中山大学硕士学位论文

顶点覆盖和反馈顶点集问题的参数算法研究

Research on Parameterized Algorithms for Vertex Cover and Feedback Vertex Set

学位申请人: 林泽群

指 导 教 师: 刘咏梅教授 林瀚讲师

专 业 名 称: 计算机科学与技术

答辩委员会主席 (签名): _____

答辩委员会成员 (签名): _____

_____

_____

_____

_____

2016 年 5 月

论文原创性声明

本人郑重声明: 所呈交的学位论文, 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, 本论文不包含任何其他个人或集体已经发表或撰写过的作品成果. 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. 本人完全意识到本声明的法律结果由本人承担.

学位论文作者签名:_____

日期:_____

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定, 即: 学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版, 有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅, 有权将学位论文的内容编入有关数据库进行检索, 可以采用复印、缩印或其他方法保存学位论文.

学位论文作者签名:　　导师签名:

日期:　　年　　月　　日　　日期:　　年　　月　　日

论文题目: 顶点覆盖和反馈顶点集问题的参数算法研究

专　　业: 计算机科学与技术

硕 士 生: 林泽群

指导教师: 刘咏梅教授 林瀚讲师

摘　　要

对于 NP 难问题，如果能基于参数 k 及问题规模 n 设计出时间复杂度为 $O(f(k)n^{O(1)})$ 的算法，则称该问题是固定参数可解的。在理论计算机科学中，为 NP-难问题设计实际有效的固定参数算法是当前研究中一个重要的方向。本文以顶点覆盖、反馈顶点集两个经典的 NP-难问题为研究对象，通过挖掘问题的性质，应用多种参数计算技术分别为之设计了相应的固定参数算法。值得一提的是，本文中所有研究都是基于一般无向图，更加具有普遍性。

在顶点覆盖问题上，我们设计了基于线性松弛下界的参数化顶点覆盖问题的固定参数算法。该问题以目标顶点覆盖集大小和最小顶点覆盖集的线性松弛下界的差值作为算法的参数 μ。我们首先使用最大网络流来维护图的顶点覆盖集的线性松弛下界，并且在最大流的残余图上线性地将问题实例核心化，最后使用分支搜索树顺利获得一个时间复杂度为 $O(2.618^{\mu}(|V|+|E|))$ 的线性固定参数算法。对比之前该方向最好的研究成果，Iwata 等[26] 提出的时间复杂度为 $O(4^{\mu}(|V|+|E|))$ 的算法，复杂度上有了明显地下降。

在反馈顶点集问题上，我们设计了一个复杂度为 $\mathcal{O}^*(3.598^k)$ 的固定参数算法。其主要思路是，首先使用迭代压缩技术将问题转化成不相交反馈顶点集问题，之后我们使用分支搜索树来求解不相交反馈顶点集问题。在其中我们创新性地同时使用两个参数限制搜索树的规模。当前在反馈顶点集问题上最好的固定参数算法是 Kociumaka 等[31] 提出的，复杂度是 $\mathcal{O}^*(3.592^k)$。与之对比，时间复杂度上十分接近，然而相比之下 Kociumaka 的算法一共使用了 12 条分支规则，而我们的算法只有 1 条更加简洁明了。

关键词: 参数计算，固定参数算法，NP 难问题, 顶点覆盖，反馈顶点集

Title:	Research on Parameterized Algorithms for Vertex Cover and Feedback Vertex Set
Major:	Computer Science and Technology
Name:	Zequn Lin
Supervisor:	Prof. Yongmei Liu, Lecturer Han Lin

Abstract

An NP-hard problem is called fixed-parameter tractable (FPT) with respect to a parameter k if it can be solved in time $O(f(k)n^{O(1)})$, where n is the input size and f is some computable function. In the area of theoretical computer science, designing FPT algorithms for NP-hard problems is an important research direction. In this paper, we study two classical NP-hard problems, the Vertex Cover problem and the Feedback Vertex Set problem and design FPT algorithms for these problems respectively. It should be noted that all algorithms in the paper are on general undirected graph.

In this paper, we design an FPT algorithm for Vertex Cover parameterized by the difference μ between the solution size and the LP lower bound (we call the problem Vertex Cover Above LP). Firstly, we use the maximum network flow to maintain the LP lower bound of Vertex Cover. And then we successfully kernelize the input instance by the residual graph. At last, we employ the case-by-case branching technique and obtain a linear-time FPT algorithm that runs in $O(2.618^{\mu}(|V|+|E|))$, surpassing perviously fastest linear-time FPT algorithm due to Iwata et al.(2014)[26] that runs in $O(4^{\mu}(|V|+|E|))$.

For the Feedback Vertex Set problem parameterized by the solution size k, we design a deterministic $\mathcal{O}^*(3.598^k)$-time FPT algorithm. In our algorithm, we first employ the iterative compression technique in a standard manner to reduce the problem to the disjoint compression variant(Disjoint-FVS). Then we use a branching rule to cope with the Disjoint-FVS problem. In the branching steps we use two parameters to analysis the size of the search tree innovatively. The fastest known algorithm on this problem due to Kociumaka et al.(2013)[31] runs in $\mathcal{O}^*(3.592^k)$-time, which is very close to our algorithm. But our algorithms is much simpler in that we use only one branching rule while Kociumaka uses twelve branching rules.

Key Words: Parameterized Complexity, Fixed-Parameter Tractable Algorithm, NP-hard Problem, Vertex Cover, Feedback Vertex Set

目 录

第一章 绪论

参数计算是现代理论计算机学科的一个重要分支。本章首先就其相关研究背景和研究意义做了简单地介绍。之后，我们总结了本文中在顶点覆盖和反馈顶点集问题上的研究工作；最后，简单介绍了整篇论文章节的组织结构。

1.1 研究背景及意义

随着现代计算机科学的迅速发展，以及计算机科学在各个研究领域的应用，许多 NP 难问题频繁地出现。尽管在 NP 完全性理论[21] 中，NP 不等于 P 依然没有被严谨地证明，但是计算机界普遍的共识是，NP 难问题是无法用现代计算机有效求解的。然而，所有这些现实应用中的 NP 难解问题又是必须解决的问题，因此在 NP 完全性理论的发展过程中，人们提出了一系列实际解决 NP 难问题的方法，如启发式算法[35]、近似算法[1, 25]、随机算法[37, 38] 等。但是，这些算法都存在着各自明显的不足。人们需要新的方法来解决 NP 难问题。参数计算与复杂性理论正是在这种情况下诞生的。二十世纪末，以 DowneyR 和 FellowsM 为代表的计算机理论专家首次提出了参数复杂性的概念，并在其专著《参数复杂性》[18] 中系统地阐述了参数计算与复杂性理论，标志着参数计算与复杂性理论的诞生。自从参数计算与复杂性理论的诞生以来的十几年的时间里，参数计算与复杂性理论得到迅速发展和壮大，很快成为理论计算机科学的一个重要分支。

参数算法的研究起源于对很多计算问题的观察，研究人员发现其中很多问题都与一个关键参数 k 相联系。而很多实际应用中，这一参数只在一个较小的区间内变化，与问题的规模并无直接关系。从这个方面着手，很多理论上难解的问题可以在实际应用中得到解决。首先，我们把初始 NP 难问题进行参数化，将原来的问题实例 X，表示成其参数化版本 (X, k) 的形式。当一个参数化的问题，可以在 $O(f(k)n^c)$ 时间复杂度内解决，我们就称该问题是固定参数可解（Fixed-Parameterized Tractable，简称 FPT）的。这里 c 是某一常数，$f(k)$ 是一个只跟 k 有关的函数与问题的规模 n 无关。一般情况下当 k 不是很大的时候，函数 $f(k)$ 的值也不会很大。所以此时这一问题在实际应用中是可以有效求解的。如在，在文献 [23] 中，研究人员将 NP 难的基因序列冲突问题转化成参数化顶点覆盖问题。而关于此问题已经存在了时间复杂度为 $\mathcal{O}^*(1.2738^k)$ 的固定参数算法 [1]（Fixed-Parameterized Tractable Algorithm, 简称 FPT 算法）。在基因序列冲突问

[1]对于时间复杂度为 $O(f(k)n^c)$ 的固定参数算法，在参数计算领域有时候会忽略掉其中多项式部分并将其时间复杂度记为 $\mathcal{O}^*(f(k))$

题中，其特殊的生物背景决定了顶点覆盖集的规模 k 不超过 60，而图的规模 n 可能非常大。因此顶点覆盖问题的固定参数算法能在较短的时间内解决了基因序列冲突问题。

另外一方面，参数计算理论也为计算问题的复杂性提供进一步分析的工具。如，经典的最大独立集问题，在以独立集规模 k 作为参数时，研究人员一直无法找到对于一个函数 f，使得独立集问题可以在 $O(f(k)n^c)$ 的时间复杂度内求解。普遍认为，该问题不可能是固定参数可解的。为了描述问题的这一性质，人们引入了固定参数不可解的概念[1]。在实际应用中，这方面的研究尽管没有具体解决问题，但是在改善问题的思路上提供了方向性的指引。如，在文献 [43] 中，Papadimitriou 等证明了数据库查询在以询问语句规模（即语句中涉及变量数量）作为参数时候是固定参数不可解的。这一结论的证明，让许多数据库研究人员避免消耗无谓的精力在一个不可行的方向中。

从上述例子中，我们可以看到参数计算与复杂性理论在解决实际应用中的问题具有独特的优势，对其进行进一步丰富和发展正是非常有价值的研究方向。

1.2 研究内容

一般来说在参数计算理论的研究，主要可以分成以下三个方向： 参数化问题的参数可计算性，参数可枚举性以及固定参数算法的设计。其中参数化问题的可计算性主要集中在研究问题本身的复杂性，尝试证明证明 NP 难问题在某一参数定义下是否属于固定参数可解（FPT）；固定参数可枚举性的研究是指在固定参数可解的基础上，研究能否有效地枚举问题的所有或者部分解；而最后参数算法的设计，主要包括的参数算法技术的应用以及新的参数算法技术的研究。本文主要工作是对一般无向图上基于线性松弛下界的参数化顶点覆盖问题和参数化反馈顶点集问题的参数算法的研究，属于参数算法设计的范畴。

1.2.1 基于线性松弛下界的参数化顶点覆盖问题的 FPT 算法研究

顶点覆盖问题（Vertex Cover，缩写为 VC），即对于参数 k，询问给定图中是否存在大小不超过 k 的顶点集合满足：对于图中任意一条边的两个端点至少有一个在该集合中。该问题是最早被提出来的 21 个 NP 完全问题之一[28]，也是最早证明是固定参数可解 (Fixed parameter tractable) 的问题[19]。

目前已经有很多这方面的研究，其中以 k 作为参数的固定参数算法已经可以运行在 $\mathcal{O}^*(1.2738^k)$ 的时间复杂度内。近年来有研究提出了顶点覆盖问题参数化的新思路，

[1]注意是否参数可解跟选取的参数也有关系，如在一般图团问题上，在以团规模 k 作为参数时是固定参数不可解的，但是以图的最大度数 d 作为参数时其又是固定参数可解的。

开始使用 k 与某些最小顶点覆盖问题的下界的差作为新的参数[26, 34, 36, 39, 44, 47]。显然基于下界的参数化方法，有效地缩小了参数大小，比经典版本更加有实用价值。本文对该问题的研究主要是基于线性松弛下界。对于该问题的参数算法，最优秀的 $f(\mu)$ 函数已经优化到 $f(\mu) = 2.3146^{\mu}$。然而尽管该算法获得了一个优秀的 $f(\mu)$ 函数，但是算法本身多项式部分的级别十分高，实际执行效率并不高。针对该问题，有研究人员期望能在保持 c=1 的同时尽量优化 $f(u)$，当前最成功的是一个 $O(4^{\mu}(|V| + |E|))$ 的线性固定参数算法。

我们在前人的研究基础上继续优化该问题,并成功获得时间复杂度为 $O(2.618^{\mu}(|V|+|E|))$ 的线性固定参数算法，较之前的最好结果有了较大的改进。

1.2.2 参数化反馈顶点集问题的 FPT 算法研究

反馈顶点集问题（Feedback Vertex Set，简称 FVS）是指给定参数 k，询问给定图中是否存在大小不超过 k 的顶点集合满足：图中任何一个环都经过该顶点集合。该问题也是最早 21 个 NP 完全问题之一[28]。

针对参数化反馈顶点集问题的研究，分成了两条分支，其一主要研究的是固定参数的随机算法，当前基于 Cut&Count 技术，已经可以获得一个时间复杂度为 $\mathcal{O}^*(3^k)$ 的固定参数随机算法[14]；另外的方向是设计确定性的固定参数算法，当前最低的复杂度是 Kociumaka 等设计的 $\mathcal{O}^*(3.592^k)$[32]。

本文中我们研究的是参数化反馈顶点问题的确定性固定参数算法，我们成功设计了一个复杂度为 $\mathcal{O}^*(3.598^k)$ 的 FPT 算法。我们的算法与当前最好的算法相比，复杂度上十分接近，但是却简单直接很多（Kociumaka 等的算法有 12 条分支规则之多，而我们的只有 1 条）。

1.3 论文结构

本论文共有五章，组织如下：

第一章：绪论。简要介绍固定参数算法的背景概念，并指明本文的主要工作，概述文章的组织结构。

第二章：相关问题研究现状。这一章介绍了当前参数计算领域的研究现状，并重点介绍了顶点覆盖和反馈顶点集两个问题上近年来重要参数算法上的研究成果。

第三章：基于线性松弛下界的参数化顶点覆盖问题。本章之中我们对基于线性松弛下界的参数化顶点覆盖问题进行了详细的分析，提出了一个时间复杂度为 $O(2.618^{\mu}(|V| + |E|))$ 的线性固定参数算法，并给出正确性证明。

第四章：反馈顶点集问题的固定参数算法。这一章里主要是讨论反馈顶点集，针对该问题提出了一个时间复杂度为 $\mathcal{O}^*(3.598^k)$ 的固定参数算法，并给出正确性证明。

第五章：总结与展望。总结本文的工作内容，分析其中的不足之处，展望未来。

第 二 章 相关问题研究现状

本章中我们会介绍参数计算领域的研究现状，其中重点介绍本文研究的两个 NP 难问题（顶点覆盖问题和反馈顶点集问题）在该领域的研究成果。

2.1 参数计算基本概念

定义 2.1 (**参数化问题**) 参数化问题是一个判定型问题，其问题实例形式如下 $I(Q,k)$，其中 Q 是我们需要解答的问题，k 是一个非负整数称之为问题实例 I 的参数。如果问题 Q 存在满足参数 k 约束的解，则问题实例有解，否则问题实例无解。

很多 NP 难问题的标准参数化版本是用参数 k 约束目标解的规模，即询问问题 Q 是否存在一个大小小于 k 的解。当然，这并不是必须的，同一个 NP 难问题如果对于参数的定义不一样，可能会有多个参数化版本。在本章介绍参数化顶点覆盖问题的研究中，我们就会看到不同的参数化顶点覆盖问题。

定义 2.2 (**固定参数可解** (Fixed-Parameter Tractable, **简称** FPT)) 如果对于一个参数化问题的任何实例 $I(Q,k)$，存在一个算法 A 都能在 $O(f(k)n^c)$ 的时间复杂度内判定其是否有解，其中 $f(k)$ 是一个可计算函数，n 是问题 Q 的规模，c 是一个与 n 和 k 无关的常数，则称该参数化问题是固定参数可解 (FPT) 的。算法 A 称之为该问题的一个固定参数算法（FPT 算法）。

观察上述定义，我们看到一个固定参数算法的时间复杂度是由两部分组成的。对于 NP 难问题，$f(k)$ 函数的级别肯定是高于多项式的，而剩余部分 n^c 是关于问题规模的多项式函数。在很多情况下，因为多项式函数的增长远低于 f 函数，我们会忽略复杂度中多项式部分，而将一个固定参数算法的时间复杂度直接记为 $\mathcal{O}^*(f(k))$。

定义 2.3 (**核** (Kernel)) 如果对于一个参数化问题的任何实例 $I(Q,k)$，存在一个算法 A 和关于 k 的函数 $g(k)$, 其中算法 A 能在多项式时间内将其转化成另外一个问题实例 $I'(Q',k')$，使得 $k' \leq k$, 新问题实例中 Q' 的规模 $n' \leq g(k)$，并且问题实例 I 有解当且仅当新问题实例 I' 有解。则我们将 I' 称之为 I 的核，算法 A 称之为该参数化问题的一个核心化算法。

定义 2.4 (**固定参数化归约** (**简称，**FPT-**归约**)) 对于一个参数化问题 P，如果存在一个算法 A 可以在多项式时间内将其每个实例 $I(Q,k)$ 转化成另外一个转化成另外

一个参数化问题 P' 中对应的实例 $I'(Q', k')$，并且 $k' = g(k)$。则我们称问题 P 可以参数化归约到另外一个参数化问题 P'。

注意上述两个定义中，核心化是在同个参数化问题里的实例之间的转化，而归约化则是问题之间的转化。

2.2 参数化顶点覆盖问题研究现状

经典的参数化顶点覆盖问题是以目标顶点覆盖集大小作为问题的参数，如下

参数化顶点覆盖问题 (Vertex Cover)
输入： 图 $G(V, E)$ 及整数 k
参数： k
问题： 图 G 中是否存在一个顶点覆盖集合，其大小不超过 k

该问题的第一个 FPT 算法是由 Buss 和 Goldsmith 设计的，时间复杂度为 $O(2^k k^{2k+2} + kn)$[7]。之后的 20 多年了，有一系列研究在不断地改进其 FPT 算法的时间复杂度[2, 10, 12, 41, 42, 50]。当前最快的算法是由 Chen 和 Kanj 在 2010 提出的[11]，他们成功将该问题的下界改进到 $\mathcal{O}^*(1.2738^k)$。这其中大部分算法的设计是基于分支搜索技术，他们罗列出各种情形下适用的分支规则，并考虑最坏情况下的搜索树规模跟参数 k 之间的关系。

在设计 FPT 算法的同时也有一些研究人员对于参数化顶点覆盖问题的核感兴趣[10, 13, 33, 49]。已经证明了不可能存在核心化算法使得该问题的任意实例都能产生小于 $O(ck)$ 的核，其中 c 是小于 2 的常数[30]。目前最好的研究成果是文献 [33]，在其中 Lampis 设计了一个算法对于任意常数 c 都能在多项式时间内生成一个 $O(2k - c\log k)$ 级别的问题实例的核。

在经典的参数化顶点覆盖问题中，到当 k 小于某些最小顶点覆盖集的下界时，讨论时间复杂度是没有意义的，此时问题是平凡的。于是在最几年该问题的研究中，研究人员开始聚焦于基于某些下界的参数化版本。如用 $mm(G)$ 表示图 G 的最大匹配数，显然对于任意图最小顶点覆盖数总是大于等于其最大匹配数。在文献 [47]，作者 Razgon and O'Sullivan 定义了该问题的一个新参数 $\mu_1 = k - mm(G)$，并且设计了一个时间复杂度为 $\mathcal{O}^*(15^{\mu_1})$ 的 FPT 算法。这里给出该版本的参数化顶点覆盖问题的定义，如下

基于保证值的参数化顶点覆盖问题

(Above-Guarantee Vertex Cover, 简称 AGVC)

输入： 图 $G(V,E)$ 及整数 k

参数： $\mu_1 = k - mm(G)$

问题： 图 G 中是否存在一个顶点覆盖集合，其大小不超过 k

假设 $vc^*(G)$ 表示图 G 的最小顶点覆盖线性松弛下界。在文献 [22, 39] 中，又分别定义了参数 $\mu_2 = k - vc^*(G)$ 和 $\mu_3 = k - 2vc^*(G) + mm(G)$，并且分别证明基于这两个参数定义的问题版本也是固定参数可解的。下面将这两个新问题的定义给出，如下：

基于线性松弛下界的参数化顶点覆盖问题

(Vertex Cover Above LP, 简称 VCAL)

输入： 图 $G(V,E)$ 及整数 k

参数： $\mu_2 = k - vc^*(G)$

问题： 图 G 中是否存在一个顶点覆盖集合，其大小不超过 k

基于 Lovász-Plummer 下界的参数化顶点覆盖问题

(Vertex Cover Above Lovász-Plummer, 简称 VCAL-P)

输入： 图 $G(V,E)$ 及整数 k

参数： $\mu_3 = k - 2vc^*(G) + mm(G)$

问题： 图 G 中是否存在一个顶点覆盖集合，其大小不超过 k

在这个 4 个不同的参数化顶点覆盖版本中，可以证明其中 4 个参数大小关系有 $k \geq \mu_1 \geq \mu_2 \geq \mu_3$，显然参数越小问题更加具有实际意义。在表2-1中，我们列举了近几年中对于这几个参数化顶点覆盖问题的研究成果。

表 2-1 参数化顶点覆盖问题 FPT 算法

参数问题	作者 (文献)	时间复杂度	发表会议（期刊） 时间
VC	Chen 和 Kanj[11]	$\mathcal{O}^*(1.2738^k)$	TCS 2008
AGVC	Razgon 等[47]	$\mathcal{O}^*(15^{\mu_1})$	ICALP 2008
AGVC	Raman 等[44]	$\mathcal{O}^*(9^{\mu_1})$	ESA 2011
AGVC	Cygan 等[15]	$\mathcal{O}^*(4^{\mu_1})$	IPEC 2011, TOCT 2013
VCAL	Narayanaswamy 等[39]	$\mathcal{O}^*(2.618^{\mu_2})$	STACS 2012
VCAL	Lokshtanov 等[34]	$\mathcal{O}^*(2.3146^{\mu_2})$	TALG 2014
VCAL	Iwata 等[26]	$O(4^{\mu_2}(\|V\|+\|E\|))$	SODA 2014
VCAL-P	Garg 等[22]	$\mathcal{O}^*(3^{\mu_3})$	SODA 2015

2.3 参数化反馈顶点集问题研究

反馈顶点集问题也是最早被证明是 NP 难的 21 个问题之一[28]，同样也有很多研究人员在参数计算领域下研究该问题。这里我们首先给出参数化反馈顶点问题的完整定义，如下:

参数化反馈顶点集问题 (Feedback Vertex Set，简称 FVS)
输入： 图 $G(V,E)$ 及整数 k
参数： k
问题： 图 G 中是否存在一个反馈顶点集合，其大小不超过 k

对于参数化反馈顶点集问题，早在 1992 年就由 Downey 和 Fellows 证明了是固定参数可解，他们给出了一个 $\mathcal{O}^*((2k+1)^k)$ 级别的固定参数算法。然而，这样的复杂度显然是没有实际应用价值的，当 k=10 时 $\mathcal{O}^*((2k+1)^k) \approx \mathcal{O}^*(10^{12})$，几乎不存在计算的可能性。在之后的 20 多年里，经过持续不断的改进，当前该问题上最快的固定参数算法已经到达了 $\mathcal{O}^*(3.592^k)$ 级别。下面表中，我们列出了对于该问题的所有改进。

表 2-2 参数化反馈顶点集问题 FPT 算法

作者 (文献)	时间复杂度	时间
Downey 和 Fellows[17]	$\mathcal{O}^*((2k+1)^k)$	1992
Bodlaender[4]	$\mathcal{O}^*(17(k^4)!)$	1994
Raman 等[45]	$\mathcal{O}^*(max(12^k,(4\log k)^k)))$	2002
Kanj 和 Pelsmajer 等[27]	$\mathcal{O}^*(((2\log k+2\log\log k+18)^k)$	2004
Raman 等[46]	$\mathcal{O}^*(((12\log k/\log\log k+6)^k)$	2006
Guo 和 Gramm[24]	$O(4^{\mu_2}(37.7^k))$	2006
Dehne 和 Fellows 等[16]	$\mathcal{O}^*(10.6^k)$	2007
Chen 和 Fomin 等[9]	$\mathcal{O}^*(5^k)$	2008
Cao 和 Chen 等[8]	$\mathcal{O}^*(3.83^k)$	2008
Kociumaka 和 Pilipczuk 等[31]	$\mathcal{O}^*(3.592^k)$	2008

在上述表2-2，我们列举的所有算法都是确定性的固定参数算法。还有另外一些研究人员尝试使用随机性的固定参数算法来求解 FVS 问题。这里的随机性算法在问题实例存在合法解的时候，会有一定概率返回“YES”，其他时候均返回“NO”。显然这里多次运行算法后，得出的结论基本上是可信的。早在 2000 年时，Bar-Yehuda 等就设计出时间复杂度为 $\mathcal{O}^*(4^k)$ 的随机固定参数算法[3]。而最近，Cygan 等设计了一种基于树宽求解连通性问题的新技术—Cut&Count，在此基础上他们将该技术与迭代压缩技术结合设计了一个时间复杂度为 $\mathcal{O}^*(3^k)$ 的随机固定参数算法 [14]。然而比较可惜的是这两个算法都很难被改造成确定性算法。

另外，FVS 问题的核心化技术也是我们设计该问题的固定参数算法的重要组成部分。如果在求解之前我们先计算出问题实例的核，之后在核上进行求解也能大大地降低问题实例的求解难度。对于这个方向的研究，最早由 Burrage 和 Estivill-Castro 等[6] 证明参数化反馈顶点集问题存在关于 k 的多项式规模的核。同样经过很多人的改进，现在已经有核心化算法可以获得平方级别的核，Thomassé 等通过一些系列的归约规则使得最终问题实例的规模不大于 $4k^2$[52]。下面表2-3中，我们列举出关于 FVS 问题的核的研究轨迹。

表 2-3 参数化反馈顶点集问题的核

作者 (文献)	**核的问题规模**	**时间**
Burrage 和 Estivill-Castro 等[6]	$O(k^{11})$	2006
Bodlaender[5]	$O(k^3)$	2007
Thomassé 等[52]	$4k^2$	2009

第 三 章　基于线性松弛下界的参数化顶点覆盖问题

顶点覆盖问题 (Vertex Cover，缩写为 VC)，即对于参数 k，询问给定图中是否存在大小不超过 k 的顶点集合满足：对于图中任意一条边的两个端点至少有一个在该集合中。该问题是最早被提出来的 21 个 NP 完全问题之一 [28]，也是最早证明是固定参数可解 (Fixed parameter tractable) 的问题 [19]。

本章针对顶点覆盖基于线性松弛下界的参数化版本（VACL）设计固定参数算法。我们在限制多项式部分复杂度为线性的前提下，尽量优化 $f(k)$ 函数，最终设计出了一个时间复杂度为 $O(2.618^{\mu}(|V|+|E|))$ 的线性固定参数算法。该算法由两部分组成核心化部分及分支搜索部分，其中核心化的方法我们使用了之前 Iwata 等[26] 的算法中使用的利用网络流来优化的方法，同时我们在该基础上通过对于分支搜索的策略优化使得整体算法的复杂度得到有效地降低。

3.1　相关术语及问题的定义

本节中，我们定义了算法中会使用的图论相关的术语并且引进了部分前人已经证明的结论。另外对于算法所解决的问题也给予了更加数学的全面的定义。

3.1.1　有关图的一些术语的定义

对于无向图 $G(V,E)$，V 是图 G 的顶点集合，E 是图 G 的边集合，令 $n=|V|$ 表示图的顶点数量，$m=|E|$ 表示图的边数量。如果 $e\in E$ 的两个端点分别为 u 和 v，则使用集合 $\{u,v\}$ 来表示边 e。对于图顶点的某个子集 $V'\subseteq V$，我们定义图 G 的关于 V' 的导出子图为 $G'=(V',E')$（简记为 $G[V']$），其中 $E'=\{\{u,v\}\in E\mid u\in V',v\in V'\}$。对于图 G 如下形式的导出子图 $G[V\setminus V']$，我们有时会更加直观的写成 $G\setminus V'$。

设 $v\in V$，定义由与 v 相邻的顶点构成的集合为 $N_G(v)=\{u\in V|\{u,v\}\in E\}$。同理对于 $V'\subseteq V$，定义 $N_G(V')=(\bigcup_{u\in V'}N_G(u))\setminus V'$。同时因为在某些情况下讨论 V' 的邻居集合时候我们不需要排除其本身中的顶点，故额外定义 $N_G[V']=\bigcup_{u\in V'}N_G(u)$。（注意，在前后文没有歧义的情况下，我们会简称这三个集合为 $N(v),N(V'),N[V']$）

另外在图中所有与顶点 v 相邻的边的集合，我们称之为 $\delta(v)$。对于顶点集合 S，与之相邻的边的集合，定义为 $\delta(S)=\bigcup_{v\in S}\delta(v)$。

定义 3.1 (**顶点覆盖集**)　对于图 $G(V,E)$，若集合 $V'\subseteq V$ 满足对于任意 $\{u,v\}\in E$，$u\in V'$ 或者 $v\in V'$，则称 V' 是图 G 的一个顶点覆盖集。若 V' 是图 G 所有顶点覆盖集

中最小的一个，则称顶点覆盖集 V' 是图 G 的最小顶点覆盖集。这里我们约定用 $vc(G)$ 表示图 G 的最小顶点覆盖集大小。

定义 3.2 (**独立集**)　对于图 $G(V,E)$，若集合 $V' \subseteq V$ 满足对于任意 $u,v \in V'$，$\{u,v\} \notin E$，则称 V' 是图 G 的一个独立集。

设 $V' \subseteq V$ 是图 G 的一个独立集，定义盈余值 $surplus(V') = |N(V')| - |V'|$。同时对于整个图 G 其盈余值 $surplus(G)$ 等于其所有独立集的盈余值的最小值。

对于有向图 $G(V,E)$，若 $e \in E$ 是从 u 到 v 的一条边，则使用元组 (u,v) 来表示边 e。[1] 设 v 是有向图 G 的一个顶点，我们使用 $\delta^-(v) = \{(u,v) \in E \mid u \in V\}$ 来表示所有进入顶点 v 的边的集合，用 $\delta^+(v) = \{(v,u) \in E \mid u \in V\}$ 来表示所有从顶点 v 出发的边的集合。同样我们将这两个定义扩展到有向图 G 中的顶点集合，令 $S \subseteq V$，则 $\delta^-(S) = \{(u,v) \in E \mid u \notin S, v \in S\}$ 表示所有进入 S 的边的集合，$\delta^+(S) = \{(v,u) \in E \mid u \notin S, v \in S\}$ 表示所有离开 S 的边的集合。

定义 3.3 (**强连通分量** (Strongly Connected Component))　在有向图 $G(V,E)$ 中，若集合 $S \subseteq V$ 满足对于任意 $u,v \in S$ 都存在一条路径可以从 u 到达 v，则称 S 是有向图 G 中的一个强连通分量。如果集合 S 同时满足 $\delta^+(S) = \emptyset$，则称 S 是有向图 G 的一个尾强连通分量（tail strongly connected component）；相反，如果集合 S 同时满足 $\delta^-(S) = \emptyset$，则称 S 是有向图 G 的一个头强连通分量（head strongly connected component）。

在文献 [48, 51] 中，已经分别给出了著名的 Tarjan 算法和 Kosaraju 算法。这两个算法都能在线性时间复杂度（即 $O(n+m)$）内求得有向图的所有强连通分量。因为它们都是计算机领域非常重要且经典的算法，本文默认读者都已经对其有一定了解，后续不再进行展开，而是直接使用结论：**求有向图所有强连通分量可以在线性时间复杂度（$O(n+m)$）内完成**。

接下来我们给出网络及其流的定义。

在图论中，网络是指一个边带权的有向图，即有向图 $G(V,E)$ 加上一个边的权值函数 $c : E \to \mathbb{N}^+$（在网络中又称容量函数）。对于 V 中的两个顶点 $s,t \in V$，一个可行 s-t 流（在上下文没有歧义的情况下，简称可行流），是指一个从边集到自然数集的映射

[1] 注意，在无向图中，顶点 u 和 v 之间的边是用集合 $\{u,v\}$ 来表示的。

$f : E \to \mathbb{N}^+$ 满足：

$$\begin{aligned} \sum_{e\in\delta^+(s)} f(e) &= \sum_{e\in\delta^-(t)} f(e) \\ \sum_{e\in\delta^+(v)} f(e) &= \sum_{e\in\delta^-(v)} f(e), \quad v \in V \setminus \{s,t\} \\ 0 \le f(e) &\le c(e), \qquad\qquad e \in E \end{aligned} \tag{3.1}$$

一般我们用 $F(f) = \sum_{e\in\delta^+(s)} f(e)$ 来表示可行流的流量。

最后，定义网络 $W(G,c)$ 在可行流 f 下的流量残余图，我们用有向图 $G(f)$ 来表示：

$$\begin{aligned} G(f) =& (V, E(f)) \\ E(f) =& \{(u,v) \mid (u,v) \in E,\ f(u,v) < c(u,v)\} \cup \\ & \{(v,u) \mid (u,v) \in E,\ f(u,v) > 0\} \end{aligned}$$

在文献 [20]，已经给出著名的 Ford-Fulkerson 算法。本文也不打算对其进行展开，这里直接给出结论：**使用 Ford-Fulkerson 算法可以在 $O(F(n+m))$ 时间复杂度内计算一个流量为 F 的 s-t 可行流；同时将一个 s-t 可行流增广 Δ 流量所需时间复杂度为 $O(\Delta(n+m))$。**

3.1.2 顶点覆盖问题的数学描述及其线性规划松弛

对于给定图 $G(V,E)$，求其最小顶点覆盖集合可以表示成下面整数规划模型：

$$\begin{aligned} &minimize\ (\mathbf{val(x)} = \sum_{v\in V} x(v)) \\ &subject\ to\ x(u) + x(v) \ge 1, \qquad (u,v) \in E \\ &\qquad\qquad x(u) \in \{0,1\}, \qquad\qquad u \in V \end{aligned} \tag{3.2}$$

其中对于图的每一个顶点 v，当 $x(v) = 1$ 时表示该顶点被选进顶点覆盖集中。显然上述整数规划模型的约束条件保证了对于图 G 的任意一条边，至少有一个端点处于目标顶点覆盖集中。同时最小化目标函数 $\sum_{v\in V} x(v)$ 保证了该顶点覆盖集最小。这里我们用 $vc(G)$ 表示最小顶点覆盖集合的大小。

在整数规划模型中，取消值必须是整数的约束条件，被称为线性规划松弛，松弛后可以获得对应线性规划模型。这里我们对上面的模型3.2中 $x(u) \in \{0,1\}$ 的约束条件进

行线性松弛，获得模型如下：

$$\begin{aligned} &minimize\ (\mathbf{val(x)} = \sum_{v \in V} x(v)) \\ &subject\ to\ x(u) + x(v) \geq 1, \qquad (u,v) \in E \\ &\qquad\qquad 0 \leq x(u) \leq 1, \qquad\quad u \in V \end{aligned} \tag{3.3}$$

对于线性规划松弛后的顶点覆盖问题，本文中我们用 $LPVC(G)$ 来表示。用 $vc^*(G)$ 来表示线性规划松弛后的顶点覆盖问题的最优解的值。显然，$vc(G) \geq vc^*(G)$。这里 $vc^*(G)$ 是最小顶点覆盖的一个下界，我们称之为**线性松弛下界**。

如果某个函数 $x : V \to \mathbb{R}$，满足上述模型3.3中的所有约束条件，则 x 被称为 $LPVC(G)$ 的一个解。在所有解中，$val(x)$ 达到最小化要求的，就是 $LPVC(G)$ 的最优解。另外如果对于所有顶点 $v \in V$，有 $x(v) = \frac{1}{2}$，则 x 被称为**全 $\frac{1}{2}$ 函数**，如果它同时也是 $LPVC(G)$ 的解则被称为**全 $\frac{1}{2}$ 解**。

早在 19 世纪 80 年代，纯粹的线性规划问题已经被证明是可以在多项式时间内求解的 [29]。显然 $vc^*(G)$ 也可以在多项式时间内获得。这里尽管由于线性规划松弛后的顶点覆盖问题中顶点的取值可以是分数，我们不能将 LPVC 的解直接转化成顶点覆盖问题的解，但是松弛后的线性规划问题还是为原问题的近似算法以及固定参数算法的设计提供了很多有价值的参考。下面我们给出若干 LPVC 的已经被证明的性质。

引理 3.1 ([40])　*对于图 $G(V,E)$，总是存在一个 LPVC(G) 的最优解 x，对于所有的 $v \in V$ 满足 $x(v) \in \{0, \frac{1}{2}, 1\}$。我们称 x 为 LPVC(G) 的一个半整数最优解，如果对于所有的 $v \in V$ 有 $x(v) = \frac{1}{2}$，则称 x 是 LPVC 的全 $\frac{1}{2}$ 最优解。*

在后续章节中因为我们总是处理 LPVC(G) 的半整数解，所以默认情况下我们说 x 是 LPVC(G) 的解是指 x 是 LPVC(G) 的半整数解。同时为了更清晰的使用 $LPVC(G)$ 的半整数解，我们约定对于 $i \in \{0, \frac{1}{2}, 1\}$ 使用 V_i^x 表示集合 $\{v \in V \mid x(v) = i\}$。

引理 3.2 ([40])　*如果 x 是 LPVC(G) 的最优解，那么存在一个图 G 的最小顶点覆盖集包含所有 V_1^x 中的顶点，并且不包含任何 V_0^x 中的顶点。*

引理3.2，揭示了线性松弛后的问题与原问题的之间最直观的联系。后面我们会利用该引理来有效缩小原问题的规模。

引理 3.3 ([40])　*对于图 $G(V,E)$，$surplus(G) \geq 0$ 等价于全 $\frac{1}{2}$ 函数是 LPVC(G) 的一个最优解。进一步看，$surplus(G) \geq 1$ 等价于全 $\frac{1}{2}$ 函数是 LPVC(G) 的唯一最优解。*

引理3.3，给出图 G 盈余值 $surplus(G)$ 与 $LPVC(G)$ 问题之间的联系。

3.1.3 基于线性松弛下界的参数化顶点覆盖问题

在前面我们已经说到在顶点覆盖问题中 $vc(G) \geq vc^*(G)$，当顶点覆盖固定参数算法中取所求顶点覆盖集大小 k 作为的参数，在 $k < vc^*(G)$ 的时候算法可以立刻返回“NO”，此时 k 是无意义的。当且仅当 $k \geq vc^*(G)$，固定参数算法才是非平凡的。显然此处会很自然的考虑是不是所求顶点覆盖集大小大于 $vc^*(G)$ 的部分才有实际意义，能不能以***所求顶点覆盖集大小减去*** $vc^*(G)$ ***的值***作为参数。

现给出基于线性松弛下界的参数化顶点覆盖问题 (VERTEX COVER ABOVE LP，简称 VCAL) 的完整定义。

基于线性松弛下界的参数化顶点覆盖问题 (VCAL)
输入： 图 $G(V, E)$ 及整数 k
参数： $\mu = k - vc^*(G)$
问题： 图 G 中是否存在一个顶点覆盖集合，其大小不超过 k

在文献 [39] 中，作者最早定义了 VCAL 问题，并且基于分支搜索技巧获得一个 $\mathcal{O}^*(2.618^{\mu})$ 的 FPT 算法。在后续的研究中，Lokshtanov 等将该问题的复杂度提升到了 $\mathcal{O}^*(2.3146^{\mu})$ [34]。这两个算法都是非常巧妙的，然而尽管他们在问题指数部分复杂度获得了一个优秀的上界，但是算法本身因为多项式部分糟糕的级别，实际执行效率并不高。针对该问题，Iwata 等[26] 尝试在缩小 FPT 算法指数部分的同时优化多项式部分复杂度，他们成功获得了一个时间复杂度为 $O(4^{\mu}(n+m))$ 的线性固定参数算法。

我们将在前人的研究基础上继续优化该问题,并成功获得时间复杂度为 $O(2.618^{\mu}(n+m))$ 的线性固定参数算法，较之前的最好结果有了较大的改进。本章后面部分将给出具体算法及其证明分析。

本文提出求解 VCAL 的算法主体部分是分支搜索，通过仔细设计分支规则以及分支前的核心化技巧，保证整棵搜索树的节点数量不超过 $2.618^{k-vc^*(G)}$。同时我们借鉴 Iwata 等[26] 核心化的思路，在搜索树的每个节点中，通过网络流算法 Ford-Fulkerson 来提升核心化的效率。最终成功获得搜索树节点个数不超过 $2.618^{k-vc^*(G)}$ 的线性固定参数算法。

3.2 将 LPVC 问题转换成最大网络流问题

LPVC 问题求解的是最小顶点覆盖的线性松弛下界，纯粹的线性规划问题已经被证明是可以在多项式时间内求解的[29]。然而我们算法的要求是希望每次在图发生改变之后更快地对新图进行求解。为了解决这一问题，本节中我们会将 LPVC 问题转换成最大网络流问题，维护最小顶点覆盖的线性松弛下界的任务因此也转换成了维护网络的最大流。

具体做法是我们首先使用其对偶模型来表示 LPVC 问题，之后将基于其对偶模型构造等价的网络。

3.2.1 原始对偶技巧

基于线性规划问题的对偶性，所有纯粹线性规划问题存在对应的原始对偶模型。下面，我们将顶点覆盖问题进行线性松弛后的获得的 LPVC 问题（模型3.3）转化成其原始对偶模型，如下：

$$\begin{aligned} &maximize\ (\mathbf{val(y)} = \sum_{e\in E} y_e) \\ &subject\ to \sum_{e\in\delta(v)} y_e \le 1, \qquad v \in V \\ &\qquad\qquad y_e \ge 0, \qquad\qquad e \in E \end{aligned} \tag{3.4}$$

这里为了区分两个求解 LPVC 问题的模型，我们将式子3.3命名为 Primal-LPVC，将式子3.4命名为 Dual-LPVC。下面的引理给出两个模型之间的联系。

引理 3.4　对于图 $G(V,E)$，假设函数 x 是其 Primal-$LPVC(G)$ 上的解，函数 y 是其 Dual-$LPVC(G)$ 上的解，则 $val(x) \ge val(y)$。若 $val(x) = val(y)$，则 x 与 y 分别 Primal-$LPVC(G)$ 和 Dual-$LPVC(G)$ 上的最优解。

证明　对于任意 E 中的边 $e = \{i,j\}$，在 Primal-$LPVC(G)$ 中有 $x_i + x_j \ge 1$，在 Dual-$LPVC(G)$ 中有 $y_e \ge 0$。结合上述两个不等式，显然有，

$$\sum_{e=\{i,j\}\in E} y_e(x_i + x_j) \ge \sum_{e\in E} y_e$$

对任意 V 中的顶点 v, $x_v \ge 0$ 且 $\sum_{e\in\delta(v)} y_e \le 1$，显然结合上述两个不等式可以获得，

$$\sum_{v\in V}(\sum_{e\in\delta(v)} y_e)x_v \le \sum_{v\in V} x_v$$

最后利用结合率，我们可以将 $\sum_{e=\{i,j\}\in E} y_e(x_i + x_j)$ 整理成 $\sum_{v\in V}(\sum_{e\in\delta(v)} y_e)x_v$。综上有,

$$\sum_{e\in E} y_e \leq \sum_{e=\{i,j\}\in E} y_e(x_i + x_j) = \sum_{v\in V}(\sum_{e\in\delta(v)} y_e)x_v \leq \sum_{v\in V} x_v$$

另外当 $\sum_{v\in V} x_v = \sum_{e\in E} y_e$ 时，对任意满足 Primal-$LPVC(G)$ 约束条件的 x', 我们可以获得 $\sum_{v\in V} x'_v \leq \sum_{e\in E} y_e = \sum_{v\in V} x_v$。因此 x 是 Primal-$LPVC(G)$ 的最优解，同理 y 也是 Dual-$LPVC(G)$ 的最优解。 ■

3.2.2 将 Dual-$LPVC(G)$ 转化成最大网络流问题

本节中我们首先将根据图 $G(V, E)$ 构造一个网络 W。之后并通过证明任意一个网络 W 上的可行流 f 与一个 Dual-$LPVC(G)$ 模型的解 y 一一对应且 $F(f) = 2val(y)$，来证明求 Dual-$LPVC(G)$ 模型的最优解等价与求解该网络上的最大流。最后，我们给出通过网络 W 最大可行流构造 Primal-$LPVC(G)$ 与 Dual-$LPVC(G)$ 最优解的方法。

求解 Dual-$LPVC(G)$ 的网络定义如下：

定义 3.4　对于图 $G = (V, E)$, 定义与其对应的网络 $W = (G_W,\ c)$，如下:

$$G_W = (V_W, E_W)$$

$$V_W = \{l_v \mid v \in V\} \cup \{r_v \mid r \in V\} \cup \{s, t\}$$

$$E_W = \{(s, l_v) \mid v \in V\} \cup \{(l_u, r_v) \mid \{u, v\} \in E\} \cup \{(r_v, t) \mid v \in V\}$$

$$c(e) = \begin{cases} 1, & \text{if } e = (s, l_v) \text{ or } e = (r_v, t) \\ \infty, & \text{otherwise.} \end{cases}$$

性质 3.1　图 G 上的一个 Dual-$LPVC(G)$ 模型的解 y, 对应网路 W 上的一个流量为 $2val(y)$ 的可行流。

证明　首先，我们构造一个流量函数 $f : E_W \to \mathbb{N}$, 如下：

$$\begin{cases} f(s, l_v) = f(r_v, t) = \sum_{e\in\delta(v)} y_e, & \text{for } v \in V \\ f(l_u, r_v) = y_e, & \text{for } \{u, v\} \in E. \end{cases}$$

回顾可行流需要满足的条件 (式子3.1), 首先由 Dual-$LPVC(G)$ 的约束可以得到，$0 \leq f(s,l_v) = f(r_v,t) = \sum_{e\in\delta(v)} y_e \leq 1$。其次，对于任意 $v \in V$ 显然,

$$\sum_{e\in\delta^+(l_v)} f(e) = f(s,l_v) = \sum_{e\in\delta(v)} y_e = \sum_{u\in N(v)} f(l_v,r_u) = \sum_{e\in\delta^-(l_v)} f(e)$$
$$\sum_{e\in\delta^+(r_v)} f(e) = \sum_{u\in N(v)} f(l_u,r_v) = \sum_{e\in\delta(v)} y_e = f(r_v,t) = \sum_{e\in\delta^-(r_v)} f(e)$$

综上 f 是网络 W 上的一个**可行流**, 其流量 $F = 2\sum_{e\in E} y_e = 2val(y)$（每条边上的 y_e 被计入两次)。证毕。 ■

性质 3.2　网络 W 上的一个可行流 f，对应一个图 G 上 Dual-$LPVC(G)$ 模型的解 y, 且 $val(y) = \frac{F}{2}$。

证明　构造 y 如下，$y_{e=\{u,v\}} = \frac{1}{2}(f(l_u,r_v) + f(l_v,r_u))$。此时对于任意顶点 $v \in V$ 有,

$$\sum_{e\in\delta(v)} y_e = \sum_{u\in N(v)} \frac{1}{2}(f(l_u,r_v) + f(l_v,r_u)) \leq \frac{1}{2}(c(s,l_v) + c(r_v,t)) = 1$$

因此 y 满足 Dual-$LPVC(G)$ 约束。

另外显然 $val(y) = \sum_{e\in E} y_e = \frac{1}{2} \sum_{\{u,v\}\in E} (f(l_u,r_v) + f(l_v,r_u)) = \frac{F}{2}$。证毕。 ■

最后我们给出通过网络 W 上的最大可行流，构造 Primal-$LPVC(G)$ 最优解方法。

假设 f 是网络 W 上的最大可行流，则可以通过网络 W 在可行流 f 下的残余图 $G_W(f)$ 构造 Primal-$LPVC(G)$ 的半整数最优解 x 如下：

$$x_v = \begin{cases} 0, & \text{如果在图 } G_W(f) \text{ 中，} l_v \text{ 是从 s 出发可达的并且 } r_v \text{ 是从 s 出发不可达的} \\ 1, & \text{如果在图 } G_W(f) \text{ 中，} l_v \text{ 是从 s 出发不可达的并且 } r_v \text{ 是从 s 出发可达的} \\ \frac{1}{2}, & \text{其他。} \end{cases}$$

表 3-1 $x(v)$ 的取值情况

l_v 是否可达	r_v 是否可达	$x(v)$
是	否	0
是	是	1/2
否	否	1/2
否	是	1

引理 3.5　*按照上述方法构造出来的 x 是 Primal-$LPVC(G)$ 的半整数最优解。*

证明　对于该引理的证明，我们将分成两步，首先证明 x 是 Primal-$LPVC(G)$ 的半整数解，再证明 $val(x) = \frac{1}{2}F(f)$。由于前文该网络的性质已经告诉我们，网络 W 上的一个可行流 f，对应一个图 G 上 Dual-$LPVC(G)$ 模型的解 y，且 $val(y) = \frac{F}{2}$。因此 $val(x) = val(y)$，根据引理3.4，可以得知 x 是 Primal-$LPVC(G)$ 的半整数最优解。

首先证明 x 是 Primal-$LPVC(G)$ 的半整数解。我们使用反证法。假设 x 并不是这样的一个半整数解。则存在图 G 的边 $\{u, v\}$，使得 $x(u) + x(v) < 1$。不失一般性，令 $x(u) = 0, x(v) \leq 1/2$。因此从表3-1我们可以看出，l_u 可达，r_u 不可达。回顾网络 W 的构造，可知其中存在边 $(l_v, r_u), (l_u, r_v)$ 并且边上的流量是无穷，因此残余图中 r_v 也是可达的。通过表3-1可知，顶点 v 符合情况 2，l_v 可达，r_v 可达。此时，在残余图 $G_W(f)$ 中，我们发现了增广轨 $s \to l_v \to r_u \to t$，这与 f 是最大流矛盾，因此假设不成立，x 是 Primal-$LPVC(G)$ 的半整数解。

证明 $val(x) = \frac{1}{2}F(f)$。令 $A = \{l_v \mid$ 如果在图 $G_W(f)$ 中，l_v 是从 s 出发不可达的$\}, B = \{r_v \mid$ 如果在图 $G_W(f)$ 中，r_v 是从 s 出发可达的$\}$。观察表3-1，可以知 $2val(x) = |A| + |B|$。如果不考虑任何残余图中的反向边，显然 $|A| = F(f)$。对于任何 $r_v \in B$，显然存在顶点 l_u 使得 $f(s, l_u) = f(l_u, r_v) = f(r_v, t) = 1$，否则残余图中会出现增广轨。顶点 l_u 原本是从 s 出发不可达的，考虑反向边 (r_v, l_u) 后，顶点 l_u 变成是从 s 出发可达的，因此 A 集合缩小 1。依次考虑完所有 B 中顶点后，可得 $|A| = F(f) - |B|$。综上有 $val(x) = \frac{1}{2}(|A| + |B|) = \frac{1}{2}F(f)$。

证毕。 ■

3.3 VCAL 问题的核心化

核心化技巧，是指通过多项式时间复杂度的算法将原本的参数化问题实例转化成一个规模更小（参数变小）的新问题实例，期间需要保证两个问题实例是等价的（其中一个问题实例存在解等价于另外一个存在解）。这是固定参数算法（FPT）的设计过程中一个非常重要的技巧。最直观的角度上看，通过核心化技巧可以缩小问题实例的规模，显然有利于后续的求解。同时当问题实例已经到不能进一步核心化的状态，那么可以从这一点上推导出当前实例很多特殊的性质。

本节中的核心化算法时间复杂度是 $O(|V|+|E|)$，会在每次运行分支算法之前被调用，以保证在分支算法运行的图中，全 $\frac{1}{2}$ 函数是 Primal-$LPVC(G)$ 问题的唯一最优解（$surplus(G) \geq 1$）。另外，我们要求核心化算法的输入中已经包含了求解当前图的 Dual-$LPVC(G)$ 的网络 W 以及 W 上的最大可行流 f，并且算法会构造并输出好基于新图的网络以及相应最大可行流。这个是我们本节里的核心化算法的目标。

接下来给出具体的算法，其包含两个阶段。

3.3.1 核心化算法阶段一

在本阶段中，算法对输入的图 G 进行收缩，使得全 $\frac{1}{2}$ 函数成为图 G 上 Primal-$LPVC(G)$ 问题的最优解（根据引理3.3，即 $surplus(G) \geq 0$）。

首先根据3.2.2节, 我们可以通过 W 上的最大可行流构造出图 G 上 Primal-$LPVC(G)$ 问题的最优半整数解以及 Dual-$LPVC(G)$ 问题的最优半整数解，分别用 x 和 y 表示。

从引理3.2上，我们知道存在一个图 G 的最小顶点覆盖包含所有 V_1^x 中的顶点，并且不包含所有 V_0^x 中的顶点。因此我们可以首先将图 G 中 V_1^x 里的顶点选进顶点覆盖集合里，再判断图的剩余部分是否存在最多 $k-|V_1^x|$ 个顶点的顶点覆盖。同时 V_0^x 中顶点，在 V_1^x 被取走后变成孤立点，也可以被删除，所以我们从图 G 中删除掉所有 V_1^x 和 V_0^x 中顶点以及与这些顶点相邻的所有边，得到：

$$G^* = (V^*, E^*),\ k^* = k - |V_1^x|$$

$$V^* = V_{1/2}^x$$

$$E^* = \{\{u,v\} \in E \mid u \in V^* \ and \ v \in V^*\}$$

我们已经成功将图 G 收缩成了图 G^*，接下来要考虑如何在快速构造出图 G^* 上 Dual-$LPVC(G)$ 模型的最优解。因为图 G^* 本质上是图 G 在顶点子集 $V^x_{1/2}$ 上的导出子图，很自然的我们会猜想，是不是将函数 x, y 定义域收缩为 V^* 和 E^* 后就分别是图 G^* 上 Primal-$LPVC(G)$ 与 Dual-$LPVC(G)$ 的最优解。接下来，我们对该猜想进行证明。

将函数 x, y 定义域分别限定为 V^* 以及 E^*，获得函数 $x^* : V^* \to \mathbb{R}$ 及 $y^* : E^* \to \mathbb{R}$，即对任意图 G^* 上的顶点 v，有 $x^*_v = x_v$；对任意图 G^* 上的边 e，有 $y^*_e = y_e$。

断言 3.1　x^*, y^* 分别是图 G^* 上 Primal-$LPVC(G)$, Dual-$LPVC(G)$ 的半整数最优解。

证明　因为 x, y 分别是图 G 上 Primal-$LPVC(G)$，Dual-$LPVC(G)$ 的半整数最优解，显然 x^*, y^* 也应该分别是图 G^* 上 Primal-$LPVC(G)$，Dual-$LPVC(G)$ 的半整数解。否则在图 G^* 上存在线形规划模型的约束条件不被函数 x^*（或 y^*）满足，而该条件在图 G 上也同样不被函数 x（或 y）不满足，这与 x（或 y）作为最优解冲突。

根据定义可以知道，x^* 是从 x 中移除掉定义域 $V^x_1 \cup V^x_0$ 后获得的，故 $val(x) - |V^x_1| = val(x^*)$。另外，$y^*$ 定义域相比 y 移除了与 V^x_1 相邻的边集，即

$$\begin{aligned} val(y) - val(y^*) &= \sum_{e \in (E \setminus E^*)} y_e \\ &= \sum_{e \in \delta(V^x_1)} y_e \le \sum_{v \in V^x_1} \sum_{e \in \delta(v)} y_e \le |V^x_1| \end{aligned}$$

结合, $val(y) = val(x)$, 可知 $val(y^*) \ge val(x^*)$。

因此 x^*, y^* 分别是图 G^* 上 Primal-$LPVC(G)$, Dual-$LPVC(G)$ 的解，且 $val(y^*) \ge val(x^*)$，根据引理3.4有 x^*, y^* 分别是图 G^* 上 Primal-$LPVC(G)$, Dual-$LPVC(G)$ 的最优解。

证毕。 ■

比较两个问题实例 (G, k) 和 (G^*, k^*)，根据引理3.2，可以很容易判断他们是等价的。对于现在新的 VCAL 上的问题实例 (G^*, k^*) 显然其参数相比于原实例 (G, k) 保持了不变，如下：

$$\begin{aligned} \mu(G^*, k^*) &= k^* - vc^*(G^*) \\ &= (k - |V^x_1|) - (vc^*(G) - |V^x_1|) \\ &= k - vc^*(G) = \mu(G, k) \end{aligned}$$

对于图 G^* 上求解 Dual-$LPVC(G^*)$ 的网络 W^*，因为图 G^* 是原来的图 G 的导出子图，所以根据定义3.4，网络 W^* 显然是原网络 W 的子网络。同时既然我们已经证明了 y^* 是图 G^* 上 Dual-$LPVC(G^*)$ 模型的最优解，回顾3.2.2节里，Dual-$LPVC$ 模型的最优解与对应网络最大可行流之间关系，可以知道将网络 W 的最大可行流 f 定义域限定到子网络 W^* 的边集上便是网络 W^* 的最大可行流。

3.3.2 核心化算法阶段二

因为在上一节中已经将 VCAL 问题实例 (G,k) 收缩至使得全 $\frac{1}{2}$ 函数是图 G 上 Primal-$LPVC(G)$ 模型的最优解（不保证唯一性），并且构造出对应的网络 W 的最大流 f。接下来在本阶段内，我们会对其问题进行进一步核心化，使得全 $\frac{1}{2}$ 函数成为图 G 上 Primal-$LPVC(G)$ 模型的**唯一半整数最优解**。

首先，我们给出该阶段算法的基本思路。

如果在当前图 G 满足 $surplus(G) = 0$，根据定义可以找到图 G 的某个独立集 $S \subseteq V$，使得 $surplus(S) = 0$。构造函数 $x' : V \to \mathbb{R}$，如下：

$$\begin{cases} x'_v = 0, & \text{对于顶点 } v \in S; \\ x'_v = 1, & \text{对于顶点 } v \in N(S); \\ x'_v = \frac{1}{2}, & \text{其他。} \end{cases}$$

显然 x' 也是 G 的 LPVC 最优解。类似上一阶段的做法，我们可以通过将 $V_1^{x'}$（即 $N(S)$）中顶点选进顶点覆盖集，来将问题实例从 (G,k) 收缩至 $(G[V \setminus (S \cup N(S))], k - |N(S)|)$。重复寻找满足条件的独立集 S，收缩，直到在当前图 G 中 $surplus(G) > 0$，则此时已经达到我们核心化的目标，全 $\frac{1}{2}$ 函数已经是图 G 上 Primal-$LPVC(G)$ 模型的唯一半整数最优解。

观察上述思路，我们发现将算法改进到线性时间复杂度的瓶颈在于如何快速的找到图 G 中满足 $surplus(S) = 0$ 的独立集 S。为了突破该瓶颈，我们需要借助网络 W 在最大可行流 f 下的残余图 $G_W(f)$。下面给出若干引理来揭示网络 W 在最大可行流 f 下的残余图 $G_W(f)$ 与满足条件的独立集之间的关系。

回忆网络 W 的构造（定义3.4），我们将原本图 G 中的每一个顶点 v，拆分成有向图 G_W 里的 l_v 和 r_v 两个顶点。为了描述的方便，此处引入几个新的集合的定义，假设 $S_W \subseteq V_W$ 是网络 W 中有向图 G_W 的顶点子集，令 $S_L = \{v \mid l_v \in S_W\}, S_R =$

$\{v \mid r_v \in S_W\}$。注意此处，S_W 是网络 W 中的顶点集合，而 S_L, S_R 是图 G 中的顶点集合。令 $L_W = \{l_v \in V_W\}, R_W = \{r_v \in V_W\}$。假设 $V' \subseteq V$ 是图 G 中顶点集合，另 $L_{V'} = \{l_v \in V_W \mid v \in V'\}, R_{V'} = \{r_v \in V_W \mid v \in V'\}$。注意此处，$L_W, R_W, L_{V'}, R_{V'}$ 均是网络 W 中的顶点集合。

引理 3.6 ([26])　对于网络 W 里有向图 G_W 的顶点子集 $S_W \subseteq (L_W \cup R_W)$, 以下两个条件等价：[1]

- （1）在网络 W 的残余图中 $G_W(f)$ 中不存在从 S_W 到 $(L_W \cup R_W) \setminus S_W$ 的边；
- （2）$N_G[S_L] = S_R$ 并且 $|S_L| = |S_R|$。

证明　对于当前的图 G，我们知道 $vc^*(G) = |V|/2$，网络W最大可行流 f 的流量 $F = 2vc^*(G) = |V|$。所以对于图 G 的任意顶点 $v \in V$，显然有 $f(s, l_v) = f(r_v, t) = 1$。假设 $V' \subseteq V$，可得流入 $L_{V'}$ 的流量等于流出 $R_{V'}$ 的流量等于 $|V'|$。回顾残余图的定义，其边有正向反向两种，这里我们也将残余图 $G_W(f)$ 中 S_W 到 $(L_W \cup R_W) \setminus S_W$ 的边分成两个集合，如下：

$$E_L = \{(u, v) \in E_W \mid u \in S_W, v \in (L_W \cup R_W) \setminus S_W\}$$

$$E_R = \{(v, u) \mid (u, v) \in E_W, f(u, v) > 0, u \in (L_W \cup R_W) \setminus S_W, v \in S_W\}$$

(1) → (2): 假设条件（1）成立，等价于 $E_L = E_R = \emptyset$。由 $E_L = \emptyset$，可以知道所有 S_L 中顶点的邻居都在 S_R 中，既 $N_G[S_L] \subseteq S_R$。由 $E_R = \emptyset$，可以知道没有从 S_W 外的顶点有流量流入 $S_W \cap R_W$ 中，所以从 $S_W \cap R_W$ 流出的流量应该小于等于流入 $S_W \cap L_W$ 中的流量，即 $|S_R| \leq |S_L|$。最后，从 $S_W \cap L_W$ 流出的流量显然不可能超过从 $R_{N_G[S_L]}$ 流到汇点 t 的流量，即 $|S_L| \leq |N_G[S_L]|$。综上有，$|S_L| = |N_G[S_L]| = |S_R|$，再结合前面得到的 $N_G[S_L] \subseteq S_R$，可以获得 $N_G[S_L] = S_R$。

(2) → (1): 假设 $N_G[S_L] = S_R$ 并且 $|S_L| = |S_R|$。由于 $N_G[S_L] = S_R$，那么 E_L 肯定是空集。因为 $|S_L| = |S_R|$，那所有流到 $S_W \cap R_W$ 的流量肯定都来自 $S_W \cap L_W$，那么也可以得到 E_R 也为空集。故在网络 W 的残余图中 $G_W(f)$ 中不存在从 S_W 到 $(L_W \cup R_W) \setminus S_W$ 的边。

证毕。 ■

观察上述引理3.6，我们发现条件 (2) 与是“S_L 是满足 surplus 值为 0 的独立集”的弱化条件，接下来我们对引理3.6里的两个条件中都增加约束，来获得更强的结论。

[1]该引理来自于文献 [26]，但是与之相比本文中我们加强了条件 2

注意上文已经提及，对于图 G 的任意顶点 $v \in V$，显然有 $f(s,l_v) = c(s,l_v) = f(r_v,t) = f(r_v,t) = 1$。所以在残余图 $G_W(f)$ 上，顶点 s,t 会构成单独两个非常特殊的强连通分量。其中强连通分量 $\{t\}$ 会连向所有其他强连通分量；所有其他强连通分量会连向强连通分量 $\{s\}$，而这两个强连通分量毫无意义。所以，后面我们讨论残余图 $G_W(f)$ 上强连通分量时候会排除掉这两个特殊的顶点。即，**实际上讨论的是有向图 $G_W(f) \setminus \{s,t\}$ 上的强连通分量**。

引理 3.7 ([26])　*如果网络 W 的流量残余图 $G_W(f)$ 中，有一个尾强连通分量 S_W 满足 $S_L \cap S_R = \emptyset$，那么 S_L 是图 G 中的一个独立集并且 $|S_L| = |N_G(S_L)|$。*

证明　S_W 是 $G_W(f)$ 中一个尾强连通分量，可以推导出 $G_W(f)$ 中不存在从 S_W 到 $(L_W \cup R_W) \setminus S_W$ 的边。故依据引理3.6，可以获得 $N_G[S_L] = S_R$ 并且 $|S_L| = |S_R|$。又因为 $S_L \cap S_R = \emptyset$，可以得知 $N_G(S_L) = N_G[S_L] = S_R$ 并且 S_L 是图 G 中一个独立集。证毕。 ■

引理 3.8 ([26])　*如果图 G 中存在独立集 $T \subseteq V$ 满足 $|T| = |N_G(T)|$, 那么网络 W 在最大可行流 f 下的残余图 $G_W(f)$ 中存在一个尾强连通分量 S_W 满足 $S_L \cap S_R = \emptyset$。*

证明　令 T 表示图 G 中满足 $|T| = |N_G(T)|$ 的最小独立集。构造 $S_W = L_T \cup R_{N_G(T)}$，根据定义 $S_L = T, S_R = N_G(T)$。因为 T 是图 G 中满足 $|T| = |N_G(T)|$ 的独立集，我们可以获得 $N_G(T) = N_G[T]$，即 $N_G[S_L] = S_R$。可以进一步整理得引理3.6中的条件 (2), $N_G[S_L] = S_R$ 并且 $|S_L| = |S_R|$。故网络 W 在最大可行流 f 下的残余图 $G_W(f)$ 中不存在从 S_W 到 $(L_W \cup R_W) \setminus S_W$ 的边。

假设 S_W 不是网络 W 在最大可行流 f 下的残余图 $G_W(f)$ 中的强连通分量，则存在 $S'_W \subset S_W$ 使得 $G_W(f)$ 中不存在从 S'_W 到 $S_W \setminus S'_W$ 的边。又因为之前已经发现 $G_W(f)$ 中不存在从 S_W 到 $(L_W \cup R_W) \setminus S_W$ 的边，两者结合可以进一步发现 $G_W(f)$ 中不存在从 S'_W 到 $(L_W \cup R_W) \setminus S'_W$ 的边。根据引理3.7，可以发现 S'_L 是图 G 中的一个独立集并且 $|S'_L| = |N_G(S'_L)|$。因为 S'_W 是 S_W 的真子集，故 $S'_L \subset S_L = T$，这与 T 是满足 $|T| = |N_G(T)|$ 的最小独立集矛盾。故假设不成立，S_W 是网络 W 的残余图 $G_W(f)$ 中的强连通分量。

综上，我们构造的顶点集合 S_W 是网络 W 的残余图 $G_W(f)$ 中的尾强连通分量，且满足 $S_L \cap S_R = \emptyset$。

证毕。 ■

引理3.7和引理3.8，为我们提供了寻找图 G 中满足 $surplus$ 值为 0 的独立集的方法。当我们从残余图 $G_W(f)$ 中找到一个尾强连通分量 S_W 满足 $S_L \cap S_R = \emptyset$，则可以判

断 S_L 是图 G 的独立集，且满足 $surplus(S_L) = 0$。通过将 $N(S_L)$ 中顶点选进顶点覆盖集，可以获得新的 VCAL 问题实例如下，

$$(G' \leftarrow G \setminus (S_L \cup N(S_L)),\ k' \leftarrow k - |N(S_L)|)$$

对于求解图 G' 上 Dual-$LPVC(G')$ 问题的网络 W'，同理于上一阶段的证明，我们知道网络 W' 是网络 W 的子网络，并且网络 W' 的最大可行流 f'，可以通过将原网络 W 的最大可行流 f 定义域缩小获得。现在剩下的唯一的问题就是，当前我们每一次找到一个满足盈余值为 0 的独立集，就需要重构一遍残余图，重新求解一次强连通分量。我们希望可以进一步优化，使得只求一次残余图的强连通分量便可以找出所有满足条件的独立集。为了解决该问题，我们需要分析当把某个顶点集 $N(S_L)$ 选进当前顶点覆盖集后，新的网络最大流残余图 $G_{W'}(f')$ 与之前的残余图 $G_W(f)$ 之间关系。

为了描述方便，令 $T = S_L$，则 $S_W = L_T \cup R_{N_G(T)}$。对比网络 W 与网络 W' 发现，除了移除顶点集合 $L_T \cup R_{N_G(T)}$，还移除了顶点集合 $L_{N_G(T)} \cup R_T$。如下图3-1当我们从残余图中找到尾强连通分量 $\{l_a, l_b, r_c, r_d\}$ 符合引理3.7，此时我们计划从图 G 移除顶点 a, b, c, d，则对应回网络 W 中除了上述尾强连通分量，还移除了另外 4 个顶点。实际上我们不需要重新计算新网络的残余图强连通分量，因为顶点集合 $\{l_c, l_d, r_a, r_b\}$ 在原残余图中对应着一个头强连通分量。后续给出断言3.2并证明该结论。

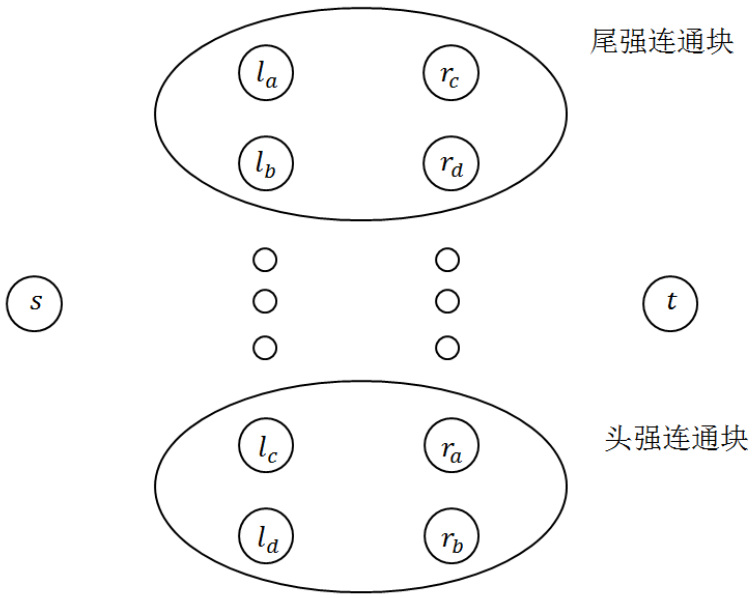

图 3-1 残余图中尾强连通分量与对应的头强连通分量

断言 3.2　在残余图 $G_W(f)$ 中没有边从顶点集合 $(L_W \cup R_W) \setminus (L_{N_G(T)} \cup R_T)$ 到顶点集合 $L_{N_G(T)} \cup R_T$ 中。

证明　首先我们将残余图 $G_W(f)$ 中从顶点集合 $L_{N_G(T)} \cup R_T$ 到顶点集合 $(L_W \cup R_W) \setminus (L_{N_G(T)} \cup R_T)$ 中边分成两个集合，注意在网络 W 中，排除 s,t 后，只有从 L_W 到 R_W 的边。

$$E_L = \{(u,v) \in E_W \mid u \in L_W \setminus L_{N_G(T)}, v \in R_T\}$$
$$E_R = \{(v,u) \mid (u,v) \in E_W, f(u,v) > 0, u \in L_{N_G(T)}, v \in R_W \setminus R_T\}$$

回顾定义3.4，显然 E_L 为空集。同时由于 $|R_T| = \left|L_{N_G(T)}\right|$，且 E_L 为空，所以所有从集合 $L_{N_G(T)}$ 流出的流量都会流入集合 R_T 中，因此 E_R 也是空集。

故，$E_L = E_R = \emptyset$，证毕。 ■

对于残余图 $G_W(f)$ 的所有强连通分量，他们之间构成了拓扑关系。而新的残余图 $G_{W'}(f')$ 与之对比，移除掉的强连通分量要不就在拓扑关系的顶端（没有外部边可以到达顶点集合 $L_{N_G(T)} \cup R_T$ 中）要不就在拓扑关系的末端（顶点集合 $L_T \cup R_{N_G(T)}$ 是一个尾强连通分量），而对于剩余强连通分量，强连通性以及拓扑关系都没有受到影响！因此只需要在求出残余图 $G_W(f)$ 的所有强连通分量之后，利用逆拓扑顺序遍历每一个强连通分量 S_W，检查该强连通分量是否不存在未被标记移除的后代以及是否满足 $S_L \cap S_R = \emptyset$，如果两个条件都成立，则将顶点集合 $L_T \cup R_{N_G(T)}$ 与顶点集合 $L_T \cup R_{N_G(T)}$ 中的强连通分量都标记为移除即可。

至此我们给出了核心化算法阶段二的所有证明，下面分析时间复杂度。在整个核心化算法阶段二，其中我们只需要构造一个残余图，求出该残余图的所有强连通分量，最后逆拓扑顺序遍历这些强连通分量。显然每个步骤都是可以做到线性时间复杂度的，故整体时间复杂度为 $O(|V| + |E|)$。

3.3.3 完整核心化算法

伪代码详见 Algorithm 1。

其中，我们将 VCAL 问题实例以及预先构造好的网络及其最大流作为输入，并且在核心化的过程中也将新图对应的网络及其最大流构造好并输出。

Algorithm 1 完整核心化算法

Input: VCAL 问题实例 (G, k), 图 G 对应的网络 W 及其最大流 f

Output: 核心化后的 VCAL 问题实例 (G', k')，图 G' 对应的网络 W' 及其最大流 f'，被取进顶点覆盖的顶点集合 S_1, ，被排除出顶点覆盖的顶点集合 S_0

function KERNELISE(G, k, W, f)
  $S_0 \leftarrow \emptyset$
  $S_1 \leftarrow \emptyset$
  ▷ 核心化算法阶段一
  根据网络 W 在最大可行流 f，构造图 G 的 Primal-$LPVC(G)$ 最优解 x
  $G_1 \leftarrow G[V^x_{1/2}]$, $k_1 \leftarrow k - |V^x_1|$
  $S_1 \leftarrow S_1 \cup V^x_1$
  $S_0 \leftarrow S_0 \cup V^x_0$
  $W_1 \leftarrow$ 从网络 W 中移除顶点集合$\{l_v, r_v | v \in V^x_1 \cup V^x_0\}$以及相关边
  $f_1 \leftarrow$ 将流量函数 f 定义域限定到网络 W' 的边集上
  ▷ 核心化算法阶段二
  构造网络 W_1 在最大可行流 f_1 下的流量残余图 $G_{W_1}(f_1)$
  $K \leftarrow$ 求解图 $G_{W_1}(f_1) \setminus \{s, t\}$ 的所有强连通分量
  for 按逆拓扑顺序遍历 K 中每一个强连通分量 $T \in K$ **do**
    if 强连通分量 T 不存在未被标记的后代 **and** $\{v \mid l_v \in T, r_v \in T\} = \emptyset$ **then**
      $S_0 \leftarrow S_0 \cup \{v \mid l_v \in T\}$
      $S_1 \leftarrow S_1 \cup \{v \mid r_v \in T\}$
      标记强连通分量 T
    end if
  end for
  $G_2 \leftarrow G \setminus (S_0 \cup S_1)$, $k_2 \leftarrow k - |S_1|$
  $W_2 \leftarrow$ 从网络 W 中移除顶点集合$\{l_v, r_v | v \in S_1 \cup S_0\}$以及相关边
  $f_2 \leftarrow$ 将流量函数 f 定义域限定到网络 W_2 的边集上
  return $(G_2, k_2, W_2, f_2, S_1, S_0)$
end function

3.4 VCAL 问题的分支策略

上一章中，已经给出了基于线性松弛下界的参数化顶点覆盖问题（VCAL）的核心化算法，将图 G 在线性时间内收缩至满足全 $\frac{1}{2}$ 函数是其 LPVC 唯一最优解（即 $surplus(G) \geq 1$）。本章之中，我们将给出 3 条分支规则作用于核心化后的图上，这些规则组成了本章的分支算法。对于每一条分支规则，我们保证其都可以在线性时间内测试是否适用，并且如果决定执行，执行时间复杂度也是线性的。最后证明在这些分支规则的共同作用下整棵搜索树的节点数量不超过 2.618^{μ}。

为了更好的分析分支算法的时间复杂度，首先给出以下引理：

引理 3.9　假设图 G 满足 $surplus(G) \geq 1$，且对于求解 Dual-$LPVC(G)$ 模型的网络 W 已经求得其最大流 f。令 $G' = G \setminus S$，则可以在 $O(|S|(|V| + |E|))$ 的时间复杂度内获得 G' 上求解 Dual-$LPVC(G')$ 模型的网络 W' 上的最大流 f'。

证明　回顾3.2.2中内容，对于网络 W' 其有向图 $G_{W'}$ 等价于有向图 G_W 移除顶点集合 $L_S \cup R_S$ 以及相应的边集，即 $G_{W'} = G_W \setminus \{L_S \cup R_S\}$。因此如果网络 W 上一个可行流没有流量经过顶点集合 $L_S \cup R_S$，则将其定义域限制为 $E_{W'}$ 后便是网络 W' 上的可行流。

对网络 W 的最大流 f 进行退流操作，移除所有经过顶点集合 $L_S \cup R_S$ 的流量，因为在有向图 G_W 只有从 s 到 L_W，从 L_W 到 R_W 以及从 R_W 到 t 的有向边，所以该操作可以在 $O(|V| + |E|)$ 的时间复杂度内完成。将退流后的可行流流量函数定义域限定为 $E_{W'}$，并命名为 f'。显然 f' 是网络 W' 上的可行流。比较两个可行流的流量，可以得 $F' \geq F - 2|S|$。

对当前网络 W' 的可行流 f' 使用 Ford-Fulkerson 算法进行增广，假设增广 Δ 流量后 f' 成为最大可行流，则增广的时间复杂度是 $O(\Delta(|V| + |E|))$。因为网络 W' 中到达汇点 t 的边的容量和为 $|V'| = F - |S|$，所以最大流流量 $F' + \Delta \leq F - |S|$。整理可以获得 $\Delta \leq |S|$。综上，将可行流 f' 增广至最大流的时间复杂度为 $O(|S|(|V| + |E|))$。

证毕。 ■

3.4.1 分支策略

在分支算法开始之前，首先我们在图 G 中任意取一个顶点 $v \in V$，由于 $surplus(G) \geq 1$，顶点 v 至少有两个邻居，取其中任意两个，记为 u_1, u_2。

对于图 G 中的任何一个顶点覆盖，显然或者包含了顶点 v, 或者同时包含了顶点 u_1, u_2。很自然，我们会考虑产生两个搜索分支，使得其中一个将顶点 v 取进顶点覆盖集合，另外一个将顶点 u_1, u_2 取进。为了限制搜索树大小，这里我们增加了一个前提条件，要求在图 $G \setminus \{u_1, u_2\}$ 上全 $\frac{1}{2}$ 函数是其 Primal-$LPVC(G \setminus \{u_1, u_2\})$ 模型的最优解，即 $vc^*(G \setminus \{u_1, u_2\}) = (|V| - 2)/2$。

分支规则一

前提： 在图 $G \setminus \{u_1, u_2\}$ 上，$vc^*(G \setminus \{u_1, u_2\}) = (|V| - 2)/2$

分支一： $(G_1 \leftarrow G \setminus \{v\},\ k_1 \leftarrow k - 1)$

分支二： $(G_2 \leftarrow G \setminus \{u_1, u_2\},\ k_2 \leftarrow k - 2)$

根据引理3.9，显然测试分支规则一的前提是否成立，更新分支一、分支二的问题实例的 $LPVC$ 问题最优解均可以在线性时间内完成。

之后我们考虑分支规则一不被执行的情形，即发现 $vc^*(G \setminus \{u_1, u_2\}) < (|V| - 2)/2$。此时我们对于图 $G \setminus \{u_1, u_2\}$ 执行3.3节给出的核心化算法，设 G' 为图 $G \setminus \{u_1, u_2\}$ 收缩后得到的新图，且已经被核心化算法取进顶点覆盖集的顶点集合为 $V_1 \subseteq V$，已经被标记不属于顶点覆盖集合的顶点集合为 $V_0 \subseteq V$。

断言 3.3　对于顶点集合 V_1, V_0 有

- （1）$|V_1| = |V_0| - 1$;
- （2）顶点集合 $V_0 \subseteq V$ 是图 G 的一个独立集，且 $N_G(V_0) = V_1 \cup \{u_1, u_2\}$。

证明　（1）构造一个图 $G \setminus \{u_1, u_2\}$ 上 Primal-$LPVC(G \setminus \{u_1, u_2\})$ 模型的解 $x^* : V \setminus \{u_1, u_2\} \to \{0, \frac{1}{2}, 1\}$ 如下，

$$\begin{cases} x_v^* = 0, & \text{对于顶点 } v \in V_0; \\ x_v^* = 1, & \text{对于顶点 } v \in V_1; \\ x_v^* = \frac{1}{2}, & \text{其他。} \end{cases}$$

由3.3节给出的核心化算法定义，显然这样构造出来的 x^* 是 Primal-$LPVC(G \setminus \{u_1, u_2\})$ 的最优解。又因为此时分支规则一不能应用，可得：$vc^*(G \setminus \{u_1, u_2\}) = val(x^*) = |V_1| + (|V| - |V_1| - |V_0| - 2)/2 < (|V| - 2)/2$，化简得：$|V_1| < |V_0|$。

构造一个图 G 上 Primal-$LPVC(G)$ 模型的解 $x : V \to \{0, \frac{1}{2}, 1\}$ 如下，

$$\begin{cases} x_v = 0, & \text{对于顶点 } v \in V_0; \\ x_v = 1, & \text{对于顶点 } v \in V_1 \cup \{u_1, u_2\}; \\ x_v = \frac{1}{2}, & \text{其他。} \end{cases}$$

又因为已知全 $\frac{1}{2}$ 函数是其图 G 上 Primal-$LPVC(G)$ 唯一最优解，可得，$val(x) = |V_1| + 2 + (|V| - |V_1| - |V_0| - 2)/2 \geq vc^*(G) + 1/2 = (|V| + 1)/2$。化简得：$|V_1| \geq |V_0| - 1$。

综上有 $|V_1| = |V_0| - 1$。

（2）V_0 在图 $G \setminus \{u_1, u_2\}$ 中是独立集，显然其在图 G 中也是独立集。假设 u_1 或者 u_2 不在 $N_G(V_0)$ 中，则有 $surplus(G) \leq surplus_G(V_0) \leq |V_1| + 1 - |V_0| = 0$。这与已知 $surplus(G) \geq 1$ 矛盾。故 $u_1, u_2 \in N_G(V_0)$，即 $N_G(V_0) = V_1 \cup \{u_1, u_2\}$。

证毕。 ■

后面根据 $V_1 \cup \{u_1, u_0\}$ 是否图 G 中的独立集进行分类讨论。

在 $V_1 \cup \{u_1, u_0\}$ 不是图 G 中的一个独立集的情况下，应用分支规则二如下：

分支规则二

前提： 顶点集合 $V_1 \cup \{u_1, u_0\} \subseteq V$ 不是图 G 中的一个独立集

分支一： $(G_1 \leftarrow G \setminus (V_0 \cup V_1 \cup \{u_0, u_1\}),\ k_1 \leftarrow k - |V_1| - 2)$

分支规则二实际上没有产生多个搜索分支，但是为了更方便分析搜索树大小这里将其归入其中。首先我们给出一个引理来证明它的正确性。

引理 3.10　对于图 $G(V, E)$，假设 $surplus(G) \geq 1$ 且存在独立集 $S \subseteq V$ 满足 $surplus(S) = 1$ 和 $N(S)$ 并不是独立集，则存在一个图 G 的最小顶点覆盖，其包含所有 $N(S)$ 中顶点且不包含所有 S 中顶点。

证明　令 $G' = G[S \cup N(S)]$，假设 $vc(G') \leq |S|$。令顶点集合 VC' 是图 G' 的最小顶点覆盖集之一，将其分解为 $VC' = A \uplus B$，其中 $A \subseteq S$ 且 $B \subseteq N(S)$。

因为 G' 中不存在孤立点且 S 是独立集，所以 S 中不被 A 包含的顶点肯定每个都与集合 B 相邻，即 $N(S \setminus A) \subseteq B$。由 $N(S)$ 并非图 G 中的一个独立集，可以得知 $B \neq \emptyset$，再考虑 $|A| + |B| = |VC'| \leq |S|$，故 $|A| < |S|$，因此 $S \setminus A$ 显然也非空。考虑独

立集 $S \setminus A$ 的盈余值 $surplus(S \setminus A) \leq |B| - (|S \setminus A|) \leq 0$，这与 $surplus(G) \geq 1$ 矛盾。所以 G' 的最小顶点覆盖集大小大于 $|S|$。

又因为我们可以获得一个大小为 $|S|+1$ 的顶点覆盖集合（$N(S)$ 就是这样一个顶点集合），综上 $vc(G') = |S| + 1$。

通过上文的图 G' 的构造我们知道任意图 G 的最小顶点覆盖至少包含 $|S|+1$ 个集合 $S \cup N(S)$ 中的顶点。假设顶点集合 VC 是图 G 的一个最小顶点覆盖集，我们可以构造一个新的顶点集合如下 $VC^* = VC \setminus (S \cup N(S)) \cup N(S)$。显然 VC^* 也会是图 G 的一个顶点覆盖集合，并且它的大小不会超过 VC。综上 VC^* 是图 G 的最小顶点覆盖，其包含所有 $N(S)$ 中顶点且不包含所有 S 中顶点。证毕。 ■

再回顾分支规则二，首先在我们应用分支规则的图 G 上有 $surplus(G) \geq 1$，其次 V_0 是图 G 上盈余值为 1 的一个独立集，最后我们只在 V_0 的邻居顶点集合 $V_1 \cup \{u_1, u_2\}$ 不是独立集的前提下应用分支规则二。所以，根据引理3.10，此时存在图 G 的最小顶点覆盖集合包含所有 $V_1 \cup \{u_1, u_2\}$ 中顶点，且不包含所有 V_0 中顶点。至此成功证明了分支规则二的正确性，我们可以在不能应用分支规则一并且 $V_1 \cup \{u_1, u_2\}$ 不是图 G 中的一个独立集的情况下，先将 $V_1 \cup \{u_1, u_2\}$ 中顶点选进顶点覆盖集合中。

接下来分析分支规则二的时间复杂度。显然应用核心化算法在图 $G \setminus \{u_1, u_2\}$ 上，和判断顶点集合 $V_1 \cup \{u_1, u_2\}$ 是否图 G 中的独立集均可以在 $O(|V| + |E|)$ 时间复杂度内完成。对于分支规则二产生的图 G_1，其等价于前文中我们对图 $G \setminus \{u_1, u_2\}$ 应用核心化算法收缩出来的新图 G'，在核心化算法中，已经获得了其对应的求解 Dual-$LPVC(G')$ 模型的网络的最大流。所以不需要额外时间复杂度去求解该网络的最大流，故分支规则二，也可以在线性时间内应用。

最后讨论剩余的情况，此时 $V_1 \cup \{u_1, u_2\}$ 是图 G 中的一个独立集。

因为 $surplus_G(V_1 \cup \{u_1, u_2\}) \geq surplus(G) \geq 1$，所以存在独立集 $V_1 \cup \{u_1, u_2\}$ 的邻居不属于点集 V_0，这里我们取其中一个这样子的邻居，设为 v'，同时将某个与 v' 相邻的 $V_1 \cup \{u_1, u_2\}$ 中的顶点记为 u'_1。显然 v' 是在图 G' 上的，由于图 G' 满足 $surplus(G') \geq 1$，所以 $N_{G'}(v') \neq \emptyset$，取其中一个顶点 $u'_2 \in N_{G'}(v')$。

分支规则三

前提： 顶点集合 $V_1 \cup \{u_1, u_0\} \subseteq V$ 是图 G 中的一个独立集

分支一： $(G_1 \leftarrow G \setminus \{v'\},\ k_1 \leftarrow k - 1$

分支二： $(G_2 \leftarrow G \setminus (V_1 \cup V_0 \cup \{u_1, u_2, u'_2\}),\ k_2 \leftarrow k - |V_1| - 3)$

为了证明分支规则三的正确性，同样我们此处也引进一个新的引理，如下：

引理 3.11　对于图 $G(V,E)$，假设 $surplus(G) \geq 1$ 且存在独立集 $S \subseteq V$ 满足 $surplus(S) = 1$ 和 $N(S)$ 是独立集，如果存在一个图 G 的最小顶点覆盖，其包含部分 $N(S)$ 中顶点，那么也一定存在包含所有 $N(S)$ 中顶点的图 G 的最小顶点覆盖。

证明　令顶点集合 VC 是图 G 的最小顶点覆盖集之一，将其分解为 $VC = A \uplus B \uplus C$，其中 $A \subseteq S$，$B \subseteq N(S)$ 且 $C \cap (S \cup N(S)) = \emptyset$。显然当 $A = \emptyset$ 时，$B = N(S)$；当 $B = \emptyset$ 时，$A = S$。因此我们只需要证明当顶点集合 A, B 同时非空时，可以构造出一新的图 G 的最小顶点覆盖包含所有 $N(S)$ 中顶点。

假设顶点集合 A, B 同时非空。同理于引理3.10的证明，可以得知此时 $|A| + |B| \geq |S| + 1$。构造一新的顶点集合 $VC' = N(S) \cup C$，显然 VC' 是图 G 的一个顶点覆盖，又因为 $|VC'| = |C| + |S| + 1 \leq |A| + |B| + |C| = |VC|$，所以它也是图 G 的一个最小顶点覆盖。

证毕。 ■

由于顶点 u_1', u_2' 均是顶点 v' 的邻居，对于图 G 的最小顶点覆盖肯定满足（1）或者包含顶点 v'；（2）或者包含顶点 u_1', u_2'。根据这一点，类似于分支规则一我们也可以分裂出两个搜索分支。再考虑在分支规则一与分支规则二均不能应用的情况下，首先在我们应用分支规则的图 G 上已经有了 $surplus(G) \geq 1$；其次当分支规则一不能应用时候，V_0 是图 G 上盈余值为 1 的一个独立集；最后当分支规则二不能应用时候，V_0 的邻居顶点集合 $V_1 \cup \{u_1, u_2\}$ 是图 G 的一个独立集。所以依据引理3.11，假设存在包含顶点 $u_1' \in V \cup \{u_1, u_2\}$ 的最小顶点覆盖，则肯定也存在包含所有 $V \cup \{u_1, u_2\}$ 中顶点的最小顶点覆盖。所以在分支二将 u_1', u_2' 提前取进顶点覆盖集，可以转化成将 $V_1 \cup \{u_1, u_2, u_2'\}$ 中所有顶点取进顶点覆盖集。至此我们成功证明了分支规则三的正确性。

接下来我们分析执行分支规则三的时间复杂度。根据我们的分类讨论情形，当分支规则一和二不能执行的时候，总会执行分支规则三，因此不需要额外测试是否执行该规则。注意在利用核心化算法构造 G' 的时候，已经求得其 Dual-$LPVC(G')$ 模型的最优解。同时分支一的图 G_1 是从图 G 中移除一个顶点 v' 获得的，分支二的图 G_2 是从图 G' 中移除顶点 u_2' 获得的，所以依据引理3.9，我们分别可以在求解 Dual-$LPVC(G)$，Dual-$LPVC(G')$ 模型的网络的最大可行流的基础上，在 $O(|V| + |E|)$ 的时间复杂度内构造两个分支的对应的最大可行流。故，分支规则三也能在线性时间复杂度内被执行。

3.4.2 算法时间复杂度分析

在上一节中，我们已经给出了 3 个分支规则，并且证明每个规则的正确性以及保证他们都能在线性时间内被执行。接下来只剩下讨论搜索树大小。这里我们利用参数 μ 的下降来进行分析。

对于我们核心化算法来说，在给出的时候已经证明其保证收缩后的 VCAL 问题实例 (G', k') 相比原来的 VCAL 问题实例 (G, k)，有：$\mu(G', k') \leq \mu(G, k)$。现在讨论每个分支规则应用后产生的每个分支的问题实例的参数，对比应用前的参数的变化。

首先回顾上一节给出的分支规则一。我们通过或者使顶点覆盖集合包含顶点 v, 或者使顶点覆盖集合包含顶点 u_1, u_2，在原来 VCAL 问题实例的基础上产生了两个搜索分支。这里对每个分支上的新问题实例的参数分别进行分析。

对于分支一，由于 $surplus(G) \geq 1$，所以删除掉一个顶点后的图 G_1 上有 $surplus(G_1) \geq 0$。根据引理3.3，可以得知全 $\frac{1}{2}$ 函数是图 G_1 上的 Primal-$LPVC(G_1)$ 模型的最优解，即 $vc^*(G_1) = \frac{|V|-1}{2}$。所以有 $\mu(G_1, k_1) = k_1 - vc^*(G_1) = k - 1 - \frac{|V|-1}{2} = \mu(G, k) - \frac{1}{2}$。

对于分支二，由分支规则一应用的前提可以得知 $vc^*(G_2) = \frac{|V|-2}{2}$。所以有 $\mu(G_2, k_2) = k_2 - vc^*(G_2) = k - 2 - \frac{|V|-2}{2} = \mu(G, k) - 1$。

综上，这里更新分支规则一的描述，如下：

分支规则一：

前提：	在图 $G \setminus \{u_1, u_2\}$ 上，$vc^*(G \setminus \{u_1, u_2\}) = (\|V\| - 2)/2$
分支一：	问题实例：　$(G_1 \leftarrow G \setminus \{v\},\ k_1 \leftarrow k - 1)$ 参数：　$\mu(G_1, k_1) = \mu(G, k) - \frac{1}{2}$
分支二：	问题实例：　$(G_2 \leftarrow G \setminus \{u_1, u_2\},\ k_2 \leftarrow k - 2)$ 参数：　$\mu(G_2, k_2) = \mu(G, k) - 1$

再考虑分支规则二。当分支规则一不能执行的时候，我们利用核心化算法对图 $G \setminus \{u_1, u_2\}$ 进行收缩。获得了算法收缩后的新图 G'，被算法取进该图顶点覆盖集的顶点集合 V_1 以及被算法排除出该图顶点覆盖集合的顶点集合 V_0。而当顶点集合 $V_1 \cup \{u_1, u_2\}$ 不是图 G 的独立集时候，我们应用分支规则二，产生了一个新的搜索分支。

对于分支规则二产生的唯一搜索分支，由于应用核心化算法得到的新图 G' 满足全 $\frac{1}{2}$ 函数是图 G' 上的 Primal-$LPVC(G')$ 模型的唯一最优解，所以 $vc^*(G_1) = \frac{|V|-|V_0|-|V_1|-2}{2}$。另外已经证明了 $|V_1| = |V_0| - 1$。故，

$$\begin{aligned}\mu(G_1, k_1) &= k_1 - vc^*(G_1) \\ &= k - |V_1| - 2 - \frac{|V| - |V_0| - |V_1| - 2}{2} \\ &= k - (|V_0| - 1) - 2 - \frac{|V| - |V_0| - (|V_0| - 1) - 2}{2} \\ &= k - \frac{|V|}{2} - \frac{1}{2} = \mu(G, k) - \frac{1}{2}\end{aligned}$$

更新分支规则二描述，如下；

分支规则二：

前提：	顶点集合 $V_1 \cup \{u_1, u_0\} \subseteq V$ 不是图 G 中的一个独立集
分支一：	问题实例：　$(G_1 \leftarrow G \setminus (V_0 \cup V_1 \cup \{u_0, u_1\}),\ k_1 \leftarrow k - |V_1| - 2)$ 参数：　　$\mu(G_1, k_1) = \mu(G, k) - \frac{1}{2}$

最后讨论分支规则三。当分支规则一和二不满足应用条件的时候，该分支规则会被应用，并且类似规则一，也会分裂出两个新的搜索分支。

对于分支一，同理于分支规则一的分支一，可以得知 $\mu(G_1, k_1) = \mu(G, k) - \frac{1}{2}$。对于分支二，注意图 G_2 是从图 G' 中移除顶点 u_2' 获得的，而且图 G' 满足全 $\frac{1}{2}$ 函数是图 G' 上的 Primal-$LPVC(G')$ 模型的最唯一优解，所以 $surplus(G') \geq 1$，$surplus(G_2) \geq 0$。因此我们知道 $vc^*(G_2) = (|V| - |V_0| - |V_1| - 3)/2$。故，

$$\begin{aligned}\mu(G_2, k_2) &= k_2 - vc^*(G_2) \\ &= k - |V_1| - 3 - \frac{|V| - |V_0| - |V_1| - 3}{2} \\ &= k - |V_1| - 3 - \frac{|V| - (|V_1| + 1) - |V_1| - 3}{2} \\ &= k - \frac{|V|}{2} - 1 = \mu(G, k) - 1\end{aligned}$$

更新分支规则三描述，如下；

分支规则三：

前提：	顶点集合 $V_1 \cup \{u_1, u_0\} \subseteq V$ 不是图 G 中的一个独立集
分支一：	问题实例：　$(G_1 \leftarrow G \setminus \{v'\},\ k_1 \leftarrow k-1)$ 参数：　$\mu(G_1, k_1) = \mu(G, k) - \frac{1}{2}$
分支二：	问题实例：　$(G_2 \leftarrow G \setminus (V_1 \cup V_0 \cup \{u_1, u_2, u_2'\}),\ k_2 \leftarrow k - \lvert V_1\rvert - 3)$ 参数：　$\mu(G_2, k_2) = \mu(G, k) - 1$

定理 3.1　整个算法可以在 $O(2.618^{\mu}(|V|+|E|))$ 的时间复杂度内求解顶点覆盖问题。

证明　由于我们的核心化算法保证问题实例的参数 μ 不会增加，同时核心化算法和所有的分支规则都可以在 $O(|V|+|E|)$ 时间复杂度内完成，所以为了证明该定理，只需要证明搜索树的节点个数不超过 $O(2.618^{\mu})$。

假设对于参数为 μ 的问题实例，执行该分支算法遍历的搜索树节点数量为 $T(\mu)$。回顾我们三个分支规则，可以获得不等式，如下

$$T(\mu) \leq T(\mu - \frac{1}{2}) + T(\mu - 1)$$

解得 $T(\mu) \leq 2.618^{\mu}$。

综上，搜索树的节点个数不超过 $O(2.618^{\mu})$，每个节点只需要 $O(|V|+|E|)$ 的时间复杂度，总时间复杂度为 $O(2.618^{\mu}(|V|+|E|))$。 ■

3.4.3 伪代码

在分支算法中我们将算法分解成两个函数，首先给出引理3.9对应伪代码，详见 Algorithm 2。在 Algorithm 2 中，我们执行的是对网络进行退流、删点以及增广 3 个步骤，在引理3.9的证明中已经告诉我们这整个过程的时间负责度是删除顶点个数乘以图的规模。

其次，我们给出完成的 VCAL 问题的分支搜索算法，详见 Algorithm 3。这也是本章的主要算法，其中会调用 Algorithm 1 和 Algorithm 2，如第 2、9、15 行。还有一点要注意，该算法是一个递归算法其中有 3 种调用自己的方式，分别对应着本章中我们的三条分支规则。

Algorithm 2 引理3.9算法

Input: 图 G 对应的网络W及其最大流 f，
需要移除的顶点集合 S

Output: 图 $G \setminus S$ 对应的网络 W' 及其最大流 f'

1: **function** MAINTAINMF(W, f, S)
2: $f' \leftarrow$ 在最大流 f 的基础进行退流操作，移除所有经过顶点集合 $\{l_v, r_v \mid v \in S\}$ 的流量
3: $W' \leftarrow$ 在网络 W 的基础上移除顶点集合 $\{l_v, r_v \mid v \in S\}$ 以及相应的边
4: $f' \leftarrow$ 将网络 W' 可行流 f' 增广至最大流
5: **return** W', f'
6: **end function**

3.5 小结

至此，本章中完整地给出了一个求解基于线性松弛下界的顶点覆盖问题（VCAL）的线性固定参数算法。通过运用最大网络流进行核心化的技巧，以及更加细致地分支策略讨论，我们的算法时间复杂度到达了 $O(2.618^{\mu}(|V|+|E|))$，较大地改进了之前 $O(4^{\mu}(|V|+|E|))$ 的最好结果。

下一步关于此问题的研究工作，我们将主要考虑能不能将时间复杂度继续提升到 $O(2.314^{\mu}(|V|+|E|))$。此外当前我们的线性固定参数算法并不能直接移植到带权的顶点覆盖问题上，这也是需要继续改进地地方。

Algorithm 3 完整求解 VCAL 问题的分支算法

Input: VCAL 问题实例 (G, k), 图 G 对应的网络W及其最大流 f

Output: VCAL 问题实例 (G, k) 的答案是"YES" 或者 "NO"

function VCAL-All(G, k, W, f)
  $(G, k, W, f, VC_1, VC_0) \leftarrow Kernelise(G, k, W, f)$　▷ 调用核心化算法进行核心化
  $\mu \leftarrow k - F(f)/2$
  if $\mu < 0$ **then return** False　▷ 搜索的终止条件
  else if 图 G 中已经没有顶点 **then return** True
  end if
  从图 G 中任意取一个顶点 $v \in V(G)$　▷ 分支规则一
  从 v 的邻居 $N(v)$ 中任意取两个顶点 $u_1, u_2 \in N(v)$
  $W', f' \leftarrow MaintainMF(W, f, \{u_1, u_2\})$
  if $F(f') = |V| - 2$ **then**
    $branchA \leftarrow$ VCAL-All$(G \setminus \{v\}, k-1, MaintainMF(W, f, \{v\}))$
    $branchB \leftarrow$ VCAL-All$(G \setminus \{u_1, u_2\}, k-2, W', f'))$
    return $branchA \vee branchB$
  end if
  $(G', k', W', f', V_1, V_0) \leftarrow Kernelise(G \setminus \{u_1, u_2\}, k-2, W', f')$
  $V_1 \leftarrow V_1 \cup \{u_1, u_2\}$
  if V_1 不是图 G 中的独立集 **then**　▷ 分支规则二
    return VCAL-All(G', k', W', f')
  end if
  从图 G' 中任意取一个顶点 $v' \in N_G(V_1)$　▷ 分支规则三
  从图 G' 中任意取一个顶点 $u'_2 \in N_{G'}(v')$
  $branchA \leftarrow$ VCAL-All$(G \setminus \{v'\}, k-1, MaintainMF(W, f, \{v'\}))$
  $branchB \leftarrow$ VCAL-All$(G' \setminus \{u'_2\}, k'-1, MaintainMF(W', f', \{u'_2\}))$
  return $branchA \vee branchB$
end function

第 四 章 反馈顶点集问题的固定参数算法

反馈顶点集问题 (Feedback Vertex Set, 缩写为 FVS), 即对于参数 k，询问给定图中是否存在大小满足 k 的顶点集合满足：对于图中任意一条环都与这个顶点集有交集。与顶点覆盖问题类似，该问题也是最早被提出来的 21 个 NP 完全问题之一[28]。

本章里对于该问题的经典参数化版本进行了研究，并且提出了一个时间复杂度为 $\mathcal{O}^*(3.598^k)$ 的固定参数算法。该算法与 Kociumaka 的算法[32] 一样，都是在之前 Cao 和 Chen 等的算法[8] 的算法的基础上进行优化。其中，我们与这两个文献最主要的区别是我们引入了双参数来评估 Disjoint-FVS 问题实例[1]。

4.1 相关术语及问题的定义

本节中，我们定义了算法中会使用的图论相关的术语[2] 并且引进了部分前人已经证明的结论。此外对于算法所解决的问题也给予了更加数学的全面的定义。

4.1.1 有关图的一些术语的定义

首先给出无向图上一些常见名词的定义，如环，树和森林等。

定义 4.1 (**环**) 对于无向图 $G(V,E)$，假设存在一个顶点序列 $v_1v_2...v_n$，使得 $\{v_0,v_n\}\in E$ 并且对于任意 $0\le i<n$ 有 $\{v_i,v_{i+1}\}\in E$。则该序列为图 G 上的一个**环**。如果该序列中没有重复顶点，则称之为图 G 的一个**简单环**。

定义 4.2 (**树** (Tree) **和森林** (Forest)) 对于无向图 $G(V,E)$，如果其中不存在任何环，则我们称图 G 为一个**森林 (forest)**。同时如果图 G 是连通的，则其也被称为**树 (tree)**。

定义 4.3 (**反馈顶点集**) 对于图 $G(V,E)$, 若集合 $V'\subseteq V$ 使得 $G\setminus V'$ 中不存在任何环（即，图 $G\setminus V'$ 是森林）, 则称 V' 是图 G 的一个**反馈顶点集**（Feedback Vertex Set, 简称 FVS）。

[1]Disjoint-FVS 问题是在原 FVS 问题上应用迭代压缩技术后出现的一个中间问题，后文中会给出详细定义。

[2]注意本章里我们会继续沿用上一大章中关于图的符号和术语定义，对于上一章已经定义过的术语，我们不会在本节中重复定义，而是直接使用。

为了方面对于算法的描述，我们继续引进一些图的术语。对于图 G，$cc(G)$ 表示图 G 上连通块的个数。

4.1.2 参数化反馈顶点集问题

对于一般图来说，回答是否存在大小不超过 k 的反馈顶点集，是非常经典的组合优化问题。在该问题上，固定参数算法是常用来求解答案的方法。具体参数化反馈顶点集问题表述如下：

参数化反馈顶点集问题 (Feedback Vertex Set，简称 FVS)
输入： 图 $G(V,E)$ 及整数 k
参数： k
问题： 图 G 中是否存在一个反馈顶点集合，其大小不超过 k

在参数计算领域，参数化 FVS 也是其中一个非常重要的问题，一系列的研究一直在不停的改进其下限，如文献 [4, 8, 9, 14, 16, 17, 19, 24, 27, 32, 46] 等。

在文献 [14] 中作者 Cygan 等利用其发明的 Cut&Count 技术，设计了时间复杂度为 $\mathcal{O}^*(3^k)$ 的固定参数算法来求解 FVC 问题。该算法是一个蒙特卡罗算法（Monte Carlo algorithm），如果不存在大小不超过 k 的 FVS 集合，那么算法一定会返回“NO”，否则有 1/2 概率返回“YES”。据我们所知，该算法是迄今为止 FVS 问题上复杂度最低的固定参数算法。而如果不希望使用随机性算法的话，迄今最快的算法是 Kociumaka 等[32] 在 2014 年提出的基于迭代压缩技术的算法，他们可以在 $\mathcal{O}^*(3.592^k)$ 的时间复杂度内确切的给出问题的答案。

本章中，我们同样使用迭代压缩技术，最后获得一个 $\mathcal{O}^*(3.598^k)$ 时间复杂度的固定参数算法，对比 Kociumaka 的算法，时间复杂度上我们十分接近，然而 Kociumaka 的算法一共使用了 12 条分支规则，而我们的算法只有 1 条更加简洁明了。

关于迭代压缩技术（Iterative Compression Technology），是指将原问题拆分成一系列规模逐渐增长的子问题，并依次对这些子问题进行求解。这样可以保证在求解每个子问题的时候，我们便已经知道上一个与其规模十分接近的子问题的解，利用上个问题的解来降低问题求解当前问题的难度。很多固定参数算法的设计都使用了该技术。

在本文中我们在求解原反馈顶点集问题分解出来的子问题时，利用了不相交反馈顶点集问题（Disjoint Feedback Vertex Set，简称 Disjoint-FVS 问题），这里我们给出它详细的定义，具体使用方法我们留在下一节中。该问题是在原一般图上 FVS 问题增加额

外的约束获得的，要求图 G 可以分解成两个森林，并且所求反馈顶点集必须只取其中一个森林中的顶点，详细定义如下：

不相交反馈顶点集问题

(Disjoint Feedback Vertex Set，简称 Disjoint-FVS)

输入： 图 $G(V,E)$，整数 k 以及顶点集合 V 的划分 $V = U \uplus D$ 满足 $G[U]$ 和 $G[D]$ 都是森林

问题： 图 G 中是否存在一个反馈顶点集合 $X \subseteq D$，其大小不超过 k

4.2 使用迭代压缩技术转化 FVS 问题

与近年来大多数 FVS 问题的参数算法相似，我们基于迭代压缩技术将 FVS 问题转化成另外一个与之高度相关的问题，Disjoint-FVS 问题。

在本节中首先，我们直接给出关于 Disjoint-FVS 问题的时间复杂度[1]，并基于该结论和迭代压缩技术，给出 FVS 问题的时间复杂度。之后，本章的剩余部分将注意力集中在 Disjoint-FVS 问题上，证明下述结论的正确性。

定理 4.1 对于一个 Disjoint-FVS 问题实例 $I = (G, U, D, k)$，可以在 $\mathcal{O}^*(C^k_{k+cc(G[U])/2})$[2]时间复杂度内构成出符合条件的反馈顶点集合或者回答不存在这样的反馈顶点集合。

在定理4.1中, 我们使用了组合计数的方式来表达其复杂度，不过这并不是最终结果。在后续应用中，我们只处理参数 k 与图 $G[U]$ 连通块个数成一定比例的 Disjoint-FVS 问题实例，在这样情况下可以使用 Stirling 公式将其化简成 c^k 形式。

在定理4.1的基础上，我们应用迭代压缩技术构造出下面求解 FVS 问题的算法。

定理 4.2 对于一个 FVS 问题实例 $I = (G, k)$, 可以在 $\mathcal{O}^*(3.598^k)$ 时间复杂度内构成出大小不超过 k 的反馈顶点集合或者回答不存在这样的反馈定点集合。

证明 令 $v_1, v_2, ..., v_n$ 是图 G 顶点集合 V 的任意排列，对于 $1 \leq i \leq n$ 定义顶点集合 $V_i = \{v_1, v_2, ..., v_i\}$ 和图 $G_i = G[V_i]$。我们依次求解 FVS 问题实例 (G_i, k) 对于 $i = k+1, ..., n$。[3] 显然如果对于某个 i，FVS 问题实例 (G_i, k) 无解，那么原问题实例 I 也无解，因为图 G_i 是图 G 的一个导出子图。

[1]定理4.1的证明在下一节

[2]C^m_m 表示从 m 个不同元素中取出 n 个元素的组合数

[3]注意，对于问题实例 (G_k, k) 顶点集合 V_k 显然是符合条件的反馈顶点集，不需要进行求解

如果我们已经获得图 G_i 上大小不超过 k 的反馈顶点集合 X_i，那么集合 $X_i \cup \{v_{i+1}\}$ 则构成了图 G_{i+1} 的一个大小不超过 $k+1$ 反馈顶点集合。令 $Z = X_i \cup \{v_{i+1}\}, D = V_i \setminus X_i = V_{i+1} \setminus Z$。由定义可知 $G_{i+1}[D] = G_i[D]$ 是一个森林。

接下来，我们枚举集合 Z 的所有子集 Y，尝试回答当图 G_{i+1} 的反馈顶点集一定包含所有 Y 中顶点，并且一定不包含 $Z \setminus Y$ 中顶点时，其大小是否能够不超过 k。令 $U = Z \setminus Y$，显然当 $G[U]$ 不是一个森林时，肯定不存在符合要求的反馈顶点集，因此我们后续只讨论 $G[U]$ 是森林的情况。此时根据上述约束，我们可以把当前的问题转换成 Disjoint-FVS 问题实例 $I_Y = (G_{i+1} \setminus Y, U, D, k-|Y|)$。由定理4.1，可以得知求解问题实例 I_Y 的时间复杂度为 $\mathcal{O}^*(C^k_{k+cc(G[U])/2})$，其中 $cc(G[U]) \leq |U| = k-|Y|$，故时间复杂度可以化简成，$\mathcal{O}^*(C^{k-|Y|}_{3(k-|Y|)})$。应用 Stirling 公式，可以得到，$\mathcal{O}^*(C^{k-|Y|}_{3(k-|Y|)}) = \mathcal{O}^*(1.61185^{k-|Y|})$。最后，如果对于某个子集 $Y \subseteq Z$，我们求解出 Disjoint-FVS 问题实例 I_Y 的反馈顶点集 X，则 $X \cup Y$ 是问题实例 (G_{i+1}, k) 的解。反之，如果不存在这样的子集，那么问题实例 (G_{i+1}, k) 不存在解。

考虑整个过程的运行时间，求解问题实例 (G_{i+1}, k) 时间复杂度，如下：

$$\mathcal{O}^*(\sum_{Y \subseteq Z} C^{k-|Y|}_{3(k-|Y|)}) = \mathcal{O}^*(\sum_{Y \subseteq Z} 1.61185^{k-|Y|}) = \mathcal{O}^*((1+1.61185^2)^k) = \mathcal{O}^*(3.598^k)$$

从 G_{k+1} 到 G_n，我们重复上述过程 $n-k$ 次便可以求解原 FVS 问题实例 (G, k)，整个过程时间复杂度也可以表示成 $\mathcal{O}^*(3.598^k)$。

证毕。 ■

4.3 Disjoint-FVS 问题算法

求解 Disjoint-FVS 问题的方法，一般都是分支搜索算法，基本思路是取出顶点集合 D 中的某个顶点 v，通过假设 v 是否存在于反馈顶点集中，我们可以获得实例的两个分支。本章中，我们将给出一个 Disjoint-FVS 问题的算法，以证明定理4.1的正确性。

4.3.1 Disjoint-FVS 问题实例的评估

我们的算法与之前 Cao[8] 和 Kociumaka[32] 的算法相似，与这两个算法相比，我们创新性地使用一个二元组来评估 Disjoint-FVS 问题实例的复杂程度[1]。这在以前的参数算法文献中，也是十分罕见的。本节中，我们将给出具体的评估（measure）方法以及对其边界的证明。

[1]改变之前一直使用单个数值来评估 Disjoint-FVS 问题实例的方式

首先，得益于前人的工作成果[8]，我们知道某些情况下 Disjoint-FVS 问题是存在多项式解法的，对于这部分满足条件的顶点，我们并不需要对进行枚举，而是可以在枚举完所有其他顶点后，直接在多项式时间内求解剩余问题。下面，我们引入其中一个算法，如下：

引理 4.1 ([8])　对于一个 Disjoint-FVS 问题实例 (G, U, D, k)，如果所有的顶点 $v \in D$，满足其邻居数量 $|N(v)| \leq 3$，那么存在一个时间复杂度为 $O(n^2 log^6 n)$ 的多项式算法，或者构造出图 G 的反馈顶点集 X 满足 $|X| \leq k$ 并且 $X \subseteq D$，或者回答不存在这样的反馈顶点集。

定义顶点集合 $T = \{v \in D \mid |N(v)| = 3 \text{ 并且} N(v) \subseteq U\}$。根据引理4.1，我们并不需要对顶点集合 T 中的顶点进行枚举。接下来，给出一个引理对顶点集合 T 的大小进行分析。该引理也来源于 Cao 等的贡献，为了算法的完整性此处我们也给出其简单证明。

引理 4.2 ([8])　对于 Disjoint-FVS 问题实例 $I(G, U, D, k)$，如果 $k + \frac{cc(G[U])}{2} - |T| < 0$ 那么该实例肯定没有符合条件的解。

证明　我们通过证明该命题的逆反命题，如果 Disjoint-FVS 问题实例 $I(G, U, D, k)$ 有解，则 $k + \frac{cc(G[U])}{2} - |T| \geq 0$，来证明引理的正确性。

假设顶点集合 X 是图 G 的一个反馈顶点集合，并且 $X \subseteq D, |X| \leq k$，即 X 是问题实例 I 符合条件的解。则对于任意顶点 $v \in (T \setminus X)$，v 连接了 3 个不同的 $G[U]$ 的连通块，所以 $cc(G[U]) \geq 2|T \setminus X| + 1$。结合 $|X| \leq k$，可以获得，$|T| = |X \cap T| + |T \setminus X| \leq k + \frac{cc(G[U])-1}{2}$。整理得，$k + \frac{cc(G[U])}{2} - |T| \geq \frac{1}{2}$。

证毕。 ■

定义 4.4　对于 Disjoint-FVS 问题实例 $I(G, U, D, k)$，定义其评估函数如下，

$$\mu(I) = (\mu_1(I), \mu_2(I)) = (k, k + \frac{1}{2} cc(G[U]) - |T|)$$

显然，对于该评估函数，当 $\mu_1(I) \leq 0$ 或者 $\mu_2(I) < 0$ 的时候，问题实例多项式可解。当 $\mu_2(I) < 0$ 时，根据引理4.2，直接返回不存在符合条件的解；当 $\mu_1(I) = 0$ 时，那么检查图 G 是否森林就可以立刻返回答案。

4.3.2 收缩规则

本节中，我们给出若干特殊情况下，应用在 Disjoint-FVS 问题实例上的收缩规则，并且证明他们是安全且有效的。假设对于一个收缩规则，将问题实例 I 转化成问题实例

I'，我们说该规则是安全的，代表新问题实例 I' 有解当且仅当原问题实例 I 有解；说该规则是有效的,则意味着其两个评估参数都没有上升,即 $u_1(I') \leq u_1(I), u_2(I') \leq u_2(I)$。

收缩规则 1　移除图 G 中所有度数为 1 的顶点。

收缩规则 2　如果存在一个顶点 $v \in D$，其有两个邻居在图 $G[U]$ 的同一个连通块中，则移除顶点 v 并令 k 减少 1, 即令 $G \leftarrow G \setminus v, k \leftarrow k-1$。

收缩规则 3　如果存在一个顶点 $v \in D$，其度数为 2 且至少有一个邻居在 U 中，则将其从顶点集合 D 中移到顶点集合 U 中，即令 $U \leftarrow U \cup \{v\}, D \leftarrow D \setminus \{v\}$。

收缩规则 4　如果存在一个顶点 $v \in D$，其度数为 2，则其删除并且在它的两个邻居间连一条边，即令 $G \leftarrow G \setminus \{v\}, E(G) \leftarrow E(G) \cup N(v)$。

收缩规则 5　如果存在一个度数为 3 的顶点 $v \in D$, 满足 $|D \cap N(v)| = 1$, 假设其 3 个顶点分别为 $w \in D, u_1 \in U, u_2 \in U$, 删除顶点 v 和 w 之间的边,添加新顶点 z 属于顶点集合 U, 并且连接其与顶点 w 和 v, 即令 $Z \leftarrow Z \cup \{z\}, E(G) \leftarrow E(G) \cup \{e_{zw}, e_{zv}\} \setminus \{e_{wv}\}$。

对于每个问题实例，我们按照给出顺序依次应用这个五个收缩规则，直到没有规则适用。显然，这五条收缩规则都可以在多项式时间内应用，并且由于每次应用都会导致问题实例的规模的收缩，因此执行次数是 $O(n)$ 级别的，故整个过程也是多项式时间的。接下来，我们证明他们都是安全且有效的。

引理 4.3　收缩规则 1-5 都是安全且有效的。

证明　对于收缩规则 1，度数为 1 的顶点不可能存在于任何环中，可以直接忽略。对于收缩规则 2，如果不将顶点 v 取进反馈顶点集中，则剩余图必然存在环。因此规则 1 和 2 都是安全的。

而在规则 3、4 中，我们可以证明假设存在一个满足问题实例 $I(G, U, D, k)$ 的反馈顶点集 X 包含顶点 v，那么一定可以构造另外一个符合条件的反馈顶点集 X' 不包含顶点 v。假设图 $G \setminus (X \setminus \{v\})$ 中存在环（否则可以令 $X' = X \setminus \{v\}$），因为图 $G \setminus X$ 不存在环，所以图 $G \setminus (X \setminus \{v\})$ 中的环肯定经过顶点 v，又因为顶点 v 度数为 2 且其邻居不属于同一个 $G[U]$ 连通块（否则会应用收缩规则 2），所以环仅有一个并且环上一定有顶点不属于 U，我们取其中一个记为 u。构造 $X' = X \setminus \{v\} \cup \{u\}$，显然符合要求。因此规则 3 和 4 都是安全的。

对于这 4 条规则，显然 k 和 $cc(G[U])$ 都不会上升。而集合 T 缩小只可能发生在规则 1 删除一个属于 U 的顶点时，并且此时 $cc(G[U])$ 也会减少 1。因此这 4 条规则都是有效的。

最后讨论收缩规则 5，收缩规则 5 使得新的问题实例增加了一个顶点 $z \in U$，然而并没有增加任何新的环。可以观察到如果新图上的环经过顶点 z，那么这个环在原图上也存在，其经过了边 e_{wv}。因此，在应用规则 5 前后的两个问题实例上求解反馈顶点集实际上是等价的，即规则 5 是安全的。观察收缩规则 5 在创造一个新的图 $cc(G[U])$ 的连通块的同时，令顶点 v 满足 $|N(v)| = 3$ 并且 $N(v) \subseteq U$，因此顶点集合 T 也增加了 1。故，应用完收缩规则 5 有，$u_a(I') = u_a(I) = k, u_b(I') = u_b(I) + 1/2 - 1 = u_b(I) - 1/2$。因此收缩规则 5 也是安全有效的。

证毕。 ■

4.3.3 分支规则

前文已经提到，我们的算法是基于分支搜索的，每次会从顶点集合 D 中取出一个顶点，通过假设它存在于反馈顶点集或者不存在来获得两个搜索分支。为了获得更加优秀运行时间上界，我们希望通过选择最合适的顶点，以提高评估函数 $\mu(I)$ 的下降程度。本节中我们首先给出合适的选择顶点的方法，并且对应用分支规则后的两个问题实例的评估函数变化进行分析。

引理 4.4 *对于 Disjoint-FVS 问题实例 $I(G, U, D, k)$，当收缩规则 1-5 都不能应用且顶点集合 $D \setminus T$ 非空时，至少存在一个顶点 $v \in (D \setminus T)$，其邻居中至少有 3 个顶点属于集合 U。*

证明 首先，收缩规则 1-5 都不能应用可以推导出 D 中所有顶点度数都大于等于 3。其次顶点集合 $D \setminus T$ 的导出子图是森林，取森林的任意孤立节点或者叶子结点 v，顶点 v 满足 $|N(v) \cap D| \leq 1$。再结合顶点 v 不满足收缩规则 5 的应用条件，可以得知顶点 v 至少有 3 个邻居在 U 中。 ■

根据上述引理，我们总能应用以下分支规则。

分支规则

前提： 存在一个顶点 $v \in (D \setminus T)$，其邻居中至少有 3 个顶点在集合 U 内。

分支一： $(G_1 \leftarrow G \setminus \{v\},\ U_1 \leftarrow U,\ D_1 \leftarrow D \setminus \{v\}, k_1 \leftarrow k - 1)$

分支二： $(G_2 \leftarrow G,\ U_2 \leftarrow U \cup \{v\},\ D_2 \leftarrow D \setminus \{v\}, k_2 \leftarrow k)$

对于分支一，我们将顶点 v 取进了反馈顶点集，即可以获得一个不包含顶点 v 的新问题实例 I_1，此时 k_1 减少 1，同时 $cc(G_1[U_1])$ 和 $|T_1|$ 显然没有发生任何变化。因此，

$$u_a(I_1) = k - 1 = u_a(I) - 1$$
$$u_b(I_1) = k - 1 + \frac{1}{2}cc(G[U]) - |T| = u_b(I) \ - 1$$

而对于分支二，当我们决定将顶点 v 排除出反馈顶点集时，可以将其移动到集合 U 中，由于收缩规则 2 不能被应用，顶点 v 的所有邻居处于不同的 $G[U]$ 连通块中，因此 $G[U_2]$ 依然是森林并且 $cc(G[U_2]) \leq cc(G[U]) - 2$。同时此时 k 保持不变，$|T|$ 有可能增加但不可能减少。因此有，

$$u_a(I_2) = u_a(I)$$
$$u_b(I_2) \leq k + \frac{1}{2}(cc(G[U]) - 1) - |T| = u_b(I) - 1$$

4.3.4 复杂度分析

对于 Disjoint-FVS 问题实例，假设函数 $T(\mu(I))$ 表示其搜索树节点个数。回顾上一节的分支规则，其产生两个搜索分支，我们已经分析了其评估函数的变化，因此对于当前问题实例的搜索树规模，有

$$\begin{aligned} T(\mu(I)) &= T(\mu_a(I), \mu_b(I)) \\ &\leq T(\mu_a(I) - 1,\ \mu_b(I) - 1) + T(\mu_a(I),\ \mu_b(I) - 1) \end{aligned}$$

此外，之前也讨论了当 $\mu_a \leq 0$ 或者 $\mu_b \leq 0$ 时，$T(\mu) = 1$。因此，我们可以得知

$$T(\mu(I)) \leq C_{\mu_b}^{\mu_a} \leq C_{k+cc(G[U])/2}^{k}$$

另外在搜索树上应用收缩规则和分支规则的时间复杂度都是多项式的，因此对于任何 Disjoint-FVS 问题实例，可以在 $\mathcal{O}^*(C_{k+cc(G[U])/2}^{k})$ 时间复杂度内构成出符合条件的反馈顶点集合或者回答不存在这样的反馈顶点集合。至此我们成功证明了引理4.1。

4.4 小结

在本节中我们首先给出本章算法的伪代码。

我们将算法分解成两个函数。首先给出 Disjoint-FVS 算法，如4.3节所描述的这是一个分支搜索算法。每次进入递归函数，首先应用收缩规则进行收缩，之后如果程序没

有到达多项式可解并且没有到达不可能存在合法解的情形，我们选取顶点并进行分支搜索。详见 Algorithm 4。

最后是完整的 FVS 问题算法，如前文所述其中我们使用了迭代压缩技术，通过依次加入每个顶点来最终求解完整问题实例。期间，我们枚举完当前顶点覆盖集 Z 的子集后会构造 Disjoint-FVS 问题实例并会调用前面的 Disjoint-FVS 算法。具体伪代码，详见 Algorithm 5。

至此，本章中完整地给出了一个求解经典参数化反馈顶点集问题的确定性固定参数算法，时间复杂度为 $\mathcal{O}^*(3.598^k)$。

Algorithm 4 Disjoint-FVS 算法

Input: Disjoint 问题实例 (G, U, D, k)

Output: 符合条件的问题实例反馈顶点集合或者“NO”

function DISJOINT-FVS(G, U, D, k)
  $G, U, D, k \leftarrow ApplyReduceRlues(G, U, D, k)$
  ▷ 首先应用收缩规则 1-5
  $T \leftarrow \{v \in D \mid |N(v)| = 3 \text{ and } N(v) \subseteq U\}$
  ▷ 构造集合 T
  $u_a \leftarrow k, u_b \leftarrow k + \frac{1}{2}cc(G[U]) - |T|$
  ▷ 计算当前问题实例的评估值
  if $(u_a < 0)$ or $(u_b < 0)$ **then return** ”NO”
  ▷ 搜索终止条件，已经不可能存在解
  if $D = T$ **then**
    在多项式时间内求解当前问题并返回结果 ▷ 引理4.1
  else
    取 D 中至少有 3 个邻居在 U 中的任一顶点 v
    ▷ 引理4.4表明这样的顶点一定存在
    $branchA \leftarrow$ Disjoint-FVS$(G \setminus v, U, D \setminus v, k-1)$
    ▷ 分支 1，此时我们假定顶点 v 在最终的 FVS 集合中
    if branchA 不等于“NO”**then return** $branchA \cup \{v\}$
    ▷ 分支 1 存在解则直接返回，否则调用分支 2
    return Disjoint-FVS$(G, U \cup \{v\}, D \setminus v, k)$
    ▷ 分支 2，此时我们假定顶点 v 不在最终的 FVS 集合中
  end if
end function

Algorithm 5 FVS 算法

Input: FVS 问题实例 (G, k)

Output: 符合条件的问题实例反馈顶点集合或者“NO”

function FVS(G, k)
    $V_i \leftarrow \emptyset, X \leftarrow \emptyset$
    ▷ 初始化
    for 遍历所有顶点 $v \in V$ **do**
        $V_i \leftarrow V_i \cup v, G_i \leftarrow G[V_i], D \leftarrow V_i \setminus X$
        $Z \leftarrow X \cup \{v\}$
        ▷ 依次加入新顶点
        If $|Z| \leq k$ **then** 令 $X \leftarrow Z$ 并且 **continue**
        ▷ 已经满足条件跳过该子问题的求解
        for 遍历所有 Z 的子集 $Y \subseteq Z$ **do**
            ▷ 枚举 Z 的子集以构造新 Disjoint-FVS 问题
            If $G[Z \setminus Y]$ 不是森林 **then continue**
            ▷ 无法构造 Disjoint-FVS 问题，此时不存在解跳过该子集 Y
            $result \leftarrow$ Disjoint-FVS$(G_i \setminus Y, Z \setminus Y, D, k - |Y|)$
            ▷ 调用 Algorithm 4 求解 Disjoint-FVS 问题
            If $result$ 不是“NO”**then** 令 $X \leftarrow Y \cup result$ 并且 **break**
            ▷ 使用 Disjoint-FVS 问题的解构造当前子问题的解
        end for
        If 对于所有 Z 的子集 $result$ 均是“NO”**then return** “NO”
        ▷ 当该子问题不存在合法解时，原 FVS 问题实例也不可能存在
    end for
    return X
    ▷ 加入所有顶点后，X 即原问题的解
end function

第 五 章 总结与展望

本章总结全文的工作，并对展望未来的研究前景和进一步努力的方向。

5.1 工作总结

本文主要从参数计算与复杂性理论的角度研究 NP 难问题。具体的研究对象有基于线性松弛下界的参数化顶点覆盖问题和参数化反馈顶点集问题。顶点覆盖和反馈顶点集都是非常经典的组合优化领域的问题，也是图论中最重要的基础性问题之一。他们在生物科学、电路设计、工程选址等现实领域都有着许多应用，因此这两个问题的固定参数算法设计具有重大理论意义和实际价值。

在基于线性松弛下界的参数化顶点覆盖问题（VCAL）上，我们参考了之前 Iwata 等的研究成果[26]，首先将线性松弛下的最小顶点覆盖问题（LPVC）转化成最大网络流问题。之后，仔细讨论了该网络流模型跟原问题之间的联系，并设计了利用网络流模型来对原问题实例进行收缩的核心化算法。注意借助网络流模型，我们成功把这一阶段的时间复杂度控制在 $O(|V|+|E|)$ 内。最后，在核心化之后的问题实例上，我们采用分支搜索技术进行求解。通过对当前图结构的仔细讨论，我们设计了 3 条搜索分支规则，使得搜索树的规模被限定在 $O(2.619^{\mu})$ 内。另外，在这个阶段中我们仔细确保了所有规则的执行时间以及执行规则后维护新的最大流的时间均是线性的。结合以上所有的工作，我们成功获得了一个时间复杂度为 $O(2.618^{\mu}(|V|+|E|))$ 的线性固定参数算法，对比之前该问题上最好的线性固定参数算法 $O(4^{\mu}(|V|+|E|))$，有十分明显的提升。

在参数化反馈顶点集问题上，我们在 CaoYixin 等[8] 等的算法基础上进行了进一步研究。主要思路是，先使用迭代压缩技术将 FVS 问题转化成该问题的一个变种，Disjoint-FVS 问题，这是当前求解参数化反馈顶点集问题的一个标准框架。之后在 Disjoint-FVS 问题的求解上，我们创新性地同时使用两个参数对搜索树的规模进行约束，得到了一个与其他相关文献都不一样的结论。最终，获得了一个时间复杂度为 $\mathcal{O}^*(3.598^k)$ 的 FPT 算法，大大地改进了原来 CaoYixin 等 $\mathcal{O}^*(3.83^k)$ 的结论。如果横向对比其他同样改进原来 CaoYixin 等算法的其他研究成果，当前最好的是 Kociumaka 等[31] 的成果，他们的时间复杂度是 $\mathcal{O}^*(3.592^k)$，与我们优化的思路不一致但是结果十分接近。

5.2 可改进点及未来研究展望

关于本文所研究的两个问题，依然有很多有待改进的地方，有待于我们的进一步研究，主要有以下几个方面：

（1）参数化顶点覆盖问题更低的时间复杂度

在基于线性松弛下界的参数化顶点覆盖问题（VCAL）上，如果只考虑 $f(\mu)$ 函数不考虑多项式部分，当前已经有研究结果可以做到低于 2.619^{μ} 了[34]，对于这样的算法能不能也对其多项式部分进行改进，将复杂度降到线性，以提高算法的可用性？基于以上考虑，后续研究中可以尝试设计复杂度为 $O(2.3146^{\mu}(|V|+|E|))$ 或者更优秀的算法。

（2）带权图上的参数化顶点覆盖问题

本文中，基于线性松弛下界的参数化顶点覆盖问题（VCAL）的讨论都是集中在不带权的普通图上，我们的算法并不能直接移植到带权图上，这个是我们研究工作里面比较遗憾的缺陷。后续工作中可以对带权图上该问题进行进一步研究。

（3）参数化反馈顶点集问题的进一步研究思路

在该问题的确定性 FPT 算法地设计中，我们和 Kociumaka 等分别从不同的思路对之前的算法进行了研究，分别获得了十分接近的新结论。而这两个思路是否能结合起来，相辅相成获得新的理论下界？这也是后续工作中一个重要的方向。另外，在随机 FPT 算法里该问题的复杂度到达了 $\mathcal{O}^*(3^k)$，我们相信肯定也有确定性的 FPT 算法能到达这个复杂度，但还有待后续的研究。

参考文献

[1] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[2] R Balasubramanian, Michael R Fellows, and Venkatesh Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, 1998.

[3] Reuven Bar-Yehuda, Ann Becker, and Dan Geiger. Randomized algorithms for the loop cutset problem. *Journal Of Artificial Intelligence Research*, 2000.

[4] Hans L Bodlaender. On disjoint cycles. *International Journal of Foundations of Computer Science*, 5(01):59–68, 1994.

[5] Hans L Bodlaender. A cubic kernel for feedback vertex set. In *STACS 2007*, pages 320–331. Springer, 2007.

[6] Kevin Burrage, Vladimir Estivill-Castro, Michael Fellows, Michael Langston, Shev Mac, and Frances Rosamond. The undirected feedback vertex set problem has a poly (k) kernel. In *Parameterized and exact computation*, pages 192–202. Springer, 2006.

[7] Jonathan F Buss and Judy Goldsmith. Nondeterminism within p^*. *SIAM Journal on Computing*, 22(3):560–572, 1993.

[8] Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set new measure and new structures. In *Algorithm Theory-SWAT 2010*, pages 93–104. Springer, 2010.

[9] Jianer Chen, Fedor V Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008.

[10] Jianer Chen, Iyad A Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

[11] Jianer Chen, Iyad A Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40):3736–3756, 2010.

[12] Jianer Chen, Lihua Liu, and Weijia Jia. Improvement on vertex cover for low-degree graphs. *Networks*, 35(4):253–259, 2000.

[13] Miroslav Chlebík and Janka Chlebíková. Crown reductions for the minimum weighted vertex cover problem. *Discrete Applied Mathematics*, 156(3):292–312, 2008.

[14] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Joham MM van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 150–159. IEEE, 2011.

[15] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. In *Parameterized and Exact Computation*, pages 1–12. Springer, 2011.

[16] Frank Dehne, Michael Fellows, Michael Langston, Frances Rosamond, and Kim Stevens. An o (2o (k) n3) fpt algorithm for the undirected feedback vertex set problem. *Theory of Computing Systems*, 41(3):479–492, 2007.

[17] Rod G Downey and Michael R Fellows. *Fixed-parameter tractability and completeness*. Cornell University, Mathematical Sciences Institute, 1992.

[18] Rod G Downey, Michael R Fellows, Rolf Niedermeier, Peter Rossmanith, et al. Parameterized complexity. 1998.

[19] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

[20] LR Ford and Delbert Ray Fulkerson. *Flows in networks*, volume 1962. Princeton Princeton University Press, 1962.

[21] Michael R Garey and David S Johnson. A guide to the theory of np-completeness. *WH Freemann, New York*, 1979.

[22] Shivam Garg and Geevarghese Philip. Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee. *arXiv preprint arXiv:1509.03990*, 2015.

[23] Gaston H. Gonnet, Michael T. Hallett, Chantal Korostensky, and Laurent Bernardin. Darwin v. 2.0: an interpreted computer language for the biosciences. *Bioinformatics*, 16(2):101–103, 2000.

[24] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.

[25] Dorit S Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

[26] Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time fpt algorithms via network flow. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1749–1761. SIAM, 2014.

[27] Iyad Kanj, Michael Pelsmajer, and Marcus Schaefer. Parameterized algorithms for feedback vertex set. In *Parameterized and Exact Computation*, pages 235–247. Springer, 2004.

[28] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[29] LG Khachiian. Polynomial algorithm in linear programming. In *Akademiia Nauk SSSR, Doklady*, volume 244, pages 1093–1096, 1979.

[30] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[31] T. Kociumaka and M. Pilipczuk. Faster deterministic Feedback Vertex Set. *ArXiv e-prints*, June 2013.

[32] Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014.

[33] Michael Lampis. A kernel of order for vertex cover. *Information Processing Letters*, 111(23):1089–1091, 2011.

[34] Daniel Lokshtanov, NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms (TALG)*, 11(2):15, 2014.

[35] Zbigniew Michalewicz and David B Fogel. *How to solve it: modern heuristics*. Springer Science & Business Media, 2013.

[36] Sounaka Mishra, Venkatesh Raman, Saket Saurabh, Somnath Sikdar, and CR Subramanian. The complexity of könig subgraph problems and above-guarantee vertex cover. *Algorithmica*, 61(4):857–881, 2011.

[37] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[38] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[39] NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Lp can be a cure for parameterized problems. In *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 338–349. LIPIcs, 2012.

[40] George L Nemhauser and Leslie E Trotter Jr. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[41] Rolf Niedermeier and Peter Rossmanith. Upper bounds for vertex cover further improved. In *STACS 99*, pages 561–570. Springer, 1999.

[42] Rolf Niedermeier and Peter Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47(2):63–77, 2003.

[43] Christos H Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 12–19. ACM, 1997.

[44] Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Paths, flowers and vertex cover. In *Algorithms–ESA 2011*, pages 382–393. Springer, 2011.

[45] Venkatesh Raman, Saket Saurabh, and CR Subramanian. Faster fixed parameter tractable algorithms for undirected feedback vertex set. In *Algorithms and computation*, pages 241–248. Springer, 2002.

[46] Venkatesh Raman, Saket Saurabh, and CR Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms (TALG)*, 2(3):403–415, 2006.

[47] Igor Razgon and Barry O'Sullivan. Almost 2-sat is fixed-parameter tractable. *Journal of Computer and System Sciences*, 75(8):435–450, 2009.

[48] Micha Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.

[49] Arezou Soleimanfallah and Anders Yeo. A kernel of order 2k- c for vertex cover. *Discrete Mathematics*, 311(10):892–895, 2011.

[50] Ulrike Stege and Michael Fellows. *An Improved Fixed Parameter Tractable Algorithm for Vertex Cover*. Citeseer, 1999.

[51] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[52] Stéphan Thomassé and Stéphan Thomassé. A quadratic kernel for feedback vertex set. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 115–119. Society for Industrial and Applied Mathematics, 2009.

致 谢

光阴飞逝，我在中山大学的第七年就这样逝去了。这七年的学习和生活，让我从一个懵懂的高中毕业生成长到今天即将踏入社会工作的硕士。这其中对母校的感恩，无以言表。衷心感谢母校给我带来诸多优秀的师长，在学习上赋予我知识，也衷心感谢母校让我认识了许多好友，在生活上予我以帮助关心。可能这七年会是我生命中最重要的一段光阴！

在此时论文完成之际，我最想感谢的人是林瀚老师。从大一在辩论队认识了林瀚师兄，到进入 acm 队遇到林瀚老师作为指导老师，再到研究生阶段选择林瀚老师作为我的导师。这期间林老师一直在我整个大学、研究生生涯中扮演着一个亦师亦兄的角色。在各个方面，他给我的帮助实在太多了，难以一一列举，只能在此再说一声，谢谢！

另外一个需要感谢的人是我女朋友。在中大 7 年，与陈结贞就在一起了 6 年多。我要感谢她这期间一直在背后给予我支持，支撑我完成自己的学业。

这里我还要感谢我的家人，爸爸、妈妈、爷爷、奶奶还有姐姐，感谢他们养育了我这么多年，一直对我的学习予以无条件的支持。

最后我要感谢那些在这里不能一一提到的师长、同学和好友，感谢所有帮助和关心过我的人们。谢谢!

此刻真希望自己能有飞扬的文采将内心的感激之情更加写实的记录下来。

林泽群

2016 年 5 月于中山大学