| ∈ {0, 2, 3}。 特 别 地， 当*G*同 构 于 超 立 方 体 时，

有|*NG*(*y*) ∩ *NG*({*u*, *x*})| ∈ {0, 2}。 口

定理 **3.2.** *G*(*mr*, *mr* 1, . . . , *m* )的1−限制连通度κ1(*G*) ≤ 2κ(*G*) − *n*，其中*r* ≥ 2，且*n* =

− 1

max{*m*1, *m*2, . . . , *mr*}。

证明**.** 若*mi* = *n*，则假设*u* = *urur*−1 . . . *u*1 ∈ *V*(*G*) 且*x* = *xr xr*−1 . . . *x*1 ∈ *NG*(*u*)。 若 *j* ∈

⟨*r*⟩ \ {0, *i*}，则*ui* 芋 *xi*，*uj* = *x j*。因此|*NG*(*x*) ∩ *NG*(*u*)| = |{*yryr*−1 . . . *y*1 | *yi* ∈ ⟨*mi* − 1⟩ \ {*ui*, *xi*}

和*yj* = *uj* = *x j*，其中 *j* 芋 *i*}| = *mi* − 2 = *n* − 2。 若|*NG*(*u*)| = |*NG* (*x*)| = κ(*G*)，则根据定

义[2.1](#_bookmark13)可知，

|*NG*({*u*, *x*})| = |*NG* (*u*) ∪ *NG*(*x*) \ (*NG*(*x*) ∩ *NG*(*u*))|

= (|*NG*(*u*)| − 1) + (|*NG*(*x*)| − 1) − (*n* − 2)

= 2(κ(*G*) − 1) − (*n* − 2)

= 2κ(*G*) − *n*

若*F* = *NG*({*u*, *x*})，显然*G* − *F*中有两个子图，其中一个子图包含{*u*, *x*}两个顶点， 另一个子图是*G* − (*NG*({*u*, *x*}) ∪ {*u*, *x*})。 子图*NG*({*u*, *x*}) ∪ *G*[{*u*, *x*}]中的每个顶点都至少

有一个无故障邻居可以通过如下断言得出。

断言： 子图*G* − (*NG*({*u*, *x*}) ∪ {*u*, *x*})中的每个顶点都至少有一个无故障邻 居。

令*y* ∈ *V*(*G*)\(*NG* ({*u*, *x*}) ∪{*u*, *x*})。若|*NG*(*y*) ∩ *NG*({*u*, *x*})| < |*NG*(*y*)|，则*y*至

少有一个无故障邻居。

根据引理[3.3](#_bookmark25)可知|*NG*(*y*) ∩ *NG*({*x*, *u*})| ∈ {0, 2, 3}。 根据定义[2.1](#_bookmark13)可知，除 了*G*(2, 2)和*G*(2, 3)之外，|*NG*(*y*)| = κ(*G*)，其中*r* ≥ 2。 当κ(*G*) > 3，本断

言显然成 立。 因此，我们只需要证明本断言在κ(*G*) = 3时成 立。 显然

除*G*(2, 3)以外，我们只需考虑*G*(2, 2, 2)。 根据引理[3.3](#_bookmark25) 可知|*NG*(2,2,2)(*y*) ∩

*NG*(2,2,2)({*u*, *x*})| ∈ {0, 2}， 也 就 是 说|*NG*(2,2,2)(*y*) ∩ *NG*(2,2,2) ({*u*, *x*})| < 3。 因

为κ(*G*(2, 2, 2)) = 3，所以*G*(2, 2, 2) − (*NG*(2,2,2)({*u*, *x*}) ∪ {*u*, *x*})中的每一个顶

点*y*都至少有一个无故障邻居。 因此，本断言适用于*G*(2, 2, 2)。

结合断言和上述讨论可知，*G* − *F*是不连通的，且其中的两个子图中每个顶点都 含有至少一个无故障邻居。 因此，κ1(*G*) ≤ |*F*| = 2κ(*G*) − *n*，本定理得证。 口

显然，综合定理[3.1](#_bookmark22)和定理[3.2](#_bookmark26)的结果可以推出定理[3.3](#_bookmark27)。

定 理 **3.3.** 除 了*G*(2, 2)和*G*(2, 3)以 外， *G*(*mr*, *mr* 1, . . . , *m* )的1*-*限 制 连 通 度 为κ1(*G*) =

2κ(*G*) − *n*，其中*r* ≥ 2且*n* = max{*m*1, *m*2, . . . , *mr*}。

− 1

### 3.2 1-限制连通度条件下的容错路由

在本小节中，我们给出当广义超立方体的总故障顶点数小于其1-限制连通度且 每个无故障顶点都至少有一个无故障邻居时，构造*G*(*mr*, *mr*−1, . . . , *m*1)中任意两个无 故障顶点之间的一条无故障路径的容错路由算法GHCFP。 之后，我们会分析并证明

该算法的时间复杂度。

我们给出的GHCFP算法引用了Guo等人的算法BuildPathSet [[28](#_bookmark102)]。 其算法给出*G*中 任意两个顶点*u*和*v*之间的κ(*G*)条不相交路径，得到的最短路径的长度为*dist*(*u*, *v*)， 最长路径的长度为*dist*(*u*, *v*) + 2，且该算法的时间复杂度为*O*(κ(*G*)2)。 接下来，在给 出GHCFP之前，我们先给出BuildPathSet。

**3.2.1** 1**-**限限制制制连连连通通通度度度条条条件件件下下下的的的路路路由由由算算算法法

算法 **1** BuildPathSet(*G*(*mr* , *mr*−1 , . . . , *m*1 ), *u*, *v*)

输入**:** *G*(*mr* , *mr*−1, . . . , *m*1)中两个不同的顶点*u* = *urur*−1 . . . *u*1和*v* = *vrvr*−1 . . . *v*1

输出**:** *G*中，以*u*为起点，*v*为终点的*mr* + *mr*−1 + ... + *m*1 − *r*条路径。

1: **function** BUILDPATHSET(*G*(*mr* , *mr*−1, . . . , *m*1), *u*, *v*)

2: *PathS et* ← ∅;

3: **for** *i* = *k* to 0 **do**

4: **if** *ui* 芋 *vi* **then**

5: *Pi* = *DCRouting*(*u*, *v*, *i*);

6: **else**

7: *c* = a neighbor of *u* at level *i*;

8: *Pi* = *AltDCRouting*(*u*, *v*, *i*, *c*);

9: **end if**

10: add *Pi* to *PathS et*;

11: **end for**

12: **return** *PathS et*;

13: **end function**

14: **function** DCROUTING(*u*, *v*, *i*)

15: *m* = *k*;

16: **for** *j* = *i* to *i* − *k* **do**

17: ∏[*m*] = *j* mod (*k* + 1); /\*∏ is a permutation of [*k*, *k* − 1, . . . , 1, 0]\*/

18: *m* = *m* − 1;

19: **end for**

20: **return** *BCubeRouting*(*u*, *v*, ∏);

21: **end function**

22: **function** ALTDCROUTING(*u*, *v*, *i*, *c*)

23: *path* ← ∅;

24: add *u* to path;

25: *m* = *k*;

26: **for** *j* = *i* − 1 to *i* − 1 − *k* **do**

27: ∏[*m*] = *j* mod (*k* + 1);

28: *m* = *m* − 1;

29: **end for**

30: *path*+ = *BCubeRouting*(*c*, *v*, ∏);

31: **return** *path*;

32: **end function**

33: **function** BCUBEROUTING(*u*, *v*, ∏)

34: *Path* ← ∅;

35: add *u* to path;

36: *Node* ← *u*;

37: **for** *i* = *k* to 0 **do**

38: **if** *u*∏[*i*] 芋 *v*∏[*i*] **then**

39: *Node*∏[*i*] = *v*∏[*i*];

40: append *Node* to *path*;

41: **end if**

42: **end for**

43: **return** *path*;

44: **end function**

接下来，我们给出容错路由算法GHCFP，该算法可以给出*G* − *F*中，任意两个无

故障顶点*u*和*v*之间的一条无故障路径，其中*F*为*G*中的故障顶点集合。

算法 **2** GHCFP(*r*, *u*, *v*, *F*, *G*(*mr* , *mr*−1 , . . . , *m*1 ))

输入**:** *r*-维广义超立方体*G*(*mr* , *mr*−1, . . . , *m*1)，一个故障顶点集合*F* ⊂ *V*(*G*)，两个不同的无故障顶 点*u* = *urur*−1 . . . *u*1 和*v* = *vrvr*−1 . . . *v*1。

输出**:** *G* − *F*中一条从*u*到*v*的无故障路径。

1: **function** GHCFP(*r*, *u*, *v*, *F*, *G*(*mr* , *mr*−1, . . . , *m*1))

2: *G* ← *G*(*mr* , *m*−1, . . . , *m*1);

3: **if** *F* = ∅ **then return** GHCP(*u*, *v*, *G*, *F*);

4: **else if** (*u*, *v*) ∈ *E*(*G*) **then return** (*u*, *v*);

5: **end if**

6: *n* ← max{*m*1, *m*2, . . . , *mr* };

*i*=0

7: *k*1 ← ∑*r*

*i*=0

(*mi* − 1), *k*2 ← ∑*r*−1(*mi* − 1), α ← *ur* , β ← *vr* ;

8: *F*α ← *F* ∩ *V*(*Gr* [α]), *F*β ← *F* ∩ *V*(*Gr* [β]);

9: **if** α 芋 β **then**

10: **if** |*F*α| ≥ *k*2 **then**

11: *P*1 ←PATHSEQ(*u*, *F*, *Gr* [α], *G* − *V*(*Gr* [α]));

12: 令*z*为*P*1的最后一个顶点;

13: **if** *z* = *v* **then return** *P*1;

14: **end if**

15: **return** (*P*1, GHCP(*z*, *v*, *G* − *V*(*Gr* [α]), *F* \ *F*α));

16: **else if** |*F*β| ≥ *k*2 **then**

17: *P*1 ←PATHSEQ(*v*, *F*, *Gr* [β], *G* − *V*(*Gr* [β]));

18: 令*z*为*P*1的最后一个顶点;

19: **if** *z* = *v* **then return** *P*1;

20: **end if**

21: **return** (GHCP(*u*, *z*, *G* − *V*(*Gr* [β]), *F* \ *F*β), *P*−1);

1

22: **else if** |*F*α| + |*F*β| ≤ 2*k*1 − 2*mr* − *n* + 3 **then**

23: *P*1 ← PATHSEQ (*u*, *Gr* [α], *Gr* [β])

24: 令*z*为*P*1的最后一个顶点;

25: **if** *z* = *v* **then return** *P*1;

26: **end if**

27: *P*1+ =GHCP(*z*, *v*, *Gr* [β], *F*β);

28: **return** *P*1;

29: **else**

30: 选取一个子图*Gr* [*i*]，使得|*F* ∩ *Gr* [*i*]| = 1，其中*i* ∈ ⟨*mr* − 1⟩ \ {α, β};

31: *G*1 ← *Gr* [*i*], *F*1 ← *F* ∩ *V*(*G*1);

32: *P*1 ← PATHSEQ(*u*, *F*α ∪ *F*1, *Gr* [α], *G*1);

33: 令*x*为*P*1的最后一个顶点;

34: *P*2 ← PATHSEQ(*v*, *F*β ∪ *F*1, *Gr* [β], *G*1);

35: 令*y*为*P*2的最后一个顶点;

36: **if** *x* = *y* **then return** *P*1 + *P*−1;

2

37: **end if**

38: *P*1+ =GHCP(*x*, *y*, *G*1, *F*1);

39: **return** *P*1 + *P*−1;

2

40: **end if**

41: **else if** |*F*α| < *k*2 **then return** (GHCP(*u*, *v*, *Gr* [α], *F*α));

42: **else**

43: *P*1 ←PATHSEQ(*u*, *F*, *Gr* [α], *G* − *V*(*Gr* [α]));

44: 令*x*为*P*1的最后一个顶点;

45: *P*2 ←PATHSEQ(*v*, *F*, *Gr* [α], *G* − *V*(*Gr* [α]));

46: 令*y*为*P*2的最后一个顶点;

47: **if** *z*为*P*1和*P*2的第一个公共顶点 **then return** Path(*P*1, *u*, *z*)+ Path(*P*−1, *z*, *v*);

2

48: **end if**

49: *P*1+ =GHCP(*x*, *y*, *G* − *V*(*Gr* [α]), *F* \ *F*α);

50: **return** *P*1 + *P*−1;

2

51: **end if**

52: **end function**

53: **function** PATHSEQ(*u*, *F*, *H*1, *H*2)

54: **if** (*NG* (*u*) ∩ *V*(*H*2)) \ *F* 芋 ∅ **then**

55: 从(*NG* (*u*) ∩ *V*(*H*2)) \ *F*中选择一个顶点*u*1;

56: **return** (*u*, *u*1);

57: **else**

58: **for all** (*u*1 ∈ *NG* (*u*) ∩ *V*(*H*1)) \ *F* **do**

59: **if** (*NG* (*u*1) ∩ *V*(*H*2)) \ *F* 芋 ∅ **then**

60: 从(*NG* (*u*1) ∩ *V*(*H*2)) \ *F* 中选择一个顶点*u*2;

61: **return** (*u*, *u*1, *u*2);

62: **end if**

63: **end for**

64: **end if**

65: 从(*NG* (*u*) ∩ *V*(*H*1)) \ *F* 中选择一个顶点*u*1;

66: *S* ← (*NG* (*u*1) ∩ *V*(*H*1)) \ (*F* ∪ (*NG* (*u*) ∩ *V*(*H*1)));

67: **for all** *u*2 ∈ *S* **do**

68: **if then**(*NG* (*u*2) ∩ *V*(*H*2)) \ *F* 芋 ∅

69: 从(*NG* (*u*2) ∩ *V*(*H*2)) \ *F*中选择一个顶点*u*3;

70: **return** (*u*, *u*1, *u*2, *u*3);

71: **end if**

72: **end for**

73: **end function**

74: **function** GHCP(*u*, *v*, *G*, *F*)

75: 在*G* − *F*中，从BuildPathSet(*G*, *u*, *v*)的*k*1条路径中选一条无故障路径;

76: **end function**

**3.2.2** 容容错错错路路路由由由算算算法法法分分分析析

在分析GHCFP的时间复杂度之前，我们先分析PATHSEQ(*u*, *F*, *H*1, *H*2) 和GHCP(*x*, *y*, *G*, *F*)的时间复杂度。 对于函数PATHSEQ(*u*, *F*, *H*1, *H*2)，其中*H*1和*H*2 是*G*的两个子 图且*u*在子图*H*1中。 依据不同的情况，该函数可能会给出*u*到子图*H*2的路径(*u*, *u*1)、 (*u*, *u*1, *u*2)或(*u*, *u*1, *u*2, *u*3)。 PATHSEQ在不同的情形下执行的时间复杂度如下：情形1.

所得到的路径为(*u*, *u*1)。 若(*NG*(*u*) ∩ *V*(*H*2)) \ *F* 芋 ∅，则该函数从(*NG*(*u*) ∩ *V*(*H*2)) \ *F*中

选择一个顶点*u*1，最后返回路径(*u*, *u*1)。 该情况下的时间复杂度为*O*(1)。 情形2. 所

得到的路径为(*u*, *u*1, *u*2)。 该函数首先会遍历集合(*NG*(*u*) ∩ *V*(*H*1)) \ *F*，令集合中每个 顶点为*u*1，然后从(*NG* (*u*1) ∩ *V*(*H*2)) \ *F* 中选择一个顶点*u*2，最终返回路径(*u*, *u*1, *u*2)。 该情况下的时间复杂度为(*O*(|*NG*(*u*) ∩ *V*(*H*1)|)) ≤ *O*(κ(*G*))。 情形3. 所得到的路径

为(*u*, *u*1, *u*2, *u*3)。在该情形下，该函数首先从(*NG*(*u*) ∩ *V*(*H*1)) \ *F*中选择一个顶点*u*1，并 取集合*S* ← (*NG* (*u*1) ∩ *V*(*H*1)) \ (*F* ∪ (*NG* (*u*) ∩ *V*(*H*1))。 该过程的时间复杂度为0(1)。 然 后函数会遍历集合*S* ，令集合中每个顶点为*u*2。 之后从(*NG* (*u*2) ∩ *V*(*H*2)) \ *F*中选择一 个顶点*u*3，最后返回路径(*u*, *u*1, *u*2, *u*3)。 该情况的时间复杂度为(*O*(|*S* |) ≤ *O*(κ(*G*))。 综

合情况1、情况2和情况3可得到函数PATHSEQ(*u*, *F*, *H*1, *H*2)的时间复杂度为*O*(κ(*G*))。

接 下 来， 我 们 分 析 函 数GHCP(*u*, *v*, *G*, *F*)的 时 间 复 杂 度。 该 函 数 首 先 调 用 函

数BuildPathSet(*G*, *u*, *v*) [[28](#_bookmark102)]，得到*u*和*v*在*G* − *F*中的κ(*G*)条不相交路径。该过程的时间复 杂度为*O*(κ(*G*)2)。 然后函数GHCP遍历这些不相交路径，从中找出一条无故障路径。

其中，若判断一条路径是否为无故障路径需要判断路径中的每个顶点是否在故障集

合*F*中。因为每条路径中至多有*dist*(*u*, *v*)+ 2个顶点且|*F*| = 2 × κ(*G*) − *n* − 1，所以该过程 的时间复杂度为*O*(*dist*(*u*, *v*) + 2) × *O*(2 × κ(*G*) − *n* − 1) ≤ *O*(κ2)。因为调用BuildPathSet总

共得到κ(*G*)条路径，那么该判断至多执行κ(*G*)次。综上所述，找到一条无故障路径的

时间复杂度为*O*(κ2) × *O*(κ) = *O*(κ(*G*)3)。 最后，我们分析GHCFP的时间复杂度。 CHCFP在α 芋 β和α = β两种情形下的时

间复杂度如下。 情形1. α 芋 β。

情形1.1. *F*α ≥ *k*2。 GHCFP首先通过PATHSEQ(*u*, *Gr*[α], *G* − *V*(*Gr*[α])) 得到一条 从*u*到*G* − *V*(*Gr*[α])中的顶点*z*的*P*1，该过程的时间复杂度为*O*(κ(*G*))。 然后，若*z* = *v*， 则GHCFP返回*P*1。 因此，得到*u*和*v*在*G* − *F*中的一条无故障路径*P*1的时间复杂度 为*O*(κ(*G*))。若*z* 芋 *v*，GHCFP会返回路径(*P*1,GHCP(*z*, *v*, *G* − *V*(*Gr*[α]), *F* \ *F*β))。该过程 需要首先调用函数GHCP(*z*, *v*, *G* − *V*(*Gr*[α]), *F* \ *F*β)建立一条无故障路径，其时间复杂 度为*O*(κ(*G*)3)。 然后拼接*P*1和上述GHCP(*z*, *v*, *G* − *V*(*Gr*[α]), *F* \ *F*β)所构建的无故障路 径，该过程的时间复杂度是*O*(1)。 因此，得到*u*和*v*在*G* − *F*中的无故障路径的时间复

杂度为*O*(κ(*G*)) + *O*(κ(*G*)3) + *O*(1) = *O*(κ(*G*)3)。

情形1.2. (|*F*β| ≥ *k*2)或(|*F*α + *F*β| ≤ 2*k*1 − 2*mr* − *n* + 3)。 类似于在情形1.1中的讨论， 我们得到*u*和*v*在*G* − *F*中的无故障路径的时间复杂度为*O*(κ(*G*)3)。

情形1.3. |*F*α + *F*β| > 2*k*1 − 2*mr* − *n* + 3。 GHCFP首先会选择一个满足条件|*F* ∩ *Gr*[*i*]| = 1的子图*Gr*[*i*]，其中*i* ∈ ⟨*mr*⟩ \ {α, β}。 令*G*1 = *Gr*[*i*]。 则该过程的由于时间 复杂度为*O*(*mr* − 2) ≤ *O*(κ(*G*))。 然后，GHCFP将得到一条从*u*到子图*G*1中顶点*x*的

无故障路径*P*1 以及一条从*v*到子图*G*1中顶点*u*的无故障路径*P*2，得到这两条路径 的时间按复杂度分别为*O*(κ(*G*))。 接下来，若*x* = *y*，则GHCFP返回*P*1。 因此，得

到*G* − *F*中一条从*u*到*v*的无故障路径的时间复杂度为*O*(κ(*G*) + 2 × κ(*G*)) = *O*(κ(*G*))； 若*x* 芋 *y*，则GHCFP返回路径(*P*1,GHCP(*x*, *y*, *G*1, *F* ∩ *V*(*G*1)), *P*−1)。 该过程需要首先调 用函数GHCP(*x*, *y*, *G*1, *F* ∩ *V*(*G*1))建议一条无故障路径，其时间复杂度为*O*(κ(*G*)3)。 然 后拼接*P*1、 上述GHCP(*x*, *y*, *G*1, *F* ∩ *V*(*G*1))所构建的无故障路径和*P*−1，该过程的时 间复杂度是*O*(1)。 因此，得到*G* − *F*中一条从*u*到*v*的一条无故障路径的时间复杂度 为*O*(κ(*G*) + 2 × κ(*G*) + κ(*G*)3 + 1) = *O*(κ(*G*)3)。

2

2

情形2. α = β。

情形2.1. *F*α ≤ *k*2。 GHCFP将直接调用算法GHCP(*u*, *v*, *Gr*[α], *F*α)建立一条无故障 路径，则该过程的时间复杂度为*O*(κ(*G*)3)。

情形2.2. *F*α > *k*2。 GHCFP将首先调用函数PATHSEQ得 到一条从*u*到子图*G* −

*V*(*Gr*[α])中*x*的无故障路径*P*1，以及一条从*v*到子图*G* − *V*(*Gr*[α])中的顶点*y*的无故障路

径*P*2。 这里得到*P*1和*P*2的时间复杂度分别为*O*(κ(*G*))。 然后，若*P*1和*P*2的存在公共顶 点，令*z*为第一个公共顶点，那么GHCFP将返回路径(Path(*P*1, *u*, *z*), Path(*P*−1, *z*, *v*))。 因 为*P*1和*P*2中至多有*dist*(*u*, *v*)+2个顶点，所以得到公共顶点*z*的时间复杂度为*O*(|*V*(*P*1)|×

2

|*V*(*P*2)|) ≤ *O*((*dist*(*u*, *v*) + 2) × (*dist*(*u*, *v*) + 2)) ≤ *O*(κ(*G*)2)。 因为得到路径(*P*1, *u*, *z*)的时

间复杂度为*O*(1)，因此，得到顶点*u*和*v*在*G* − *F*中的一条无故障路径(Path(*P*1, *u*, *z*), Path(*P*−1, *z*, *v*)) 的时间复杂度为*O*(2 × κ(*G*)) + *O*(κ(*G*)2) = *O*(κ(*G*)2)。 若*P*1和*P*2无公共顶 点，GHCFP会返回路径(*P*1,GHCP(*x*, *y*, *G* − *V*(*Gr*[α]), *F* \ *F*α), *P*−1)。该过程会先调用算 法GHCP(*x*, *y*, *G* − *V*(*Gr*[α]), *F* \ *F*α)建立一条无故障路径，它的时间复杂度为*O*(κ(*G*)3)。 然后拼接*P*1、上述GHCP(*x*, *y*, *G* − *V*(*Gr*[α]), *F* \ *F*α)所构建的无故障路径和*P*−1，该过程 的时间复杂度为*O*(1)。因此, 得到*u*和*v*在*G* − *F*中的一条无故障路径(*P*1, *GHCP*(*x*, *y*, *G* −

2

2

2

*V*(*Gr*[α]), *F* \ *F*α), *P*−1)的时间复杂度为*O*(2 × κ(*G*)2) + *O*(κ(*G*)3) + *O*(1) = *O*(κ(*G*)3)。

2

综合以上所有情形，GHCFP的时间复杂度为*O*(κ(*G*)3)。

**3.2.3** 模模拟拟拟实实实验验

本小节中对GHCFP用Python语言编程实现，用一台配置为Intel Core i5-7200 CPU 2.71GHz与8 GB 内存的计算机进行模拟实验并分析实验结果。 基于容错路由算法的 目的是在较短时间内找到两个无故障顶点之间尽可能短的一条无故障路径。 本实验

从执行效率和所获得的路径长度两个方面对算法的性能进行测试。 同时，为了更好 的模拟广义超立方体网络规模的扩大，本模拟实验将分别从改变维数和改变基数两 个方面进行实验。 同时，在改变基数的模拟实验中，我们基于4-维的广义超立方体 进行测试。

GHCFP给出的是给出当广义超立方体的总故障顶点数小于其1-限制连通度且每 个无故障顶点都至少有一个无故障邻居条件时，*G*(*mr*, *mr*−1, . . . , *m*1)中任意两个无故 障顶点之间的一条无故障路径。 在本实验中，我们首先调用*random*函数随机生成故

障顶点集合*F*，满足：

(1) *F* ⊂ *V*(*G*)，

(2) |*F*| = κ1(*G*) − 1，

(3) *G* − *F*中每个顶点都存在至少一个邻接点。

然 后， 在*G* − *F*中 随 机 生 成 一 个 无 故 障 顶 点*x*， 调 用GHCFP和BFS分 别 构 造 出 顶

点*x*到*G*中其余所有无故障顶点之间的无故障路 径。 我将上述过程分别重复执

行100次。 我们针对实验结果进行以下分析。 分析一：计算实验中这两个算法得到一条无故障路径需要花费的平均时间并进

行对比，其实验结果如图[3-3](#_bookmark33)和图[3-4](#_bookmark34)所示。

**(䣓)**

**0.018**

**0.016**

**0.014**

**0.012**

**0.010**

**0.008**

**0.006**

**0.004**

**0.002**

**0**

**G( 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 , 3 )G( 3 , 3 , 3 , 3 , 3 , 3 , 3 )**

**澻澼澷澺濄 澶澺濇**

图 3-3 改变维数的情况下，得到一条无故障路径需要花费的平均时间

从图[3-3](#_bookmark33)中可以看出GHCFP所花费的平均时长小于BFS。 尽管在维数较小时，

GHCFP的所花费的时间与BFS所花费的时间相差不大(如图[3-3](#_bookmark33)中的3-维*G*(3, 3, 3)和4-

维*G*(3, 3, 3, 3))。 但是随着维数的增加，GHCFP执行效率明显较BFS好。 根据实验结

、。

**0.0045**



**0.0040**

**0.0035**

**0.0030**

**0.0025**

**0.0020**

**0.0015**

**0.0010**

**0.0005**

**0**

**G( 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 4 ) G( 3 , 3 , 4 , 4 ) G( 3 , 4 , 4 , 4 ) G( 3 , 4 , 4 , 6 ) G( 3 , 4 , 5 , 6 )**

**澻澼澷澺濄 澶澺濇**

图 3-4 改变基数的情况下，得到一条无故障路径需要花费的平均时间

果可知，当*r* = 7时，GHCFP和BFS所花费的平均CPU时间分别为0.4毫秒和15.8毫秒。 同时，从图[3-4](#_bookmark34)中可以看出，在维数相同的情况下，随着基数的逐渐增大，BFS增 长速度要远大于GHCFP。 因此，从实验数据可以得到，本文算法的执行效率优 于BFS的效率。

分析二：分别计算实验中GHCFP和BFS所得到的所有路径的平均长度并进行对 比，其实验结果如图[3-5](#_bookmark35)和图[3-6](#_bookmark36)所示。

**6**



**5**

ᒣ

൷ **4**

䐟

ᖴ

**3**

䮯

ᓖ

**2**

**1**

**0**

**G(3,3,3漒 G(3,3,3,3) G(3,3,3,3,3) G(3,3,3,3,3,3) G(3,3,3,3,3,3,3)**

**GHCFP BFS**



图 3-5 改变维数的情况下，GHCFP和BFS得到所有路径的平均长度

对比图中的数据可以看出，GHCFP所得到的所有路径的平均长度略大于BFS的 结果。 随着维数的增加或者基数的增大，两者的差值逐渐减小，并接近于0。 也就是 说，随着网络规模的扩大，GHCFP所得到的路径长度很接近最短路径的长度或者等

**4.10**



ᒣ **4.00**



൷

䐟 **3.90**

ᖴ

䮯

**3.80**

ᓖ

**3.70**

**3.60**

**3.50**

**G(3,3,3,3) G(3,3,3,4) G(3,3,4,4) G(3,4,4,4) G(3,4,4,6) G(3,4,5,6)**

**GHCFP BFS**



图 3-6 改变基数的情况下，GHCFP和BFS得到所有路径的平均长度

于最短路径的长度。 分析三：分别计算两种算法获得的每一条路径的差值，最后得出所有路径的差

值的最小值、平均值和最大值并进行对比。 实验结果如表[3-1](#_bookmark37)和表[3-2](#_bookmark38)所示。

表 3-1 改变维数的情况下，两个算法路径差值的最小值，平均值和最大值

|  |  |  |  |
| --- | --- | --- | --- |
| *G*(*mr*, *mr*−1, . . . , *m*1) | 最小值 | 平均值 | 最大值 |
| *G*(3, 3, 3) | 0 | 0.273889 | 3 |
| *G*(3, 3, 3, 3) | 0 | 0.317941 | 3 |
| *G*(3, 3, 3, 3, 3) | 0 | 0.405044 | 3 |
| *G*(3, 3, 3, 3, 3, 3) | 0 | 0.000155 | 2 |
| *G*(3, 3, 3, 3, 3, 3, 3) | 0 | 0.000009 | 1 |

.

表 3-2 改变基数的情况下，两个算法路径差值的最小值，平均值和最大值

|  |  |  |  |
| --- | --- | --- | --- |
| *G*(*mr*, *mr*−1, . . . , *m*1) | 最小值 | 平均值 | 最大值 |
| *G*(3, 3, 3, 3) | 0 | 0.317941 | 3 |
| *G*(3, 3, 3, 4) | 0 | 0.432258 | 3 |
| *G*(3, 3, 4, 4) | 0 | 0.372519 | 3 |
| *G*(3, 4, 4, 4) | 0 | 0.010578 | 2 |
| *G*(3, 4, 4, 6) | 0 | 0.010394 | 2 |
| *G*(3, 4, 5, 6) | 0 | 0.005886 | 1 |

.

从表中可以看出，GHCFP所构造路径长度与BFS所构造路径长度的差值的最小 值始终为0，也就是说GHCFP所得到路径中的部分路径为最短路径。 同时从表[3-1](#_bookmark37)可 以看出，随着网络维数的增加(从6-维*G*(3, 3, 3, 3, 3, 3))开始) GHCFP构造的路径长度 和BFS构造的路径长度的差值平均值和最大值都逐渐减小，也就是GHCFP所得到的 路径长度逐渐接近于最短路径。 另外，明显可以看出两个算法所得到的路径长度差

值的平均值接近于最小值，也就是说GHCFP所构造的路径中路径长度等于或者接近 于最短路径的比例较大。 该实验结果表明GHCFP所得到的路径的路径长度整体是较 优的。

综合以上分析可得，GHCFP相较于BFS在执行效率上表现良好，且GHCFP所得 到的路径的长度中绝大部分接近于最短路径的长度或者等于最短路径的长度。

### 3.3 本章小结

本章基于1-限制连通度的定义，我们求出了广义超立方体的1-限制连通度，并 给出了满足总故障顶点数小于1-限制连通度，且故障网络中每个无故障顶点都至少 有一个无故障邻居条件的容错路由算法。 我们取得如下研究成果：

1. *r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1) 的1-限制连通度是2κ(*G*(*mr*, *mr*−1, . . . , *m*1))− *n*，其中*n* = max{*m*1, *m*2, . . . , *mr*}。

2. 给出了一个满足1-限制连通度条件的无故障单播路由算法GHCFP，且该算法 的时间复杂度为*O*(κ(*G*)3)。

# 第四章 广义超立方体的额外连通度及容错路由

在互连网络中，除了限制连通度以外，额外连通度是衡量网络容错性的另外一 个参数。 本章针对广义超立方体，给出广义超体方体上的1-额外连通度和2-额外连 通度，并且给出2-额外连通度条件下的容错路由算法。

本章主要研究*r*−维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1) 的2-额外连通度，主要结论 如下：(1)对于*r*−维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)，其中*r* ≥ 5且对于每个*i* ∈ ⟨*r*⟩ \

{0}都有*mi* ≥ 3 且*mi* ≥ *mi*−1。 若*mr* > *mr*−1，则广义超立方体的2-额外连通度为3κ(*G*) − 2*mr*；若*mr* = *mr*−1，则广义超立方体的2-额外连通度为3κ(*G*) − 2*mr* − 1。(2)给出一个2- 额外连通度条件下的容错路由算法，其时间复杂度为*O*(*N* × κ(*G*(*mr*, *mr*−1, . . . , *m*1)))，

其中*N*为*G*(*mr*, *mr*−1, . . . , *m*1)的总顶点个数。

### 4.1 广义超立方体的1-额外连通度和2-额外连通度

本节首先给出广义超立方体的1-额外连通度，然后给出其2-额外连通度证明相 关的性质，最后给出广义超立方体的2-额外连通度及其详细的证明过程。

**4.1.1** 广广义义义超超超立立立方方方体体体的的1**-**额额外外外连连连通通通度度

图*G*的1-额外连通度的定义是满足使*G*不连通且去掉*G*中所有故障顶点后的每个 分支中至少有2个顶点，所需要去掉的最少故障顶点的数目，用κ1(*G*)表示。 分析可 得，下面两个条件是等价的：

(a) 去掉*G*中所有故障顶点后，每个顶点都至少有一个邻接点；

(b) 去掉*G*中所有故障顶点后的每个分支中至少有2个顶点。

根据定理[3.2](#_bookmark26)可知κ1(*G*) = κ1(*G*) = 2κ(*G*) − *n*，其中*n* = max{*m*1, *m*2, . . . , *mr*}。 显然， 当一个故障的广义超立方体满足总故障顶点数|*F*| < κ1(*G*) 且*G* − *F*的每个分支中至少

有2个顶点时，第三章所提出的容错路由算法GHCFP同样适用。

**4.1.2** 广广义义义超超超立立立方方方体体体上上2**-**额额外外外连连连通通通度度度下下下的的的基基基本本本性性性质质

因为在*G*(*mr*, *mr*−1, . . . , *m*1)中，任意调整*mr*, *mr*−1, . . . , *m*1 的次序都不会影响其结 构，所以在本章中，我们假设对于任意一个2 ≤ *i* ≤ *r*都有*mi* ≥ *mi*−1。 在给出广义超立

方体的2-额外连通度的具体证明之前，本小节将先给出证明所需要的相关性质。

引理 **4.1.** 在*G*(*mr*, *mr* 1, . . . , *m* )中，对子图*Gk*[ *j*]中的任意3个顶点*u*, *v*, *w*，满足(*u*, *v*),

(*u*, *w*) ∈ *E*(*Gk*[ *j*])，则：

− 1

 3κ(*G*) 3*m* 2*m* + 3, 若*m* > *m*

|*NGk* [ *j*](*u*, *v*, *w*)| ≥ 

 − *k* − *r r r*−1

 3κ(*G*) − 3*mk* − 2*mr* + 2, 若*mr* = *mr*−1

证明**.** 令*u* = *urur*−1 . . . *u*1，*v* = *vrvr*−1 . . . *v*1 且*w* = *wrwr*−1 . . . *w*1。 根据定义[2.1](#_bookmark13)可知存在 唯一一个整数*l*使得*ul* 芋 *vl* 且对于任意一个*i* ∈ ⟨*r*⟩ \ {*l*, 0}，都有*ui* = *vi*，所以*NGk* [ *j*](*u*) ∩ *NGk* [ *j*](*v*) = {*yryr*−1 . . . *y*1 | *yl* ∈ ⟨*ml* − 1⟩ \ {*ul*, *vl*}，其中对任意一个*i* 芋 *l*都有*yi* = *ui* = *vi*}。显

然，|*NGk* [ *j*](*u*)∩ *NGk* [ *j*](*v*)| = *ml* −2。类似地，对于*u*和*v*，可以推出存在唯一一个整数*p*使 得*up* 芋 *wp*，且对于每一个*i* ∈ ⟨*r*⟩ \ {*p*, 0}都有*ui* = *ui*。 类似地，|*NGk* [ *j*](*v*) ∩ *NGk* [ *j*](*w*)| = *mp* − 2。 接下来，我们讨论以下2种情形。

情形1. (*v*, *w*) ∈ *E*(*Gk*[ *j*])。

不失一般性，假设*v*和*w*在第*q*维不同，也就是*vq* 芋 *wq*且对于任意一个*i* ∈ ⟨*r*⟩ \

{*q*, 0}，都有*vi* = *wi*。 类似地，可以推出|*NGk* [ *j*](*v*) ∩ *NGk* [ *j*](*w*)| = *mq* − 2。 假设*l* 芋 *p*。

因 为*ul* 芋 *vl*且*up* 芋 *wp*， 所 以*vl* 芋 *wl*和*vp* 芋 *wp*。 该 结 论 与*v*和*w*仅 在 第*q*维 不 同 的

假 设 矛 盾。 因 此， *l* = *p*。 类 似 地， 可 以 推 出*l* = *q*。 进 而， *ml* = *mp* = *mq*。 因

为|*NGk* [ *j*](*u*) ∩ *NGk* [ *j*](*v*) ∩ *NGk* [ *j*](*w*)| = |{*yryr*−1 . . . *y*1|*yl* ∈ ⟨*ml* − 1⟩ \ {*ul*, *vl*, *wl*}| = *ml* − 3且对于 任意一个*i* 芋 *l* 都有*yi* = *ui* = *vi*}。 因此，

|*NGk* [ *j*](*u*, *v*, *w*)| = |*NGk* [ *j*](*u*)| + |*NGk* [ *j*](*v*)| + |*NGk* [ *j*](*w*)| − |*NGk* [ *j*](*u*) ∩ *NGk* [ *j*](*v*)|

− |*NGk* [ *j*](*u*) ∩ *NGk* [ *j*](*w*)| − |*NGk* [ *j*](*v*) ∩ *NGk* [ *j*](*w*)|

+ |*NGk* [ *j*](*u*) ∩ *NGk* [ *j*](*v*) ∩ *NGk* [ *j*](*w*)| − |{*u*, *v*, *w*}|

= 3(κ(*G*) − (*mk* − 1)) − (*ml* − 2) − (*mp* − 2) − (*mq* − 2) + (*ml* − 3) − 3

= 3κ(*G*) − 3*mk* − 2*ml* + 3

因为*ml* ≤ *mk*，所以|*NGk* [ *j*](*u*, *v*, *w*)| ≥ 3κ(*G*) − 3*mk* − 2*mk* + 3。

情形2. (*v*, *w*) <! *E*(*Gk*[ *j*])。

因为*ul* 芋 *vl*且*up* 芋 *wp*， 所以*vl* 芋 *wl*和*vp* 芋 *wp*。 因为|*NGk* [ *j*](*v*) ∩ *NGk* [ *j*](*w*)| = 2

且|*NGk* [ *j*](*u*) ∩ *NGk* [ *j*](*v*) ∩ *NGk* [ *j*](*w*)| = 0，所以

|*NGk* [ *j*](*u*, *v*, *w*)| = 3(κ(*G*) − (*mk* − 1)) − (*ml* − 2) − (*mp* − 2) − 2 + 0 − 3

= 3κ(*G*) − 3*mk* − *ml* − *mp* + 2

若*mr* > *mr*−1，则*ml* + *mp* < 2*mr*。因此，|*NGk* [ *j*](*u*, *v*, *w*)| > 3κ(*G*) − 3*mk* − 2*mr* + 2；若*mr* =

*mr*−1，则*ml* + *mp* ≤ 2*mr*。 因此，|*NGk* [ *j*](*u*, *v*, *w*)| ≥ 3κ(*G*) − 3*mk* − 2*mr* + 2。

综合情形1和情形2可知本定理成立。 口

定义 **4.1.** 令*cnG*(*S* 1, *S* 2) = |*NG*(*S* 1)∩ *NG*(*S* 1)|，其中*S* 1和*S* 2为*G*(*mr*, *mr*−1, . . . , *m*1)的子图。 显然，根据定义[2.1](#_bookmark13)可推出引理[4.2](#_bookmark45)。

引理 **4.2.** 令*S* 1和*S* 2为*G*(*mr*, *mr*−1, . . . , *m*1)的子图，满足*V*(*S* 1)∩*V*(*S* 2) = ∅ 且*S* 1 <t *NG*(*S* 2)， 其 中*r* ≥ 2且 对 于 任 意 一 个*i* ∈ ⟨*r*⟩ \ {0}， 都 有*mi* ≥ 3。 若*S* 1 三 *K*1且*S* 2 三 *K*1， 则*cnG*(*S* 1, *S* 2) ≤ 2。

基于引理[4.2](#_bookmark45)，我们可以扩展出以下引理。

引理 **4.3.** 令*S* 1和*S* 2为*G*(*mr*, *mr*−1, . . . , *m*1)的子图，满足*V*(*S* 1)∩*V*(*S* 2) = ∅ 且*S* 1 <t *NG*(*S* 2)， 其 中*r* ≥ 2且 对 于 任 意 一 个*i* ∈ ⟨*r*⟩ \ {0}， 都 有*mi* ≥ 3。 若*S* 1 三 *K*1且*S* 2 三 *P*2， 则*cnG*(*S* 1, *S* 2) ≤ 3。

证明**.** 当*r* = 2时，本引理显然成立。 因此，我们将讨论*r* ≥ 3的情形。 不失一般性， 假设*V*(*S* 1) = {*u*}且*V*(*S* 2) = {*x*, *y*}，其中(*x*, *y*)是第 *j*维的边。 我们从第 *j*个维度将*G*切分 为*mj*个子图*Gj*[0], *Gj*[1], . . . , *Gj*[*mj* − 1]。 显然*x*和*y*不属于同一个子图。 不失一般性， 假设*u* ∈ *Gj*[*i*]，其中*i* ∈ ⟨*mj* − 1⟩。 根据*x*和*y*在*G*中的子图的不同，我们讨论以下两种

情形。

情形1. *x* ∈ *Gj*[*i*]或*y* ∈ *Gj*[*i*]。

不失一般性，假设*x* ∈ *Gj*[*i*]且*y* ∈ *Gj*[*i* + 1]。根据引理[4.2](#_bookmark45)可知*cnG j*[*i*] ({*u*}, {*x*}) ≤ 2(如 图[4-1](#_bookmark47)(a))。 根据定义[2.1](#_bookmark13)可知，*cnG*−*Gj*[*i*]({*u*}, {*x*}) = 0且*cnGj*[*i*+1]({*u*}, {*y*}) ≤ 1。 若*NG*−*Gj*[*i*+1] (*u*) ∩ *NG*−*Gj*[*i*+1](*y*) 芋 ∅，则(*u*, *y*) ∈ *E*(*G*)。 因为该结论与*u* <! *NG*(*y*)矛盾，所以*NG*−*Gj*[*i*+1] (*u*)∩ *NG*−*Gj*[*i*+1](*y*) = ∅，即*cnG*−*Gj*[*i*+1] ({*u*}, {*y*}) = 0。 因此，

*cnG*(*S* 1, *S* 2) = *cnGj*[*i*]({*u*}, {*x*}) + *cnG*−*Gj*[*i*]({*u*}, {*x*})

+ *cnGj*[*i*+1]({*u*}, {*y*}) + *cnG*−*Gj*[*i*+1]({*u*}, {*y*})

≤ 2 + 0 + 1 + 0

≤ 3

情形2. *x* <! *Gj*[*i*]且*y* <! *Gj*[*i*]。

不 失 一 般 性， 我 们 假 设*x* ∈ *Gj*[*i* + 1]且*y* ∈ *Gj*[*i* + 2]。 根 据 定 义[2.1](#_bookmark13)可 以 推 出*cnGj*[*i*]({*u*}, {*x*, *y*}) ≤ 1， *cnGj*[*i*+1]({*u*}, {*x*}) ≤ 1 且 *cnGj*[*i*+2]({*u*}, {*y*}) ≤ 1 (如 图[4-1](#_bookmark47)(b)所

示)。 若*NG*−(*Gj*[*i*]∪*Gj*[*i*+1])(*u*) ∩ *NG*−(*Gj*[*i*]∪*Gj*[*i*+1])(*x*) 芋 ∅，则(*u*, *x*) ∈ *E*(*G*)。 因为该结论与*u* <!

*NG*(*x*)矛盾，所以*NG*−(*Gj*[*i*]∪*Gj*[*i*+1]) (*u*) ∩ *NG*−(*Gj*[*i*]∪*Gj*[*i*+1]) (*x*) = ∅，即*cnG*−(*Gj*[*i*]∪*Gj*[*i*+1]) ({*u*}, {*x*})

= 0。 类似地，*cnG*−(*Gj*[*i*]∪*Gj*[*i*+2]) ({*u*}, {*y*}) = 0。 因此，

*cnG*(*S* 1, *S* 2) = *cnGj*[*i*]({*u*}, {*x*, *y*}) + *cnGj*[*i*+1]({*u*}, {*x*}) + *cnGj*[*i*+1]({*u*}, {*y*})

+ *cnGj*[*i*]({*u*}, {*x*, *y*}) + *cnGj*[*i*+1]({*u*}, {*x*}) + *cnGj*[*i*+1]({*u*}, {*y*})

≤ 1 + 1 + 1 + 0 + 0

≤ 3

综上所述，*cnG*(*S* 1, *S* 2) ≤ 3。 因此，本引理成立。 口

*x*

*u*

*y*

*u*

2

*x*

*y*

*G j* [*i*]

(*a*)

*G j* [*i*+1]

*G j* [*i*]

*G j* [*i*+1]

(*b*)

*G j* [*i*+2]

图 4-1 *cnG* (*S* 1, *S* 2)，其中*V*(*S* 1) = {*u*} 且*V*(*S* 2) = {(*x*, *y*)}

引理 **4.4.** 令*S* 1和*S* 2为*G*(*mr*, *mr*−1, . . . , *m*1)的子图，满足*V*(*S* 1)∩*V*(*S* 2) = ∅ 且*S* 1 <t *NG*(*S* 2)， 其 中*r* ≥ 2且 对 于 任 意 一 个*i* ∈ ⟨*r*⟩ \ {0}， 都 有*mi* ≥ 3。 若*S* 1 三 *K*1且*S* 2 三 *P*3， 则*cnG*(*S* 1, *S* 2) ≤ 4。

证明**.** 当*r* = 2时，本引理显然成立。 因此，我们将讨论*r* ≥ 3的情形。 不失一般性，

假设*V*(*S* 1) = {*u*}且*V*(*S* 2) = {*x*, *y*, *z*}，其中{(*x*, *y*), (*y*, *z*)} ⊆ *E*(*G*)且(*x*, *y*)是第 *j*维的边。 我 们从第 *j*个维度将*G*切分为*mj*个子图*Gj*[0], *Gj*[1], . . . , *Gj* [*mj* − 1]。显然，*x*和*y*不属于同 一个子图。 不失一般性，假设*u* ∈ *Gj*[*i*]，其中*i* ∈ ⟨*mj* − 1⟩。 根据*x*，*y*和*z*位置的不同，

我们讨论以下两种情形。

情形1. *z*和*y*在同一个子图。 情形1.1. *x* ∈ *Gj*[*i*]且*y* ∈ *Gj*[*i* + 1]。

根据引理[4.2](#_bookmark45)可知*cnGj*[*i*]({*u*}, {*x*}) ≤ 2。 根据定义[2.1](#_bookmark13)可知*cnGj*[*i*]({*u*}, {*y*, *z*}) = *cnGj*[*i*]

({*u*}, {*z*}) ≤ 1 且*cnGj*[*i*+1]({*u*}, {*y*, *z*}) ≤ 1(如 图[4-2](#_bookmark49)(a))。 若*NG*−*Gj*[*i*](*u*) ∩ *NG*−*Gj*[*i*](*x*) 芋 ∅， 则(*u*, *x*) ∈ *E*(*G*)。 因 为 该 结 论 与*u* <! *NG*(*x*)矛 盾， 所 以*NG*−*Gj*[*i*](*u*) ∩ *NG*−*Gj*[*i*](*x*) = ∅， 即*cnG*−*Gj*[*i*] ({*u*}, {*x*}) = 0。 类似地，*cnG*−(*Gj*[*i*]∪*Gj*[*i*+1])({*u*}, {*y*, *z*}) = 0。 因此，

*cnG* (*S* 1, *S* 2) = *cnGj*[*i*]({*u*}, {*x*}) + *cnGj*[*i*]({*u*}, {*y*, *z*}) + *cnGj*[*i*+1]({*u*}, {*y*, *z*})

+ *cnG*−*Gj*[*i*]({*u*}, {*x*}) + *cnG*−(*Gj*[*i*]∪*Gj*[*i*+1])({*u*}, {*y*, *z*})

≤ 2 + 1 + 1 + 0 + 0

≤ 4

情形1.2. *x* ∈ *Gj*[*i* + 1]且*y* ∈ *Gj*[*i* + 2]。

根据定义[2.1](#_bookmark13)可知*cnGj*[*i*]({*u*}, {*x*}) = *cnGj*[*i*]({*u*}, {*y*}) ≤ 1，*cnGj*[*i*+1]({*u*}, {*x*}) ≤ 1，*cnGj*[*i*]

({*u*}, {*z*}) ≤ 1且*cnGj*[*i*+2]({*u*}, {*y*, *z*}) ≤ 1(如图[4-2](#_bookmark49)(b))。若*NG*−(*G j*[*i*]∪*G j*[*i*+1])(*u*) ∩*NG*−(*Gj*[*i*]∪*Gj*[*i*+1])(*x*) 芋 ∅，则(*u*, *x*) ∈ *E*(*G*)。 因为该结论与*u* <! *NG*(*x*)矛盾，所以*NG*−(*Gj*[*i*]∪*Gj*[*i*+1])(*u*) ∩*NG*−(*Gj*[*i*]∪ *Gj*[*i*+1]) (*x*) = ∅，即*cnG*−(*Gj*[*i*]∪*Gj*[*i*+1]) ({*u*}, {*x*}) = 0。类似地，*cnG*−(*G j*[*i*]∪*G j*[*i*+2]) ({*u*}, {*y*, *z*}) = 0。

因此，

*cnG*(*S* 1, *S* 2) = *cnGj*[*i*]({*u*}, {*x*}) + *cnGj*[*i*+1]({*u*}, {*x*}) + *cnGj*[*i*]({*u*}, {*z*}) + *cnGj*[*i*+2]({*u*}, {*y*, *z*})

+ *cnG*−(*Gj*[*i*]∪*Gj*[*i*+1])({*u*}, {*x*}) + *cnG*−(*Gj*[*i*]∪*Gj*[*i*+2])({*u*}, {*y*, *z*})

≤ 1 + 1 + 1 + 1 + 0 + 0

≤ 4

情形1.3. *x* ∈ *Gj*[*i* + 1]且*y* ∈ *Gj*[*i*]。

根 据 引 理[4.3](#_bookmark46)可 知*cnGj*[*i*]({*u*}, {*y*, *z*}) ≤ 3。 根 据 定 义[2.1](#_bookmark13)可 知*cnGj*[*i*+1]({*u*}, {*x*}) ≤ 1。 若*NG*−*Gj*[*i*+1](*u*) ∩ *NG*−*Gj*[*i*+1](*x*) 芋 ∅，则(*u*, *x*) ∈ *E*(*G*)。 因为该结论与*u* <t *NG*(*x*)矛盾，

所以*NG*−*Gj*[*i*+1](*u*) ∩ *NG*−*Gj*[*i*+1](*x*) = ∅，即*cnG*−*Gj*[*i*+1]({*u*}, {*x*}) = 0。 类似地，*cnG*−*Gj*[*i*]({*u*},

{*y*, *z*}) = 0。 因此，

*cnG*(*S* 1, *S* 2) = *cnGj*[*i*]({*u*}, {*x*}) + *cnGj*[*i*+1]({*u*}, {*x*}) + *cnGj*[*i*]({*u*}, {*z*})

+ *cnGj*[*i*+2]({*u*}, {*y*, *z*}) + *cnG*−(*Gj*[*i*]∪*Gj*[*i*+1])({*u*}, {*x*})

+ *cnG*−(*Gj*[*i*]∪*Gj*[*i*+2])({*u*}, {*y*, *z*})

≤ 1 + 1 + 1 + 1 + 0 + 0

≤ 4

*u*

*x*

*y*

*z*

*u*

2

*x*

*y*

*z*

*u*

3

*z*

*y*

*x*

*G j* [*i*]

(*a*)

*G j* [*i*+1]

*G j* [*i*]

*G j* [*i*+1]

(*b*)

*G j* [*i*+2]

*G j* [*i*]

(c)

*G j* [*i*+1]

图 4-2 *z*和*y*在一个子图

情形2. *z*和*y*不在同一个子图。

情形2.1. *x* ∈ *Gj*[*i* − 1]，*y* ∈ *Gj*[*i*]且*y* ∈ *Gj*[*i* + 1]。

根据定义[2.1](#_bookmark13)可知*cnGj*[*i*]({*u*}, {*y*}) ≤ 2，*cnGj*[*i*−1]({*u*}, {*x*}) ≤ 1，且*cnGj*[*i*+1]({*u*}, {*z*}) ≤ 1(如 图[4-3](#_bookmark50)(a))。 若*NG*−*Gj*[*i*](*u*) ∩ *NG*−*Gj*[*i*](*y*) 芋 ∅， 则(*u*, *y*) ∈ *E*(*G*)。 因 为 该 结 论 与*u* <! *NG*(*y*)矛盾，所以*NG*−*Gj*[*i*+1](*u*) ∩ *NG*−*Gj*[*i*+1](*y*) = ∅，即*cnG*−*Gj*[*i*]({*u*}, {*y*}) = 0。 类似地，可 以推出*cnG*−*Gj*[*i*−1]({*u*}, {*x*}) = 0 且*cnG*−*Gj*[*i*+1]({*u*}, {*z*}) = 0。 因此，

*cnG*(*S* 1, *S* 2) =*cnGj*[*i*]({*u*}, {*y*}) + *cnG*−(*Gj*[*i*]({*u*}, {*y*})

+ *cnGj*[*i*−1]({*u*}, {*x*}) + *cnG*−(*Gj*[*i*−1]({*u*}, {*x*})

+ *cnGj*[*i*+1]({*u*}, {*z*}) + *cnG*−(*Gj*[*i*+1]({*u*}, {*z*})

≤ 2 + 0 + 1 + 0 + 1

≤4

情形2.2. *x* ∈ *Gj*[*i* + 1]，*y* ∈ *Gj*[*i* + 2]且*y* ∈ *Gj*[*i* + 3]。

需要声明的是，该情况发生于*mj* ≥ 4。 根据定义[2.1](#_bookmark13)可知*cnGj*[*i*+1]({*u*}, {*x*, *y*, *z*}) ≤ 1，*cnGj*[*i*+1]({*u*}, {*x*}) ≤ 1，*cnGj*[*i*+2]({*u*}, {*y*}) ≤ 1且 *cnGj*[*i*+3]({*u*}, {*z*}) ≤ 1(如图[4-3](#_bookmark50)(b)所示)。 若*NG*−(*Gj*[*i*]∪*Gj*[*i*+1]) (*u*)∩ *NG*−(*Gj*[*i*]∪*Gj*[*i*+1]) (*x*) 芋 ∅， 则(*u*, *x*) ∈ *E*(*G*)。 因 为 该 结 论 与*u* <!

*NG*(*x*)矛盾，所以*NG*−(*G j*[*i*]∪*G j*[*i*+1]) (*u*)∩*NG*−(*Gj*[*i*]∪*Gj*[*i*+1])(*x*) = ∅，即*cnG*−(*G j*[*i*]∪*G j*[*i*+1])({*u*}, {*x*}) =

0。 类似地，可以推出*cnG*−(*Gj*[*i*]∪*Gj*[*i*+2]) ({*u*}, {*y*}) = 0 且*cnG*−(*Gj*[*i*]∪*Gj*[*i*+2])({*u*}, {*z*}) = 0。 因

此，

*cnG*(*S* 1, *S* 2) = *cnGj*[*i*+1]({*u*}, {*x*, *y*, *z*}) + *cnGj*[*i*+1]({*u*}, {*x*})

+ *cnG*−(*Gj*[*i*]∪*Gj*[*i*+1])({*u*}, {*x*}) + *cnGj*[*i*+2]({*u*}, {*y*})

+ *cnG*−(*Gj*[*i*]∪*Gj*[*i*+2])({*u*}, {*y*}) + *cnGj*[*i*+3]({*u*}, {*z*})

+ *cnG*−(*Gj*[*i*]∪*Gj*[*i*+3])({*u*}, {*z*})

≤ 1 + 1 + 0 + 1 + 0 + 1 + 0

≤ 4

综上所述，*cnG*(*S* 1, *S* 2) ≤ 4。 因此，本引理成立。 口

*u*

2

*x*

*y*

*z*

*u*

*x*

*y*

*z*

*G j* [*i* 一1]

*G j* [*i*]

( *a* )

*G j* [*i*+1]

*G j* [*i*]

*G j* [*i*+1]

( *b* )

*G j* [*i*+2]

*G j* [*i*+3]

图 4-3 *z*和*y*不在一个子图

引理 **4.5.** 令*S* 1和*S* 2为*G*(*mr*, *mr*−1, . . . , *m*1)的子图，满足*V*(*S* 1)∩*V*(*S* 2) = ∅ 且*S* 1 <t *NG*(*S* 2)， 其 中*r* ≥ 2且 对 于 任 意 一 个*i* ∈ ⟨*r*⟩ \ {0}， 都 有*mi* ≥ 3。 若*S* 1 三 *K*2且*S* 2 三 *P*3， 则*cnG*(*S* 1, *S* 2) ≤ 8。

证明**.** 不失一般性，假设*V*(*S* 1) = {*u*, *v*}，*V*(*S* 2) = {*x*, *y*, *z*}且(*u*, *v*), (*x*, *y*), (*y*, *z*) ∈ *E*(*G*)。根 据引理[4.4](#_bookmark48)可知，*cnG*(*S* 1, *S* 2) ≤ *cnG*({*u*}, *S* 2) + *cnG*({*v*}, *S* 2) = 4 + 4 = 8。 口

**4.1.3** 广广义义义超超超立立立方方方体体体的的2**-**额额外外外连连连通通通度度

引理 **4.6.** 令*F*为广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)中的一个故障顶点集合，若*G*和*F*满 足：

*(a)* 若*mr* > *mr*−1，则|*F*| ≤ 3κ(*G*) − 2*mr* − 1；

*(b)* 若*mr* = *mr*−1，则|*F*| ≤ 3κ(*G*) − 2*mr* − 2；

*(c) G* − *F*中的每个分支中都包含至少3个顶点。

则对于每个 *j* ∈ ⟨*mk* − 1⟩，都有子图*Gk*[ *j*] − *F j*中的每个顶点可以连通到子图*Gk*[*J*] − *FJ* ， 其中*J* = ⟨*mk* − 1⟩ \ { *j*}。

证明**.** 令*u* ∈ *V*(*Gk*[ *j*] − *F j*)。 若*NGk* [*J*](*u*) cJ *F*，则本定理得 证。 若*NGk* [*J*](*u*) cJ *F*，因 为*G* − *F*中不存在孤立点，所以存在一个顶点*v* ∈ *NGk* [ *j*](*u*) − *F j*。 若*NGk* [*J*](*v*) cJ *FJ* ， 则本定理得 证。 若*NGk* [*J*](*v*) ⊆ *FJ* ，因为*G* − *F*中不存在孤立边，所以存在一个顶 点*w* ∈ *NGk* [ *j*]({*u*, *v*}) − *F j*。 若*NGk* [*J*](*w*) cJ *FJ* ，则本定理得证。 若*NGk* [*J*](*w*) ⊆ *FJ* ，因为顶

点*w*可能和顶点*u*或*v*相邻，所以不失一般性，假设*w*和*v*相邻。 我们考虑以下两种情

形。

情形1. *mr* > *mr*−1。

根据引理 [4.1](#_bookmark44)可知|*NGk* [ *j*]({*u*, *v*, *w*})| ≥ 3κ(*G*) − 3*mk* − 2*mr* + 3。令*N* = *NGk* [ *j*]({*u*, *v*, *w*}) ∪ *NGk* [*J*]{*u*, *v*, *w*}，则|*N*| ≥ (3κ(*G*) − 3*mk* − 2*mr* + 3) + 3(*mk* − 1) = 3κ(*G*) − 2*mr*。 若*N*中的任 何一个顶点都无法和子图*Gk*[*J*]连通。 则可以推出3κ(*G*) − 2*mr* − 1 ≥ |*F*| ≥ |*N*|，该结论 与|*N*| ≥ 3κ(*G*) − 2*mr* > |*F*|矛盾。 因此，*N*中必定存在一个顶点和子图*Gk*[*J*]连通。

情形2. *mr* = *mr*−1。

类 似 地， 根 据 引 理 [4.1](#_bookmark44)可 知|*NGk* [ *j*]({*u*, *v*, *w*})| ≥ 3κ(*G*) − 3*mk* − 2*mr* + 2。 令*N* = *NGk* [ *j*]({*u*, *v*, *w*})∪*NGk* [*J*]{*u*, *v*, *w*}，则|*N*| ≥ (3κ(*G*)−3*mk* −2*mr* +2)+3(*mk* −1) = 3κ(*G*)−2*mr* −1。 若*N*中的任何一个顶点都无法和子图*Gk*[*J*]连通。 则3κ(*G*) − 2*mr* − 2 ≥ |*F*| ≥ |*N*|，该结 论与|*N*| ≥ 3κ(*G*) − 2*mr* − 1 > |*F*|矛盾。 因此，*N*中必定存在一个顶点和子图*Gk*[*J*]连

通。

综上所述，子图*Gk*[ *j*] − *F j*中任意一个顶点*u*可以和子图*Gk*[*J*] − *FJ* 连通。 因此，

本引理得证。 口

引理 **4.7.** 令*Gk* [α]和*Gk* [β]为*G*(*mr*, *mr*

−1, . . . , *m*1)中的两个子图，且*F*α

和*F*β分别为*Gk*

[α]和*Gk* [β]两个子图中的故障集合，其中|*F*α| ≤ κ1(*Gk*[α]) |*F*β| ≤ κ1(*Gk*[β])，且满足：

*(a)* 当*mr* > *mr*−1时，|*F*α| ∪ *F*β| ≤ 3κ(*G*) − 3*mk* − 2*mr* + 5

*(b)* 当*mr* = *mr*−1时，|*F*α| ∪ *F*β| ≤ 3κ(*G*) − 3*mk* − 2*mr* + 4

若*Gk*[α] − *F*α和*Gk*[β] − *F*β中的每个顶点都至少有一个无故障邻居，则*Gk*[α] − *F*α中的 任意一个顶点可以和*Gk*[β] − *F*β中的某些顶点相连。

证明**.** 令*u* ∈ *V*(*Gk*[α] − *F*α)。 若*NGk* [β](*u*) cJ *F*β，则本定理得证。 若*NGk* [β](*u*) ⊆ *F*β，因

为*u*在*Gk*[α] − *F*α中不是一个孤立点，所以存在一个顶点*v* ⊂ *NGk* [α]

*F* (*u*)。 若*N k*

(*v*) cJ

− α *G* [β]

*F*β， 则本 引 理得 证。 若*NGk* [β](*v*) ⊆ *F*β， 因 为|*F*α| ≤ κ1(*Gk*[α])， 且(*u*, *v*)不是 子 图*Gk*

[α]− *F*α一条孤立边，所以*Gk*[α] − *F*α中必定存在一个无故障顶点*w* ∈ *NGk* [α] (*u*, *v*)。 若*NGk* [β](*w*) cJ *F*β，则本引理得证。 若*NGk* [β](*w*) ⊆ *F*β，我们讨论以下两种情形。

情形1. *mr* > *mr*−1。

根据引理 [4.1](#_bookmark44)可知|*NGk* [ *j*]({*u*, *v*, *w*})| ≥ 3κ(*G*) − 3*m*1 − 2*mr* + 3。 令*N* = *NGk* [α](*u*, *v*, *w*) ∪ *NGk* [β](*u*, *v*, *w*)。则|*N*| ≥ (3κ(*G*) − 3*mk* − 2*mr* + 3) + 3 = 3κ(*G*) − 3*mk* − 2*mr* + 6。若*N*中的任 意一个顶点都无法连接到*Gk*[β] − *F*β，则|*F*α ∪ *F*β| ≥ |*N*| ≥ 3κ(*G*) − 3*mk* − 2*mr* + 6，该结 论与|*F*α ∪ *F*β| ≤ 3κ(*G*) − 3*mk* − 2*mr* + 5矛盾。因此，*N*中必存在一个顶点和*Gk*[β] − *F*β连

通，则本引理得证。

情形2. *mr* = *mr*−1。

类 似 地， 根 据 引 理 [4.1](#_bookmark44)可 知|*NGk* [ *j*]({*u*, *v*, *w*})| ≥ 3κ(*G*) − 3*m*1 − 2*mr* + 2。 令*N* = *NGk* [α](*u*, *v*, *w*) ∪ *NGk* [β](*u*, *v*, *w*)，则|*N*| ≥ (3κ(*G*) − 3*mk* − 2*mr* + 2)+ 3 = 3κ(*G*) − 3*mk* − 2*mr* + 5。 若*N*中的任意一个顶点都无法连接到*Gk*[β]−*F*β。则|*F*α∪*F*β| ≥ |*N*| ≥ 3κ(*G*)−3*mk*−2*mr* +4， 该结论与|*F*α ∪ *F*β| ≤ 3κ(*G*) − 3*mk* − 2*mr* + 5矛 盾。 因此， *N*中必定存在一个顶点 和*Gk*[β] − *F*β连通，则本引理得证。 口

基于引理 [4.6](#_bookmark53)和引理 [4.7](#_bookmark54)，我们可以推出定理[4.1](#_bookmark55)。

定理 **4.1.** 对于*G*(*mr*, *mr*−1, . . . , *m*1)，其中*r* ≥ 5，且对于所有的*i* ∈ ⟨*r*⟩ \ {0}，都有*mi* ≥ 3， 则

 3κ(*G*) − 2*mr*, 若*mr* > *mr*−1

2 ≥

κ (*G*) 

 3κ(*G*) − 2*mr* − 1, *mr* = *mr*−1

若

证明**.** 令*F* ⊂ *V*(*G*)为一个故障顶点集合，且满足： (a) 当*mr* > *mr*−1时，|*F*| ≤ 3κ(*G*) − 2*mr* − 1

(b) 当*mr* = *mr*−1时，|*F*| ≤ 3κ(*G*) − 2*mr* − 2 (c) *G* − *F*满足每个分支至少包含3个顶点

证明本定理，等价于证明对于*G* − *F*中任意两个顶点*u*和*v* 都存在一条从*x*到*v*的无

故障路径。

若*u*和*v*相邻，则本定理显然成立。 因此，我们讨论*u*和*v*不相邻的情况。

3κ(*G*)−2*mr* −1

因为若*mr* > *mr*−1且|*F*| ≤ 3κ(*G*) − 2*mr* − 1，则

κ(*G*)−1 ≤ 3；若*mr* = *mr*−1且|*F*| ≤

3κ(*G*) 2*m* 2，则 3κ(*G*)−2*mr* −2

κ(*G*)−1

− *r* −

≤ 3。 因此，*G* − *F*中至多存在3个度为1的顶点。 若存

在3个度为1的顶点，不失一般性，假设3个度为1的顶点的关联边分别为第*i*, *j*, *l*维的

边，其中{*i*, *j*, *l*} ⊂ ⟨*r*⟩ \ {0}。 因为*r* ≥ 5，所以必定存在一个维度*k*使得*k* ⊂ ⟨*r*⟩ \ {0, *i*, *j*, *l*}

且*mk* = min{{*m*1, *m*2 . . . , *mr*} \ {*mi*, *mj*, *ml*}}。类似地，若存在小于3个度为1的顶点时，必

定存在这样的维度*k*。 因此，我们从第*k*个维度将*G*切分为*mk*个子图*Gk*[0], *Gk*[1], . . . ,

*Gk* [*mk* − 1]，显然，每个子图*Gk*[ *j*]中不存在孤立点，其中 *j* ⊂ ⟨*mk* − 1⟩。

令*u* ∈ *Gr*[α]且*v* ∈ *Gr*[β], 其中α, β ∈ ⟨*mr* −1⟩。令*F*α = *V*(*Gr*[α])∩*F* 且*F*β = *V*(*Gr*[β])∩

*F*。 根据*u*和*v*所在的子图的不同，我们讨论以下两种情形。

情形1. α 芋 β。

情形1.1. |*F*α| ≥ κ1(*Gk*[α])或|*F*β| ≥ κ1(*Gk*[β])。

不失一般性，假设|*F*α| ≥ κ(*Gr*[α]) = 2κ(*G*)−2*mk* −*mr* +2。令*J* = ⟨*mr* −1⟩\{α}，*FJ* =

∪ *j*∈*J F j*且*G* [*J*]为由*V*(*G*)\*V*(*G* [α])导出的子图。根据引理 [4.6](#_bookmark53)可知，*G*中必定存在一条

*r r*

从*u*到子图*Gk*[*J*]中顶点*u*′的无故障路径*P*1。若*mr* > *mr*

−1，则|*FJ*

| = |*F* − *F*α

| ≤ (3κ−2*mr* −

1)−(2κ(*G*)−2*mk* −*mr* +2) = κ−*mr* +2*mk* −3。因为κ1(*Gk*[*J*]) = 2(κ−1)−*mr* = 2κ(*G*)−*mr* −2，

所以κ1(*Gk*[*J*])−|*FJ* | = κ−2*mk* +1。显然κ−2*mk* +1 > 0。因此，κ1(*Gk*[*J*]) > |*FJ* |。类似地， 若*mr* = *mr*−1，则|*FJ* | = |*F* − *F*α| ≤ (3κ− 2*mr* − 2) −(2κ(*G*) − 2*mk* − *mr* + 2) = κ− *mr* + 2*mk* − 4。 因为κ1(*Gk*[*J*]) = 2(κ − 1) − *mr* = 2κ(*G*) − *mr* − 2，所以κ1(*Gk*[*J*]) − |*FJ* | = κ − 2*mk* + 1。 显

然，κ − 2*mk* > 0。 因此，κ1(*Gk*[*J*]) > |*FJ* |。 综合上述*mr* > *mr*

−1和*mr*

= *mr*−1

的两个情况

的讨论可知κ1(*Gk*[*J*]) > |*FJ* |。 因为*Gk*[*J*]中每个无故障顶点都至少有一个无故障邻居，

根据定理[3.3](#_bookmark27) 可知*Gr*[*J*] − *FJ* 是连通的，也就是说*Gr*[*J*] − *FJ* 中必定存在一条从*u*′到*v*的 无故障路径*P*2。 因此，路径(*P*1, *P*2)是*u*和*v*在*G* − *F*中的一条无故障路径(如图[4-4](#_bookmark56))。



*F* － *F*

*u* '

*P*2

*v*

*F*

*u*

*P*1

*Gk* [**]

*G* －*Gk* [**]

图 4-4 情形|*F*α| ≥ κ1(*Gk* [α])或|*F*β| ≥ κ1(*Gk* [β])

情形1.2. |*F*α| < κ1(*Gk*[α]) 且|*F*β| < κ1(*Gk*[β])。

因为|*F*β| < κ1(*Gk*[β])且子图*Gk*[β]− *F*β中不存在孤立点和孤立边，根据定理[3.3](#_bookmark27)可知 子图*Gk*[β] − *F*β是连通的。若*NGk* [β](*u*) cJ *F*，因为*Gr*[β] − *F*β是连通的，所以子图*Gk*[β] − *F*β中必定存在一条从*x*到*v*的无故障路径*P*。 因此，路径(*x*, *P*)即为*u*和*v*在*G* − *F*中的一 条无故障路径。 若*NGk* [β](*u*) ⊆ *F*，我们讨论以下两种情况。

情形1.2.1. (*mr* > *mr*−1且|*F*α ∪ *F*β| ≤ 3κ(*G*) − 2*mk* − 2*mr* + 5) 或(*mr* = *mr*−1且|*F*α ∪ *F*β| ≤

3κ(*G*) − 2*mk* − 2*mr* + 4)。

因为*Gk*[α]中无孤立点，所以*u*在*Gk*[α]中至少有一个无故障邻居*y*。 根据引理

[4.7](#_bookmark54)可知，*G*中必定存在一条从*u*到子图*Gk*[β]顶点*v*的无故障路径*P*。 若*z* = *v*，那么*P*即

为*u*和*v*在*G* − *F*中的一条无故障路径。 若*z* 芋 *v*，因为*Gr*[β]是连通的，且*z*和*v*都在子 图*Gr*[β]中，所以*Gr*[β]中必定存在一条从*z*到*v*的无故障路径*P*′。 因此，路径(*P*, *P*′)即 为*u*和*v*在*G* − *F*中的一条无故障路径(如图[4-5](#_bookmark57))。

情形1.2.2. (*mr* > *mr*−1且|*F*α ∪ *F*β| ≥ 3κ(*G*) − 2*mk* − 2*mr* + 6) 或(*mr* = *mr*−1且|*F*α ∪ *F*β| ≥

3κ(*G*) − 2*mk* − 2*mr* + 5)。

若*mr* > *mr*−1，则|*F* \(*F*α ∪ *F*β)| ≤ (3κ(*G*) − 2*mr* − 1) −(3κ(*G*) − 2*mr* − 2*mr* + 6) = 3*mk* − 7； 若*mr* = *mr*−1，则|*F* \ (*F*α ∪ *F*β)| ≤ (3κ(*G*) − 2*mr* − 2) − (3κ(*G*) − 2*mr* − 2*mr* + 5) = 3*mk* − 7。 令*G* \(*Gk*[α]∪*Gk*[β])中的每个子图为*Gk*[*i*]，其中*i* ∈ ⟨*mk*⟩\{0, α, β}, 则共有*mk* − 2个子图。 若每个子图中的总故障顶点数都大于等于3，则|*F* \ (*F*α ∪ *F*β)| ≥ 3(*mk* − 2) = 3*mk* − 6 > 3*mk* − 7，该结论与|*F* \ (*F*α ∪ *F*β)| ≤ 3*mk* − 7矛盾。因此，*G* − *F*中必定存在一个子图的



*F*

*z*

*P*2

*v*

*F*

*u*

*P*1

*Gk* [**]

*Gk* [** ]

图 4-5 定理[4.1](#_bookmark55)的情形1.2.1

总故障顶点数小于等于2。 不失一般性，假设子图*Gr*[γ]的总故障顶点数小于等于2，

其中γ ∈ ⟨*mr* − 1⟩ \ {α, β}。 因为子图*Gk*[β] − *F*β中不存在孤立点和孤立边，所以必定

存在顶点*u*1和*u*2使得(*u*, *u*1) ∈ *E*(*G*)且(*u*1, *u*2) ∈ *E*(*G*)。 令*NGk* [γ]({*u*, *u*1, *u*2}) = {*u*′, *u*′ , *u*′ }。

1 2

因为|*F*γ| ≤ 2，所以{*u*′, *u*′ , *u*′ }中必定存在一个顶点是无故障的。 因此，*G*中必定存

1 2

在以*u*为起点且以{*u*′, *u*′ , *u*′ }中的某个无故障顶点为终点的无故障路径为*P*1。 类似

1 2

地，在*G*中必定存在以*v*为起点且以{*v*′, *v*′ , *v*′ }中的某个无故障顶点为终点的无故障

1 2

路径为*P*3。 因为|*F*γ| < 3*mk* − 7 < κ1(*Gk*[γ])，且*Gk*[γ] − *F*γ中每个顶点都有一个无故 障邻居，根据定义[3.3](#_bookmark27)可知*Gk*[γ]是连通的，也就是说，在*Gk*[γ] − *F*γ中必定存在一条 从*u*′到*v*′的无故障路径*P*2。 因此，路径(*P*1, *P*2, *P*−1)是*u*和*v*在*G* − *F*中的一条无故障路

3

径(如图[4-6](#_bookmark58))。

情形2. α = β。

因为*Gr*[α] − *F*α中无孤立点，所以*Gr*[α] − *F*α中的每个顶点都至少有一个无故障 顶点，若|*F*α| < κ1(*Gr*[α])，则根据定义[3.3](#_bookmark27)可知*Gr*[α] − *F*α是连通的，本定理得证。 因 此，我们讨论|*F*α| ≥ κ1(*Gk*[α]) = 2κ(*G*) − 2*mk* − *mr* + 2的情况。

令*J* = ⟨*mr* − 1⟩ \ {α}且*Gk*[*J*]由*V*(*G*) − *Gk*[α]导出的子图。根据引理 [4.6](#_bookmark53)可知，在*G* −

*F*中必定存在一条从*u*到*Gk*[*J*]中顶点*u*′为终点的路径*P*1。类似地，可以推出在*G* − *F*中

必定存在一条从*v*到*Gk*[*J*]中顶点*v*′为终点的路径*P*3。 若*mr* > *mr*

−1，则|*FJ*

| = |*F* − *F*α| ≤



*F*

*F*

*F*

*u*

*P*1

*u*1

*u* '

*u*1 '

*u* '

2

*u*2

*v*2

*v*2 '

*P*

2

*v*1

*v* '

1

*v*

*P*

3

*v* '

*Gk* [**]

*Gk* [** ]

*Gk* [**]

图 4-6 定理[4.1](#_bookmark55)的情形1.2.2

(3κ− 2*mr* − 1) −(2κ(*G*) − 2*mk* − *mr* + 2) = κ− *mr* + 2*mk* − 3且κ1(*Gk*[*J*]) −|*FJ* | = κ− 2*mk* + 1 > 0。

因此，κ1(*Gk*[*J*]) > |*FJ* |。 若*mr* = *mr*

−1，类似地，可以推出|*FJ*

| = |*F* − *F*α

| ≤ (3κ − 2*mr* −

2) − (2κ(*G*) − 2*mk* − *mr* + 2) = κ − *mr* + 2*mk* − 4且κ1(*Gk*[*J*]) − |*FJ* | = κ − 2*mk* > 0。 因此，

κ1(*Gk*[*J*]) > |*FJ* |。 综合上述*mr* > *mr*

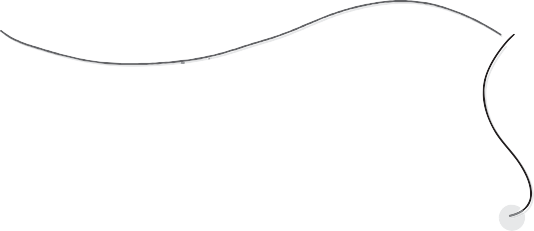
−1和*mr*

= *mr*−1

两个情形的讨论可知κ1(*Gk*[*J*]) > |*FJ* |。

因为*Gk*[*J*]中每个顶点都至少有一个无故障邻居，根据定义[3.3](#_bookmark27)可知*Gr*[*J*] − *FJ* 是连

通的，也就是说在*Gr*[*J*] − *FJ* 中必定存在一条从*u*′到*v*′的无故障路径*P*2。 因此，路 径(*P*1, *P*2, *P*3)即为*u*和*v*在*G* − *F*中的一条无故障路径(如图[4-7](#_bookmark59))。



*F* \ *F*

*P*1

*u*

*u* '

*P*2

*F*

*v*

**

*P*3

*v* '

*Gk* [**]

*G* *Gk* [**]

图 4-7 定理[4.1](#_bookmark55)的情形：α = β且|*F*α| ≥ κ1(*Gk* [α])

综上所述，因为对于*G* − *F*中任意两个不同的顶点*u*和*v*，必定存在一条从*u*到*v*的 无故障路径，所以*G* − *F*是连通的。 因此，本定理得证。 口

定理 **4.2.** 对于*G*(*mr*, *mr*−1, . . . , *m*1)，其中*r* ≥ 5，且对于所有的*i* ∈ ⟨*r*⟩ \ {0}，都有*mi* ≥ 3，

则：

 3κ(*G*) − 2*mr*, 若*mr* > *mr*−1

2 ≤

κ (*G*) 

 3κ(*G*) − 2*mr* − 1, *mr* = *mr*−1

若

证明**.** 令*u* = *urur*−1 . . . *u*1, *v* = *vrvr*−1 . . . *v*1, *w* = *wrwr*−1 . . . *w*1 ∈ *V*(*G*) 其中{(*u*, *v*), (*v*, *w*), (*v*, *w*)} ⊂ *E*(*G*)，且对于任意*i* ∈ ⟨*r*⟩ \ {0, *r*}，都有*ui* = *vi* = *wi*且*ur* 芋 *vr* 芋 *wr*。 根据 定义[2.1](#_bookmark13)可知，若*mr* > *mr*−1，则|*NG*(*u*, *v*, *w*)| = 3(κ − (*mr* − 1)) + (*mr* − 3) = 3κ − 2*mr*。 若*mr* = *mr*−1，则|*NG* (*u*, *v*, *w*)| = 3κ − 2*mr* − 1。

令*F* = *NG*({*u*, *v*, *w*})，显然*G* − *F*分为两个子图，其中一个是包含*u*, *v*和*w*三个顶点 的子图，另一个是子图*G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})。 显然，*G* − *F*是不连通的。 接下 来，我们证明*G* − *F*中的每个分支都至少有3个顶点。

断言： *G* − *F*中的每个分支都至少有3个顶点。

已知*G* − *F*分为两个子图，其中一个是包含*u*, *v*和*w*三个顶点的子图，另一个是 子图*G* − (*NG* ({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})。 显然对于前者，该断言成立。 因此，我们讨论子 图*G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})的情形。 若*G* − (*NG* ({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})中包含顶点个数

小于3的分支，则其包含孤立点或孤立边。 因此，我们考虑以下两种情况。 情形1. *G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})中包含孤立点。

不失一般性，假设*x*为孤立点。显然，*NG*({*x*}) ⊂ *F*。因此，*NG*({*x*}) = *NG* ({*x*})∩*F* = *NG*({*x*}) ∩ *NG*({*u*, *v*, *w*})。 根据引理[4.4](#_bookmark48)可知*NG*({*x*}) = |*NG*({*x*}) ∩ *NG*({*u*, *v*, *w*}| ≤ 4。 根据 定义[2.1](#_bookmark13)可知|*NG*({*x*})| = κ(*G*) ≥ 10。 显然， *NG*({*x*}) ≤ 4与|*NG*({*x*})| ≥ 10矛 盾。 因此， *G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})中不包含孤立点。

情形2. *G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})中包含孤立边。 不失一般性，假设(*x*, *y*)为孤立 边。 显然， *NG*({*x*, *y*}) ⊂ *F*。 因此， *NG*({*x*, *y*}) =

*NG*({*x*, *y*})∩ *F* = *NG*({*x*, *y*})∩ *NG*({*u*, *v*, *w*})。根据引理[4.5](#_bookmark51)可知|*NG*({*x*})| = |*NG*({*x*, *y*})∩ *NG*({*u*,

*v*, *w*}| ≤ 8，根据定义[2.1](#_bookmark13)可知|*NG*({*x*})| = κ(*G*) ≥ 17。 显然，|*NG*({*x*})| ≤ 8与|*NG*({*x*})| ≥

17矛盾。 因此，*G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})中不包含孤立边。 综合情形1和情形2的讨论可知*G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})无孤立点和孤立边。因

此，*G* − (*NG*({*u*, *v*, *w*}) ∪ {*u*, *v*, *w*})包含至少3个顶点，断言得证。

综上所述，当*mr* > *mr*−1且|*F*| = |*NG*({*u*, *v*, *w*})| = 3κ − 2*mr* 时；或当*mr* = *mr*−1且|*F*| =

|*NG*({*u*, *v*, *w*})| = 3κ − 2*mr* − 1时，*G* − *F*是不连通的，且每个分支至少包含3个无故障顶

点。 因此，本定理得证。 口

综合定理[3.1](#_bookmark22)和定理[4.3](#_bookmark60)，我们可以得到定理[4.3](#_bookmark60)。

定理 **4.3.** 对于*G*(*mr*, *mr*−1, . . . , *m*1)，若*r* ≥ 5，且对于所有的*i* ∈ ⟨*r*⟩\{0}，都有*mi* ≥ 3。那 么，当*mr* > *mr*−1时，有κ2(*G*) = 3κ(*G*)−2*mr*；当*mr* = *mr*−1时，有κ2(*G*) = 3κ(*G*)−2*mr* −1。

### 4.2 广义超立方体的2-额外连通度下的容错路由

我们在第三章给出的GHCFP可以在*G*(*mr*, *mr*−1, . . . , *m*1)的总故障顶点数小于其1-

限制连通度，且每个无故障顶点都至少有一个无故障邻居时，得到*G*中任意两个无

故障顶点之间的一条无故障路径[2](#_bookmark30)。 在本小节中，我们给出当*G*的总故障顶点数小于其2-额外连通度，且除去故障顶

点后的每个分支中至少包含3个无故障顶点条件下的容错路由算法GHCFP2。 之后，

我们会分析该算法的时间复杂度。

**4.2.1** 2**-**额额外外外连连连通通通度度度条条条件件件下下下的的的容容容错错错路路路由由由算算算法法

算法 **3** GHCFP2(*r*, *u*, *v*, *F*, *G*(*mr* , *mr*−1 , . . . , *m*1 ))

输入**:** *r*-维广义超立方体*G*(*mr* , *mr*−1, . . . , *m*1)， 故障顶点集合*F* ⊂ *V*(*G*)， 两个不同的顶点*u* =

*urur*−1 . . . *u*1且*v* = *vrvr*−1 . . . *v*1。

输出**:** 一条在*G* − *F*中以*u*起点为且*v*为终点的无故障路径。

1: **function** GHCFP2(*r*, *u*, *v*, *F*, *G*(*mr* , *mr*−1, . . . , *m*1))

2: *G* ← *G*(*mr* , *m*−1, . . . , *m*1);

3: **if** *F* = ∅ **then return** GHCP(*r*, *u*, *v*, *G*, *F*);

4: **else if** (*u*, *v*) ∈ *E*(*G*) **then return** (*u*, *v*);

5: **end if**

6: *k* ← GetDimension(*G*);

7: *k*1 ← ∑*r*

*i*=0

(*mi* − 1), κ2 ← 2κ − *mr* , α ← *uk* , β ← *vk* ;

8: *F*α ← *F* ∩ *V*(*Gk* [α]), *F*β ← *F* ∩ *V*(*Gk* [β]);

9: **if** α 芋 β **then**

10: **if** |*F*α| ≥ *k*2 **then**

11: *P*1 ← PATHSEQ (*u*, *F*, *Gk* [α], *G* − *V*(*Gk* [α]));

12: 令*z*为*P*1的最后一个顶点;

13: **if** *z* = *v* **then return** *P*1;

14: **end if**

15: **return** (*P*1, GHCFP(*r*, *z*, *v*, *G* − *V*(*Gk* [α]), *F* \ *F*α));

16: **else if** |*F*β| ≥ *k*2 **then**

17: *P*1 ← PATHSEQ (*v*, *F*, *Gk* [β], *G* − *V*(*Gk* [β]));

18: 令*z*为*P*1的最后一个顶点;

19: **if** *z* = *v* **then return** *P*1;

20: **end if**

21: **return** (GHCFP(*r*, *u*, *z*, *G* − *V*(*Gk* [β]), *F* \ *F*β), *P*−1);

1

22: **else if** (*mr* > *mr*−1且|*F*α| + |*F*β| ≤ 3*k*1 − 2*mk* − 2*mr* + 5)或(*mr* = *mr*−1 且|*F*α| + |*F*β| ≤ 3*k*1 −

2*mk* − 2*mr* + 4) **then**

23: *P*1 ← PATHSEQ (*u*, *F*α ∪ *F*β, *Gk* [α], *Gk* [β]);

24: 令*z*为*P*1的最后一个顶点;

25: **if** *z* = *v* **then return** *P*1;

26: **end if**

27: **return** (*P*1, GHCFP(*r*, *z*, *v*, *Gk* [β], *F*β));

28: **else**

29: 选取一个子图*Gk* [γ]，使得|*F* ∩ *Gk* [γ]| = min {|*F* ∩ *Gk* [*i*]|}，其中γ, *i* ∈ ⟨*mk* − 1⟩ \ {α, β};

30: *F*γ ← *F* ∩ *V*(*Gk* [γ]);

31: *P*1 ← PATHSEQ (*u*, *F*α ∪ *F*γ, *Gk* [α], *Gk* [γ]);

32: 令*x*为*P*1的最后一个顶点;

33: *P*2 ← PATHSEQ (*v*, *F*β ∪ *F*γ, *Gk* [β], *Gk* [γ]);

34: 令*y*为*P*2的最后一个顶点;

35: **if** *x* = *y* **then return** (*P*1, *P*−1);

2

36: **end if**

37: **return** (*P*1, GHCFP(*r*, *x*, *y*, *G*1, *F* ∩ *V*(*Gk* [γ])), *P*−1);

2

38: **end if**

39: **else if** |*F*α| < *k*2 **then return** (GHCFP(*r*, *u*, *v*, *Gk* [α], *F*α));

40: **else**

41: *P*1 ← PATHSEQ (*u*, *F*, *Gk* [α], *G* − *V*(*Gk* [α]));

42: 令*x*为*P*1的最后一个顶点;

43: *P*2 ← PATHSEQ (*v*, *F*, *Gk* [α], *G* − *V*(*Gk* [α]));

44: 令*y*为*P*2的最后一个顶点;

45: **if** *z*为*P*1和*P*2的第一个公共顶点 **then**

46: **return** (Path(*P*1, *u*, *z*), Path(*P*−1, *z*, *v*));

2

47: **end if**

48: **return** (*P*1,GHCFP(*r*, *x*, *y*, *G* − *V*(*Gr* [α]), *F* \ *F*α), *P*−1);

2

49: **end if**

50: **end function**

51: **function** PATHSEQ(*u*, *F*, *H*1, *H*2)

52: **if** (*NG* (*u*) ∩ *V*(*H*2)) \ *F* 芋 ∅ **then**

53: 从(*NG* (*u*) ∩ *V*(*H*2)) \ *F*中任选一个顶点*u*1;

54: **return** (*u*, *u*1);

55: **else**

56: **for all** *u*1 ∈ (*NG* (*u*) ∩ *V*(*H*1)) \ *F* **do**

57: **if** (*NG* (*u*) ∩ *V*(*H*1)) \ *F* 芋 ∅ **then**

58: 从(*NG* (*u*) ∩ *V*(*H*1)) \ *F* 中*G* − *F*中一个顶点*u*2;

59: **return** (*u*, *u*1, *u*2);

60: **end if**

61: **end for**

62: **end if**

63: 从(*NG* (*u*) ∩ *V*(*H*1)) \ *F*中一个顶点*u*1;

64: *S* ← (*NG* (*u*1) ∩ *V*(*H*1)) \ (*F* ∪ ((*NG* (*u*) ∩ *V*(*H*1)));

65: **for all** *u*2 ∈ *S* **do**

66: **if** (*NG* (*u*2) ∩ *V*(*H*2)) \ *F* = ∅ **then**

67: 从(*NG* (*u*2) ∩ *V*(*H*2)) \ *F*中任选一个顶点*u*3;

68: **return** (*u*, *u*1, *u*2, *u*3);

69: **end if**

70: **end for**

71: 从(*NG* (*u*) ∩ *V*(*H*1)) \ *F*中任选一个顶点*u*1;

72: 从(*NG* (*u*1) ∩ *V*(*H*1)) \ (*F* ∪ ((*NG* (*u*) ∩ *V*(*H*1)))中任选一个顶点*u*2;

73: *S* ← (*NG* (*u*2) ∩ *V*(*H*1)) \ (*F* ∪ (*NG* ({*u*, *u*1}) ∩ *V*(*H*1)));

74: **for all** *u*3 ∈ *S* **do**

75: **if** (*NG* (*u*3) ∩ *V*(*H*2)) \ *F* = ∅ **then**

76: 从(*NG* (*u*3) ∩ *V*(*H*2)) \ *F*中任选一个顶点*u*4;

77: **return** (*u*, *u*1, *u*2, *u*3, *u*4);

78: **end if**

79: **end for**

80: **end function**

81: **function** GETDIMENSION(*G*)

82: *S* ← ∅;

83: **for all** *u* ∈ *V*(*G*) **do**

84: **if** |*NG* (*u*)| = 1 **then**

85: 添加和*u*相关联的边所对应的维度到*S* ;

86: **end if**

87: **end for**

88: **return** min({*m*1, *m*2, ...*mr* } \ *S* );

89: **end function**

**4.2.2** 容容错错错路路路由由由算算算法法法分分分析析

在 分 析 该 算 法 的 时 间 复 杂 度 之 前， 我 们 首 先 分 析 函 数GetDimension(*G*)， PATHSEQ(*u* , *F* , *H*1 , *H*2)和函数GHCFP(*x*, *y*, *G*, *F*)的时间复杂 度。 首先，根据第三 章 的 讨 论 结 果 可 知， GHCFP的 时 间 复 杂 度 为*O*(*k*(*G*)3)。 接 下 来， 我 们 分 析 函

数GetDimension(*G*) 和PATHSEQ(*u* , *F* , *H*1 , *H*2)。

首先，我们分析函数GetDimension(*G*)，该函数最终会返回*G*中的一个维 度。 GetDimension会遍历*G*中的*N*个顶点，每次遍历进行κ(*G*)次判断来得到该顶点的度。 若该顶点的度为1, 则将该顶点相关联的边所对应的维度添加到集合*S* 中，其时

间复杂度为*O*(1)。 因此，该过程的时间复杂度为*N* × κ(*G*)。 然后，GetDimension返 回{*m*1, *m*2, ...*mr*} \ *S* 的最小值，该过程的时间复杂度为*O*(1)。从而，GetDimension的时 间复杂度为*O*(*N* × κ(*G*)) + *O*(1) ≤ *O*(*N* × κ(*G*))。

然 后， 我 们 分 析 函 数PATHSEQ(*u*, *F*, *H*1, *H*2)， 该 函 数 中*H*1和*H*2是*G*的 两 个 子 图 且*u*在*H*1中。 依 据 不同 的 情 况， PATHSEQ可 能会 给 出*u*到子 图*H*2的 路径(*u*, *u*1)、 (*u*, *u*1, *u*2)、(*u*, *u*1, *u*2, *u*3) 或(*u*, *u*1, *u*2, *u*3, *u*4)。 PATHSEQ在不同的情形下的时间复杂度

如下：情形1. 得到的路径为(*u*, *u*1)。 若(*NG*(*u*) ∩ *V*(*H*2)) \ *F* 芋 ∅，则PATHSEQ需 要 从(*NG* (*u*) ∩ *V*(*H*2)) \ *F*中任选一个顶点*u*1，然后返回路径(*u*, *u*1)。 该过程的时间复杂度 为*O*(1)。情形2. 得到的路径为(*u*, *u*1, *u*2)。PATHSEQ首先会遍历集合(*NG*(*u*)∩*V*(*H*1))\*F*， 取 集 合 中 每 个 顶 点 为*u*1， 然 后 从(*NG*(*u*) ∩ *V*(*H*1)) \ *F*中 任 选 一 个 顶 点*u*2， 最 终， PATHSEQ返回路径(*u*, *u*1, *u*2)。该情况的时间复杂度为*O*((*NG*(*u*)∩*V*(*H*1))\*F*) ≤ *O*(κ(*G*))。 情形3. 得到的路径为(*u*, *u*1, *u*2, *u*3)。在该情形下，PATHSEQ首先会从(*NG*(*u*) ∩ *V*(*H*1)) \ *F*中任选一个顶点*u*1 且得到集合*S* ← (*NG*(*u*1) ∩ *V*(*H*1)) \ (*F* ∪ (*NG*(*u*) ∩ *V*(*H*1))。 因此， 该过程的时间复杂度为*O*((*NG* (*u*) ∩ *V*(*H*1)) \ *F*) ≤ *O*(κ(*G*))。 然后，PATHSEQ会遍历集 合*S* ，取集合中每个顶点为*u*2，然后从(*NG*(*u*2) ∩ *V*(*H*2)) \ *F*中任选一个顶点*u*3，该过程 的时间复杂度为*O*(|*S* |) ≤ *O*(κ(*G*))。 最后，PATHSEQ返回路径(*u*, *u*1, *u*2, *u*3)。 因此，该 情况下PATHSEQ的时间复杂度为*O*(κ(*G*)) + *O*(κ(*G*)) ≤ 2 × *O*(κ(*G*))。 情形4. 得到的路 径为(*u*, *u*1, *u*2, *u*3, *u*4)。 在该情形下，PATHSEQ首先会从(*NG*(*u*) ∩ *V*(*H*1)) \ *F* 中任选一 个顶点*u*1 且得到集合*S* ← (*NG*(*u*1) ∩ *V*(*H*1)) \ (*F* ∪ (*NG*(*u*) ∩ *V*(*H*1)))。 然后PATHSEQ会 从*S* 中任选一个顶点*u*2 且得到集合*S* ← (*NG*(*u*2) ∩ *V*(*H*1)) \ (*F* ∪ (*NG*({*u*, *u*1}) ∩ *V*(*H*1)))。 该过程的时间复杂度为2 × *O*((*NG*(*u*) ∩ *V*(*H*1)) \ *F*) ≤ 2 × *O*(κ(*G*))。接下来，PATHSEQ会 遍历集合*S* ，取集合中每个顶点为*u*3，然后从(*NG*(*u*3) ∩ *V*(*H*2)) \ *F*中任选一个顶点*u*4。 该过程的时间复杂度为*O*(|*S* |) ≤ *O*(κ(*G*))。 最后，PATHSEQ返回路径(*u*, *u*1, *u*2, *u*3, *u*4)。 因此，该情况下的时间复杂度为2 × *O*(κ(*G*)) + *O*(κ(*G*) ≤ *O*(κ(*G*))。 综合以上情况可

知PATHSEQ(*u*, *F*, *H*1, *H*2)的时间复杂度为*O*(κ(*G*)2)。

接下来，我们分析算法GHCFP2的时间复杂度。 情形1. α 芋 β。

情形1.1. *F*α ≥ *k*2。 GHCFP2首先会通过PATHSEQ(*u*, *Gk*[α], *G* − *V*(*Gr*[α])) 得到 一条从*u*到*G* − *V*(*Gr*[α])中顶点*z*为终点的路径*P*1，该过程的时间复杂度为*O*(κ(*G*))。 若*z* = *v*，GHCFP2将返回*P*1，那么得到*G* − *F*中一条从*u*到*v*的无故障路径*P*1的时间复 杂度为*O*(κ(*G*)3)；若*z* 芋 *v*，GHCFP2会返回路径(*P*1,GHCFP(*r*, *z*, *v*, *G* −*V*(*Gr*[α]), *F* \ *F*β)。 因为运行GHCFP(*r*, *z*, *v*, *G* − *V*(*Gr*[α], *F* \ *F*β)的时间复杂度为*O*(κ(*G*)3)，且拼接*P*1和上 述GHCFP(*r*, *z*, *v*, *G* − *V*(*Gr*[α]), *F* \ *F*β)所建立的无故障路径的时间复杂度是*O*(1)，所以 得到*G* − *F*中一条从*u*到*v*的路径的时间复杂度为*O*(κ(*G*)3 + κ(*G*)3 + 1) = *O*(κ(*G*)3)。 情形1.2. |*F*β| ≥ *k*2或|*F*α + *F*β| ≤ 2*k*1 − 2*mr* − *n* + 3，类似于在情形1.1中的讨论，我 们在*G* − *F*中得到一条以*u*为起点且*v*为终点的无故障路径的时间复杂度为*O*(κ(*G*)3)。 情形1.3. (*mr* > *mr*−1且|*F*α| + |*F*β| ≤ 3*k*1 − 2*mk* − 2*mr* + 5)或(*mr* = *mr*−1且|*F*α| + |*F*β| ≤

3*k*1 − 2*mk* − 2*mr* + 4)。 首先，GHCFP2选择一个满足条件|*F* ∩ *Gk*[γ]| = 1的子图*Gk*[γ]， 其中*i* ∈ ⟨*mr*⟩ \ {α, β}。 令*G*1 = *Gk*[γ]，因为*mk* − 2 ≤ κ(*G*)，所以得到*G*1的时间复杂度 为*O*(*mk* − 2) ≤ *O*(κ(*G*))。 然后，GHCFP2调用PATHSEQ得到一条从*u*到子图*G*1顶点*x*为

终点的无故障路径*P*1 以及一条从*v*到子图*G*1中的顶点*u*的无故障路径*P*2，得到这两条

路径的时间按复杂度分别为*O*(κ(*G*)2)。 若*x* = *y*，GHCFP2得到一条从*u*到*v*的无故障路

径的时间复杂度为*O*(κ(*G*) + 2 × κ(*G*)2) = *O*(κ(*G*)2)；若*x* 芋 *y*，GHCFP2得到*G* − *F*中一 条从*u*到*v*的无故障路径的时间复杂度为*O*(κ(*G*) + 2 × κ(*G*)2 + κ(*G*)3 + 1) = *O* (κ(*G*)3)。

情形2. α = β。

情形2.1. *F*α ≤ *k*2。 GHCFP2将调用GHCFP(*u*, *v*, *Gk*[α], *F*α)，其时间复杂度为*O*(

κ(*G*)3)。

情形2.2. *F*α ≥ *k*2。 首先，GHCFP2将得到一条从*u*到子图*G* − *V*(*Gk*[α])中顶点*x*的 无故障路径*P*1，以及一条从*v*到子图*G* − *V*(*Gk*[α])中顶点*y*的无故障路径*P*2，该过程 的时间复杂度为*O*(2 × κ(*G*)2)。 然后，若*P*1和*P*2的第一个公共顶点为*z*，GHCFP2返 回路径(Path(*P*1, *u*, *z*), Path(*P*−1, *z*, *v*))。 因为，得到路径(*P*1, *u*, *z*)的时间复杂度为*O*(1)， 所以得到*u*和*v*在*G*中的时间复杂度为*O*(2 × κ(*G*)2) + *O*(1) = *O*(κ(*G*)2)。 若*P*1和*P*2无公 共顶点，算法GHCFP会返回路径(*P*1,GHCFP(*r*, *x*, *y*, *G* − *V*(*Gr*[α]), *F* \ *F*α), *P*−1)。 因为

2

2

得到路径Path(*P*1, *u*, *z*)或Path(*P*−1, *z*, *v*)的时间复杂度为*O*(1)，且调用GHCFP(*r*, *x*, *y*, *G* −

2

*V*(*Gr*[α]), *F* \ *F*α) 的时间复杂度为*O*(κ(*G*)3)。 因此，在*G*中得到一条从*u*到*v*的无故障路 径的时间复杂度为*O*(2 × κ(*G*)2) + *O*(κ(*G*)3) + *O*(2 × 1) = *O*(κ(*G*)3)。

综合以上所有情况，本章给出的GHCFP2的时间复杂度为*O*(*N* × κ(*G*) + κ(*G*)3) ≤

*O*(*N* × κ(*G*))。

**4.2.3** 模模拟拟拟实实实验验

本小节中对GHCFP2用Python语言编程实现，用一台配置为Intel Core i5-7200 CPU 2.71GHz与8 GB 内存的计算机进行模拟实验，并分析实验结果。 类似于第三章 的模拟实验，本实验针对改变维数和改变基数两种情况下，GHCFP2和BFS在执行效 率和所获得路径长度两个方面的表现进行模拟实验。 同时，在改变基数的模拟实验 中，我们基于5-维的广义超立方体进行测试。

GHCFP2给出的是给出当总故障顶点数小于其2-额外连通度且去除故障顶点集 合*F*后的每个分支都至少有3顶点的条件时，*G*(*mr*, *mr*−1, . . . , *m*1)中任意两个无故障顶 点之间的一条无故障路径。 本实验中，我们首先调用*random*函数随机生成*F*，满

足：

(1) *F* ⊂ *V*(*G*)，

(2) |*F*| = κ2(*G*) − 1，

(3) *G* − *F*的每个分支至少存在3个顶点。

然后，我们随机生成一个无故障顶点*x*，然后利用GHCFP2和BFS分别构造出顶

点*x*到*G*中其余所有无故障顶点之间的无故障路由。 将该过程重复执行100次。 我们 针对实验结果进行以下分析。

分析一：分别计算GHCFP2和BFS得到一条无故障路径需要花费的平均时间并进 行对比。 实验结果如图[4-8](#_bookmark65)和图[4-9](#_bookmark66)所示。

从图[4-8](#_bookmark65)中可以看出GHCFP2所花费的平均时间小于BFS。 尽管在维数较小时，

GHCFP2的所花费的时间和BFS所花费的时间相差不大，如图中的3-维*G*(3, 3, 3)和4-

维*G*(3, 3, 3, 3)两种情形的实验结果所 示。 但是，随着维数的增加，BFS花费的时 间增加迅速，GHCFP2执行效率明显比BFS好。 根据实验结果可知，当*r* = 7时， GHCFP2和BFS所花费的平均CPU时间分别为0.4毫秒和16毫秒。 另外，从图[4-9](#_bookmark66)中很 明显可以看出，在维数相同的情况下，随着基数逐渐增大，BFS增长速度要远大

、。

**0.018**



**0.016**

**0.014**

**0.012**

**0.010**

**0.008**

**0.006**

**0.004**

**0.002**

**0**

**G( 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 , 3 , 3 )**

**澻澼澷澺濄澦 澶澺濇**

图 4-8 改变维数的情况下，GHCFP2得到一条无故障路径需要花费的平均时间

、。˅

**0.010**



**0.009**

**0.008**

**0.007**

**0.006**

**0.005**

**0.004**

**0.003**

**0.002**

**0.001**

**0**

**G( 3 , 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 4 ) G( 3 , 3 , 3 , 4 , 4 ) G( 3 , 3 , 4 , 4 , 4 ) G( 3 , 3 , 4 , 4 , 5 ) G( 3 , 3 , 4 , 5 , 6 )**

**澻澼澷澺濄澦 澶澺濇**

图 4-9 改变基数的情况下，BFS得到一条无故障路径需要花费的平均时间

于GHCFP2。 因此，从实验数据可以得到，随着广义超立方体网络规模的扩大，本 文算法的执行效率优于广度优先搜索算法的效率，且网络规模越大，GHCFP2的表 现越好。

分析二：分别计算GHCFP2和BFS所得到的所有路径的平均长度并进行对比，实 验结果如图[4-10](#_bookmark67)和图[4-11](#_bookmark68)所示。

对比图中的数据可以看出，GHCFP2所得到的所有路径的平均长度略大于BFS的 结果。 随着维数的增加或者基数的增大，两者的差值逐渐减小，并接近于0。 从

图[4](#_bookmark68) − [11](#_bookmark68)的数据可以看出，在4-维广义超立方体中，GHCFP2所得到的所有路径的平

均长度接近于BFS。 也就是说，随着网络规模的扩大，GHCFP2所得到的路径长度很

接近最短路径的长度或者等于最短路径的长度。 分析三：分别计算两种算法获得的每一条路径的差值，最后得出所有路径的差

**6**



**5**

ᒣ



൷

䐟 **4**

ᖴ

䮯 **3**

ᓖ

**2**

**1**

**0**

**G(3,3,3) G(3,3,3,3) G(3,3,3,3,3) G(3,3,3,3,3,3) G(3,3,3,3,3,3,3)**

**GHCFP2 BFS**

图 4-10 改变维数的情况下，GHCFP2和BFS所得到的所有路径的平均长度

**4.80**



**4.70**

ᒣ



൷ **4.60**

䐟

ᖴ **4.50**

䮯 **4.40**

ᓖ

**4.30**

**4.20**

**4.10**

**G(3,3,3,3,3) G(3,3,3,3,4) G(3,3,3,4,4) G(3,3,4,4,4) G(3,3,4,4,5) G(3,3,4,5,6)**

**GHCFP2 BFS**

图 4-11 改变基数的情况下，GHCFP2和BFS所得到的所有路径的平均长度

值的最小值、平均值和最大值并进行对比。 实验结果如图[4-12](#_bookmark69)和图[4-13](#_bookmark70)所示。

**4**



**3**

**2**

**1**

**0**

**G( 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 3 , 3 , 3 )**

**枞⠗峰⺅䖅㘁⬐Ἵ 枞⠗峰⺅䖅⵴♈Ἵ 枞⠗峰⺅䖅㘁⠨Ἵ**

图 4-12 改变维数的情况下，GHCFP2和BFS所构造的路径差值的最小值、平均值和最大值

**3**

**2**

**1**

**0**

**G( 3 , 3 , 3 , 3 , 3 ) G( 3 , 3 , 3 , 3 , 4 ) G( 3 , 3 , 3 , 4 , 4 ) G( 3 , 3 , 4 , 4 , 4 ) G( 3 , 3 , 4 , 4 , 5 ) G( 3 , 3 , 4 , 5 , 6 )**

**峰⺅⳯Ἵ䖅㘁⬐Ἵ 峰⺅⳯Ἵ䖅⵴♈Ἵ 峰⺅⳯Ἵ䖅㘁⠨Ἵ**

图 4-13 改变基数的情况下，GHCFP2和BFS所构造的路径差值的最小值、平均值和最大值

从图中可以看出，GHCFP所构造路径长度与BFS所构造路径长度的差值的最小 值始终为0，也就是说GHCFP所得到路径中的部分路径为最短路径。 同时在较低维 度时或者基数较小时，两个算法所构造路径差值的最大值会出现上下为1的波动幅 度，该情况产生的原因是，当网络规模较小时，GHCFP给出的路径情况受随机生成 的*F*的情况影响较大。 随着网络维数的增加，GHCFP2构造路径长度和BFS构造路径 长度的差值平均值和最大值都逐渐减小，也就是GHCFP2所得到的路径长度逐渐接 近于最短路径。 另外，明显可以看出两个算法路径长度差值的平均值接近于最小值， 也就是说GHCFP2所构造的路径中路径长度等于或者接近于最短路径的比例较大。 该实验结果表明，GHCFP2所得到的路径的路径长度整体是较优的。

综合以上分析可得，在执行效率方面，随着网络规模的扩大，GHCFP2相较 于BFS在所花费的平均时间要小。 在所获得的路径长度方面：GHCFP2所得到的路 径的长度中绝大部分接近于最短路径的长度或者等于最短路径的长度。 由此可见， GHCFP2整体表现良好。

### 4.3 本章小结

本章基于额外连通度的定义，我们求出了广义超立方体的1-额外连通度和2-额 外连通度，并给出了满足总故障顶点数小于2-额外连通度，且故障网络中每个分支 至少包含3个顶点条件的容错路由算法。 并取得如下研究成果：

1. *r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1) 的1-额外连通度是2κ(*G*(*mr*, *mr*−1, . . . , *m*1))− *n*，其中*n* = max{*m*1, *m*2, . . . , *mr*}。

2. 对 于*r*-维 广 义 超 立 方 体*G*(*mr*, *mr*−1, . . . , *m*1)， 其 中*r* ≥ 5， 且 对 于 每 个*i* ∈

⟨*r*⟩ \ {0}都有*mi* ≥ 3 且*mi* ≥ *mi*−1。 若*mr* > *mr*−1，则广义超立方体的2-额外连通度 为3κ( *G*(*mr*, *mr*−1, . . . , *m*1 )) −2*mr*；若*mr* = *mr*−1，则广义超立方体的2-额外连通度 为3κ(*G*(*mr*, *mr*−1, . . . , *m*1)) − 2*mr* − 1。

3. 给出了一个2-额外连通度条件下的容错路由算法，该算法的时间复杂度

为*O*(*N* × κ(*G*(*mr*, *mr*−1, . . . , *m*1)))，其中*N*为*G*(*mr*, *mr*−1, . . . , *m*1)的总顶点数。

# 第五章 总结与展望

### 5.1 总结

互连网络的容错性在很大程度上影响着网络的通信性能，因此对于互连网络拓 扑结构容错性的研究一直是高性能计算领域的研究重点。 限制连通度和额外连通度 是衡量网络容错性的重要指标。 本文基于限制连通度和额外连通度，在广义超立方 体上取得如下研究成果：

一、广义超立方体的限制连通度及容错路由问题

我们求出了广义超立方体的1-限制连通度，并给出了满足总故障顶点数小 于1-限制连通度，且故障网络中每个无故障顶点都至少有一个无故障邻居条件的容 错路由算法。 取得如下研究成果：

1. *r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)的1-限制连通度是2κ(*G*) − *n*，其中*n* =

max{*m*1, *m*2, . . . , *mr*}。

2. 给出一个满足1-限制连通度条件的容错路由算法，该算法的时间复杂度

为*O*(κ(*G*)3)。

二、广义超立方体的额外连通度及容错路由问题

我们求出了广义超立方体的1-额外连通度和2-额外连通度，并给出了满足总故 障顶点数小于2-额外连通度，且故障网络中每个分支至少包含3个顶点条件的容错路 由算法。 并取得如下研究成果：

1. *r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)的1-额外连通度是2κ(*mr*, *mr*−1, . . . , *m*1) − *n*， 其中*n* = max{*m*1, *m*2, . . . , *mr*}。

2. 对于*r*-维广义超立方体*G*(*mr*, *mr*−1, . . . , *m*1)，其中*r* ≥ 5，且对于每个*i* ∈ ⟨*r*⟩ \

{0}都有*mi* ≥ 3 且*mi* ≥ *mi*−1。若*mr* > *mr*−1，则广义超立方体的2-额外连通度为3κ (*G*(*mr*,

*mr*−1, . . . , *m*1)) − 2*mr*；若*mr* = *mr*−1，则广义超立方体的2-额外连通度为3κ (*G*(*mr*, *mr*−1,

. . . , *m*1)) −2*mr* − 1。

3. 给出一个满足2-额外连通度条件的容错路由算法，该算法的时间复杂度

为*O*(*N* × κ(*G*(*mr*, *mr*−1, . . . , *m*1)))，其中*N*为*G*(*mr*, *mr*−1, . . . , *m*1)中的总顶点数。

### 5.2 展望

本文在广义超立方体的条件连通度与容错路由问题的研究上取得了一些创新性 结果。 但是，我们的研究仍有不足之处，今后可以做出进一步的研究和探索，主要 包括以下几方面：

(1) 进一步将广义超立方体的1-限制连通度与2-额外连通度扩展到广义超立方体 的*g*-限制连通度和*g*-额外连通度，研究更具一般性的结果。另外，还可以考虑研究超 立方体的其他变型网络上的*g*-限制连通度和*g*-额外连通度，并归纳出这类网络在该 条件下的容错性的共同特性。

(2) 第三章和第四章中提出的容错路由算法在最坏情况下得到的路径长度都约 为*r* + 5，可以更进一步深入研究广义超立方体的网络内在性质，对这些算法进行优 化，以减少算法所获得的最长路径的长度。

(3) 本文第三章和第四章所提出容错路由算法实际上是一个单播算法，我们将进 一步研究广义超立方体上的多播和广播等容错路由算法。

# 参考文献

[1] 臧大伟，曹政，孙凝晖. 高性能计算的发展[J]. 科技导报, 2016, 34(14): 22–28. [2] 陈国良，毛睿，蔡晔. 高性能计算及其相关新兴技术[J]. 深圳大学学报(理工版),

2015, 32(1): 25–31.

[3] 徐子沛. 大数据:正在到来的数据革命，以及它如何改变政府、 商业与我们的生

活[M]. 广西师范大学出版社, 2015.

[4] 廖湘科，肖侬. 新型高性能计算系统与技术[J]. 中国科学:信息科学, 2016, 46(9):

1175.

[5] 张林波. 并行计算导论[M]. 清华大学出版社, 2006.

[6] 刘晓婷，贾志淳. 高性能计算机中互连网络的可靠性研究[J]. 电子制作, 2015, 7z:

151–152.

[7] 宋玉艳. 并行计算机的互连网络[J]. 电大理工, 2012, 2: 41–42.

[8] Bhuyan L N and Agrawal D P. Generalized hypercube and hyperbus structures for a computer network[J]. IEEE Transactions on Computers, 2006, C-33(4): 323–333.

[9] Duh D R, Chen G H and Hsu D F. Combinatorial properties of generalized hypercube graphs[J]. Information Processing Letters, 1996, 57(1): 41–45.

[10] Lawson L M, Boland J W and Ringeisen R D. *i*-connectivity in a generalized hyper- cube[J]. Combinatorial Mathematics Combinatorial Computing, 2001, 39: 127–137.

[11] Shao D C, Hong S and Topor R. An efficient algorithm for constructing hamiltonian paths in meshes[J]. Parallel Computing, 2002, 28(9): 1293–1305.

[12] Latifi S and Zheng S Q. On link-disjoint hamiltonian cycles of torus networks[J].

Computers Electrical Engineering, 1997, 23(1): 25–32.

[13] Day K and Tripathi A. Unidirectional star graphs[J]. Information Processing Letters, 1993, 45(3): 123–129.

[14] Harary F, Hayes J P and Wu H J. A survey of the theory of hypercube graphs[J].

Computers Mathematics with Applications, 1988, 15(4): 277–289.

[15] Efe K. The crossed cube architecture for parallel computation[J]. IEEE Transactions on Parallel Distributed Systems, 1992, 3(5): 513–524.

[16] Wang D. On embedding hamiltonian cycles in crossed cubes[J]. IEEE Transactions on Parallel Distributed Systems, 2007, 19(3): 334–346.

[17] Cull P and Larson S. M. The Mo¨bius cubes[J]. IEEE Transactions on Computers, 1995, 44(5): 647–659.

[18] Fan J. Hamilton-connectivity and cycle-embedding of the Mo¨bius cubes[J]. Informa- tion Processing Letters, 2002, 82(2): 113–117.

[19] 来光明. 多级互连网络[C]. 中国电子学会数据通信学术会议, 1990.

[20] 傅海帆，陈良生. 基于交叉开关网络的多处理机系统研究[J]. 华中科技大学学 报(自然科学版), 1995, 23(9): 43–46.

[21] 张芳娥，衡君山，甄换强. RS485总线网络可靠性研究[J]. 仪器仪表学报, 2006,

27(z3): 2458–2459.

[22] Lakshmivarahan S and Dhall S K. A new hierarchy of hypercube interconnection schemes for parallel computers[J]. Journal of Supercomputing, 1988, 2(1): 81–108.

[23] Fragopoulou P, Akl S G and Meijer H. Optimal communication primitives on the generalized hypercube network[J]. Journal of Parallel and Distributed Computing, 1996, 32(2): 173–187.

[24] Ziavras S G and Krishnamurthy S. Evaluating the communications capabilities of the generalized hypercube interconnection network[J]. Concurrency: Practice and Experi- ence, 1999, 11(6): 281–300.

[25] Ziavras S G. Scalable multifolded hypercubes for versatile parallel computers[J]. Par- allel Processing Letters, 1995, 5(2): 241–250.

[26] Ziavras S G, Grebel H and Chronopoulos A. A low-complexity parallel system for gracious scalable performance. Case study for near PetaFLOPS computing[C]. Pro- ceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computing, 1996: 363–370.

[27] Szymanski T. A fiber optic hypermesh for SIMD/MIMD machines[C]. Proceedings of IEEE Conference on Supercomputing, 1990: 710–719.

[28] Guo C, Lu G, Li D, Wu H, Zhang X, Shi Y, Chen T, Zhang Y and Lu S. BCube: a high performance, server-centric network architecture for modular data centers[C]. Proceedings of Special Interest Group on Data Communication (SIGCOMM), 2009: 63–74.

[29] Li D and Wu J. On the design and analysis of Data Center Network architectures for interconnecting dual-port servers[C]. Proceedings of IEEE Conference on Computer Communications, 2014: 1851–1859.

[30] D Li and J Wu. On Data Center Network Architectures for Interconnecting Dual-Port Servers[J]. IEEE Transactions on Computers, 2015, 64(11): 3210–3222.

[31] Abts D, Marty M R, Wells P M, Klausler P and Liu H. Energy proportional datacen- ter networks[C]. Proceedings of International Symposium on Computer Architecture, 2010: 338–347.

[32] Ahn J H, Binkert N, Davis A, McLaren M and Schreiber R S. HyperX: topology, rout- ing, and packaging of efficient large-scale networks[C]. Proceedings of the IEEE Con- ference on High Performance Computing Networking, Storage and Analysis, 2009: 1–11.

[33] Hsien S−Y and Weng Y−F. Fault-tolerant embedding of pairwise independent hamil-

tonian paths on a faulty hypercube with edge faults[J]. Theory of Computing Systems,

2009, 45(2): 407–25.

[34] Lin T−J, Hsien S−Y and Juan S−T. Embedding cycles and paths in product networks and their applications to multiprocessor systems[J]. IEEE Transactions on Parallel and

Distributed Systems, 2012, 23(6): 1081–1089.

[35] Wada K, lkeo T, Kawaguchi K and Chen W. Highly fault-tolerant routings and fault- induced diameter for generalized hypercube graphs[J]. Journal of Parallel and Dis- tributed Computing, 1997, 43(1): 57–62.

[36] Harary F. Conditional connectivity[J]. Networks, 1983, 13(3): 347–357.

[37] Esfahanian A H. Generalized measures of fault tolerance with application to *n*-cube networks[J]. IEEE Transactions on Computers, 2015, 38(11): 1586–1591.

[38] Latifi S, Hegde M and Naraghi-Pour M. Conditional connectivity measures for large multiprocessor systems[J]. IEEE Transactions on Computers, 1994, 43(2): 218–222.

[39] Wu J and Guo G. Fault tolerance measures for *m*-ary *n*-dimensional hypercubes based on forbidden faulty sets[J]. IEEE Transactions on Computers, 1998, 47(8): 888–893.

[40] Li X J and Xu J M. Generalized measures of fault tolerance in exchanged hyper- cubes[J]. Information Processing Letters, 2013, 113(14-16): 533–537.

[41] Ye L and Liang J. On conditional *h*-vertex Connectivity of some networks[J]. Chinese Journal of Electronics, 2016, 25(3): 556–560.

[42] Hsieh S−Y, Huang H−W and Lee C−W. {2, 3}-Restricted connectivity of locally twist-

ed cubes[J]. Theoretical Computer Science, 2016, 615: 78–90.

[43] Fabrega J and Fiol M A. On the extraconnectivity of graphs[J]. Discrete Mathematics, 1996, 155(1-3): 49–57.

[44] Yang W and Meng J. Extraconnectivity of hypercubes (II)[J]. Applied Mathematics Letters, 2010, 22(6): 887–891.

[45] Chang N−W and Hsieh S−Y. {2, 3}-Extraconnectivities of hypercube-like networks[J].

Computer and System Sciences, 2013, 79(5): 669–688.

[46] Li H and Yang W. Bounding the size of the subgraph induced by *m* vertices and extra edge-connectivity of hypercubes[J]. Discrete Applied Mathematics, 2013, 161(16-17): 2753–2757.

[47] Yang W and Lin H. Reliability evaluation of BC networks in terms of the extra vertex- and edge-connectivity[J]. IEEE Transactions on Computers, 2014, 63(10): 2540–2548.

[48] Gu Q P and Peng S. Unicast in hypercubes with large number of faulty nodes[J]. IEEE Transactions on Parallel Distributed Systems, 1999, 10(10): 964–975.

[49] Li Y, Peng S and Chu W. An ffficient algorithm for fault tolerant routing based on adaptive binomial-tree technique in hypercubes[C]. Proceedings of Applications and Technologies International Conference on Parallel and Distributed Computing, 2008: 196–201.

[50] Wu J. Reliable unicasting in faulty hypercubes using safety levels[J]. IEEE Transac- tions on Computers, 1997, 46(2): 241–247.

[51] Wu J. Adaptive fault-tolerant routing in cube-based multicomputers using safety vec- tors[J]. IEEE Transactions on Parallel Distributed Systems, 1998, 9(4): 321–334.

[52] 童明生，刘长河，范天佑. 一般化超立方网络的容错寻径算法[J]. 计算机学报, 1998, 21(12): 1074–1083.

[53] 刘红美. 广义超立方体网络容错路由算法[J]. 武汉理工大学学报(交通科学与工 程版), 2006, 30(4): 682–685.

[54] 孙惠泉. 图论及其应用[M]. 科学出版社, 2004.

[55] Hsu L−H and Lin C−K. Graph theory and interconnection networks[M]. CRC Press, Inc., 2008.

# 攻读硕士学位期间发表的论文和参与的科研项目

作者在攻读硕士学位期间发表的论文

[1] **Guo L**, Wang X, Zhou J, Lin C-K and Fan J. A fault-free unicast algorithm in the generalized hypercube with restricted faulty vertices[J]. International Journal Foundation of Computer Science,2017,28(7):915-929(SCI已收录).

[2] 林政宽, 王铭成, 郭莉莉, 杜满意. 基于有故障区域的Mesh网络目标结点间的 最短路由算法[J]. 计算机科学, 2017, 44(s1):252-257.

作者在攻读硕士学位期间申请的软件著作权

[1] 郭莉莉. BCDC网络性质演示软件v1.0，登记号：2016SR361019

[2] 郭莉莉. BCDC网络容错路由算法演示软件v1.0，登记号：2017SR102725

作者在攻读硕士学位期间参与的科研项目

[1] 参与国家自然科学基金面上项目“基于一类BC图的数据中心网络及其性质 的研究”，项目编号：61572337。

[2] 参与国家自然科学基金青年项目“类超立方体网络上的容错通信性能研究”， 项目编号：61602333。

[3] 参与国家自然科学基金青年项目”递归型数据中心网络上的条件容错通信性 能研究”，项目编号：61702351。

[4] 参与国家自然科学基金面上项目“多变化环境监测系统的系统诊断结构与高 效诊断算法分析与研究”，项目编号：61572340。

作者在攻读硕士学位期间的获奖情况

[1] 2014–2015学年，获苏州大学“学业优秀二等奖学金”。

[2] 2015–2016学年，获苏州大学“学业优秀一等奖学金”。

# 致 谢

时光荏苒，岁月如梭，看似漫长的三年硕士生生活即将结束。 这三年中有过喜 悦，也有过挫折和成长，这些经历不论酸甜苦辣，都将是我永远的财富。 值此毕业 论文成文之际，我要向所有帮助过我的老师、 领导、 同学和亲人表示我最真挚的谢 意。

首先要感谢我的导师樊建席教授和林政宽副教授。 三年来，两位老师在学习、 工作和生活方面给予了我无微不至的关怀。 两位老师严谨的治学态度、 扎实深厚的 学识功底和对前沿科学不断追求的精神让我受益匪浅。 本文的选题、 研究到形成无 不凝结着你们辛勤的汗水。 从两位身上，我不仅学到了扎实的专业知识，进行科学 研究的方法，更学到了做人的道理。 感谢两位在整个研究生学习期间对我的耐心指 导和帮助。 衷心祝愿二位老师身体健康、万事如意！

感谢苏州大学计算机学院的所有老师，谢谢你们提供的良好的学习环境和所创 造的浓厚的学术氛围。 感谢韩月娟老师、 程宝雷老师、 王岩老师、 刘钊老师和周经 亚老师等，感谢你们在学习、生活和工作方面对我的关系和帮助。

感谢师兄王喜、 周东仿、 樊卫北、 杜满意和陈琪，师姐吕雅丽、 李小燕、 王桂 娟和张丽丽，师妹王丽丹、 张小菲，以及师弟徐昕、 刘宁宁他们在生活和学习中给 予了我无私的帮助和关心。 同时，感谢舍友郭安倩和邵佳烨，谢谢你们在生活中的 温馨陪伴。 感谢我的父母，感谢你们对我学习的支持，我能顺利完成学业离不开你 们的鼓励和帮助。

最后，感谢评审本论文的专家，感谢您抽出宝贵时间来阅读本文，并提出宝贵 意见和建议。

郭莉莉 二〇一八年三月二十日