哈尔滨工业大学计算机科学与技术学院

《算法设计与分析》

# 课程报告

| | |
|---|---|
| 学号 | 2021111000 |
| 姓名 | 李卓凌 |
| 班级 | 2103201 |
| 专业 | 信息安全 |
| 授课教师 | 张开旗 |
| 报告日期 | 2023 年 12 月 28 日 |

# Vertex Deletion into Bipartite Permutation Graphs

Author: Łukasz Bo˙zyk1 · Jan Derbisz2 · Tomasz Krawczyk2 · Jana Novotná1,3 ·Karolina Okrasa1,4

Received: Algorithmica (2022) 84:2271–2291
https://doi.org/10.1007/s00453-021-00923-7

## 一. 论文阅读

第一部分是我阅读下整个论文搞懂了作者在什么背景下面临什么问题得出了什么结论，不涉及对证明的理解和算法的实现。

### 1.1 摘要

排列图可以定义为线段的交集图，其中线段的端点位于两条平行线 1 和 2 上，每条线上各有一个端点。二分排列图是一个二分图，同时也是排列图。在本文中，我们研究了二分排列顶点删除问题的参数化复杂度，该问题询问对于给定的 n 个顶点的图，我们是否可以移除最多 k 个顶点以获得一个二分排列图。根据 Lewis 和 Yannakakis 的经典结果[20]，这个问题属于 NP 完全问题。我们分析了所谓的几乎二分排列图的结构，这些图可能包含孔（大的诱导圈），而与二分排列图不同。我们利用了这种图中最短孔的结构特性，并用它来获得一个时间复杂度为 O(9ˆk • nˆ9)的二分排列顶点删除算法，并且提供了一个 9-近似多项式时间算法。

关键词：排列图 • 可比图 • 偏序集 • 图修改问题

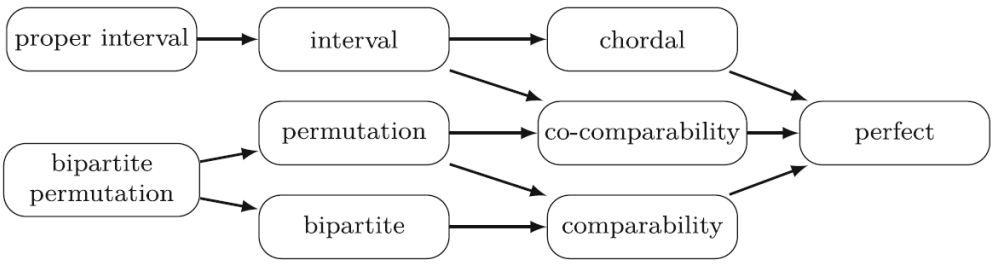
### 1.2 背景介绍及预备知识

#### 1.2.1 问题引出

许多标准的计算问题，包括最大团、最大独立集或最小染色等，在一般情况下是 NP 难的，在图的受限类中却有多项式时间的精确或近似算法。由于其在实

际和理论应用中的重要性，这类图的某些类别被特别密切地研究。其中包括：包含真实数线上区间的相交图（interval graphs）；真实数区间的相交图（proper interval graphs）；树上子树的相交图（chordal graphs）；连续和线性函数定义的区间上的相交图（function and permutation graphs）；顶点对在某个固定部分序上可比较的图（comparability graphs）；可比较图的补图（co-comparability graphs）。

众所周知，函数图的类别相当于共可比性图的类别[13]，排列图的类别相当于可比图和共可比图的交[24]（见图 1 的包含层次）。所有这些图的类别都是遗传的，这意味着它们对顶点删除是封闭的。




图一：图类之间包含的层次结构。图中的箭头 A 类到 B 类图表示 A ⊂ B

在算法设计中，遗传性是一个非常有用的性质，因为每一个遗传图类别也可以被唯一地用最小的禁止诱导子图来表征：如果一个图属于类别 G，当且仅当它不包含某个家族 F 中的图作为诱导子图。对于在上述图类别中引入的每一个图类别，已知了一个禁止子图的表征，详见[8]完美图，[19]区间图，[12]可比图和排列图。然而，对于它们所有，禁止子图的家族是无限的，也可能相当复杂。而且，任何上述的图 G 都是完美的。Grötschel, Lovász, 和 Schrijver[14] 显示在完美图类别中，最大团、最大独立集和最小染色问题都可以在多项式时间内解决。

为上述图类别设计的多项式时间算法有时可以调整到对与这些类别中的图

“接近”的图的工作。通常，图 G 到图类别 G 之间的“接近程度”，是通过将 G 转换为类别 G 中的图所需的操作次数来衡量的，其中一个单一操作可以是从 G 中删除一个顶点，或者添加或删除一条边。这种方法引出了以下通用的问题。

根据所允许的修改种类，我们得到了此问题的四个变种：顶点删除问题、边删除问题、边补全问题以及边编辑问题（后者允许边的删除和添加）。

### 1.2.2 前置知识

本文中考虑的所有图都是简单图，即无向图，没有环和平行边。设 G = (V, E) 为一图。对于 V 的子集 S ⊆ V，由 S 所诱导的 G 的子图是指图 G[S] = (S, {uv | uv ∈ E, u, v ∈ S})。对于 u ∈ V，u 的邻居是指集合 N(u) = {v ∈ V | uv ∈ E}。类似地，对于 V 的子集 U，记 N(U) = u∈U N(u) \ U。设 u, v ∈ V。我们说 u 和 v 在图 G 中的距离为 k，若 k 是 u 和 v 之间最短路径的长度。我们用 Kn 和 Cn 分别表示 n 个顶点的完全图和环。这里所说的 "hole" 是指至少有五个顶点的诱导圈。如果一个 "hole" 包含偶数（奇数）个顶点，则称其为偶圈（奇圈）。对于一图 G = (V, E)，如果 < 是 V 上的传递且反自反的关系，并且满足对每对 u, v ∈ V，当且仅当 uv ∈ E 时 u < v 或者 v < u，那么二元组 (V,<) 是图 G 的传递有向性。一个偏序集（简称偏序或 poset）是指由集合 X 和关于 X 的自反、传递和反对称关系 ≤P 组成的二元组 P = (X, ≤P)。对于偏序 (X, ≤P)，称严格偏序 <P 是 X 上的二元关系，当且仅当 x ≤P y 且 x ≠ y 时 x <P y。换言之，当 <P 是非自反的且传递时，二元组 (X, <P) 是严格偏序。如果偏序 P 中的两个元素 x, y ∈ X 可比较，则 x, y 在 P 中是可比的；否则，x, y 在 P 中是不可比的。一个线性排序 L = (X, ≤L) 是指每两个顶点 x, y ∈ X 都是可比较的偏序。严格线性排序 (X, <L) 是指 x <L y 当且仅当 x ≤L y 且 x = y。设 P = (X, ≤P) 是一偏序集。若 ≤P ⊆ ≤L,

则称 L = (X, ≤L) 是偏序 P 的线性扩张。给定偏序族 P = {Pi = (X, ≤Pi): i ∈ I}，我们称 P 是 Pi 的交集，如果对于每对 x, y ∈ X，当且仅当对每个 i ∈ I 有 x ≤Pi y 时 x ≤P y。偏序 P 的维数是偏序 P 的线性扩张的最小个数。特别地，如果 P 是两维的，则称 P 是二维的。

一个偏序 (X, ≤P) 的可比较图（不可比较图）以 X 为其顶点集，以 P 中可比较（不可比较）的每一对顶点为其边集。值得注意的是：如果 (X, ≤P) 是一个偏序，那么 (X, <P) 是偏序的可比较图的传递有向图。若图 G = (V, E) 是一个偏序 P 的可比较图（非可比较图），那么 G 是 V 上某一偏序的可比较（不可比较）图。因此，当且仅当 G 具有传递有向性时，G 是一个可比较图。当且仅当 G 和其补图是可比较图时，G 是一个排列图 [24] （或等价地，G 和其补图具有传递有向性）。

Baker、Fishburn 和 Roberts [1] 证明了当且仅当 G 是一个二维偏序的不可比较图时 G 是一个排列图。

若两个集合 X 和 Y 是可比较的，则表示 X 和 Y 相对于 ⊆ 关系是可比较的（即，X ⊆ Y 或 Y ⊆ X 成立）。我们使用方便的（尽管非标准的）符号 [m] := {0, 1, . . . , m}，对于每个 m ∈ N。对于每个 i, j ∈ Z，满足 i ≤ j，则 [i, j] 意味着集合 {i, i + 1, . . . , j}。

## 1.3 提出的问题

**输入**：一个图 G（通常不属于 G）和一个数字 k

**问题**：是否可以通过进行≤ k 次适当种类的修改，将 G 转化为类别 G 的图？

对于上文所定义的图类别，所有四种修改问题的变体都是 NP 难的-关于 NP 难性证明，请参见[22]。特别地，Lewis 和 Yannakakis [20]证明了任何非平凡继承图类的顶点删除问题都是 NP 难的。这并不奇怪，因为许多经典的难题可以

被表述为特定类别图的顶点删除问题，例如，顶点覆盖问题可被表述为顶点删除成无边图，反馈顶点集问题可被表述为顶点删除成森林，奇环横切问题可被表述为顶点删除成二分图。图修改问题是研究参数化 NP 完全问题的热门方向。一般而言，对于一个问题Π，一个参数化问题的输入包括问题Π的一个实例 I 和一个参数 k ∈ N。然后我们称Π是固定参数可解（FPT），如果存在一个算法决定是否 1），其中 f 是一个可计算的函数。对于图修改问题，我们通常选择参数 k 作为允许的修改次数，因此此类问题的实例仍为一个二元组（G，k）。

结果表明，通过禁止结构进行特征化有时可以用于设计图修改问题的 FPT 算法。例如，Cai [4]提出了一种针对由禁止诱导子图 F 特征化的图类的修改问题的 FPT 算法。他的算法识别输入图中的禁止结构（当 F 是有限的时，这可以在多项式时间内完成），并对修改该结构的所有可能方法进行分支。由于上文引入的图类的禁止结构族是无限的，因此这些类的修改算法必须更加复杂。对于其中的一些类，修改问题有令人满意的解决方案：

-弦图：修改问题的所有四个版本均为 FPT[7,23]；

-区间图：边补全和边删除是 FPT[5,28]，顶点删除是 FPT[7]，边编辑问题仍未解决；

-正区间图：修改问题的所有四个版本均为 FPT[6]。

另一方面，已知顶点删除问题对完美图是 W[2]-难的[15]。值得一提的是，长期以来，人们一直不知道是否有类别的图能够在多项式时间内识别并且对其进行修改是困难的。Lokshtanov [21]给出了第一个这样的例子，他证明了，对于避免所有轮（即每个点都与其他顶点相邻的环）的图，顶点删除是 W[2]-难的。目前尚不清楚，可比图、补可比图和置换图是否有 FPT 的修改算法。置换图的类别，构成了区间图的超类，也是完美图的一个重要子类，从参数化的角度看，这个问题似乎特别有趣。

我们的关注点。与区间图的类别一样，置换图的类别对于一般而言是 NP 难的丰富问题存在着多项式时间的算法。除了已经提到的对于完美图而言，也存在解决如寻找哈密顿圈、反馈顶点集或支配集问题的多项式算法[3,9]。

鉴于上述种种考虑，由于置换图的类别的所有修改问题-以及相关的可比和补可比图的类别-仍未解决，我们将注意力限制在了二分置换图的类别上，这似乎是一个自然的研究方向。二分置换图本身构成一类有趣的图类，最初由 Spinrad、Brandstädt 和 Stewart [25]研究，他们通过适当选择的其二分顶点类的线性排序对其进行了特征化。关于二分置换图最有趣的结果之一是由 Heggernes 等人 [16]给出的，他们证明了计算图的切宽的 NP 完全问题（即找到图的顶点线性排序，使得任意两个相邻的顶点之间的交点数最小）对于二分置换图是多项式的。

我们的算法利用了二分置换图中一些禁止结构的缺失。鉴于这些结构不能在置换图中出现，我们相信，除了本身是一项完整的成果之外，我们的研究也是向着理解置换图的类别修改问题的参数化复杂性迈出的一步。

## 1.4 得出的定理

**定理 1**：对于顶点删除生成二部排列图问题的实例（G，k），存在一个时间复杂度为 O(9^k • |V(G)|^9)的算法。我们的算法基于对二部排列图的特征描述，首先通过分枝法消除了 size 为常数的禁止子图。在这方面，我们的主要贡献在于对含有孔(由超过十个顶点构成的大的诱导环)的几乎二部排列图的结构分析。这种方法在一定程度上受到了 van't Hof 与 Villanger 的启发，他们在处理 proper interval vertex 删除问题时使用了类似的工具。我们使用了 Spinrad、Brandstädt 和 Stewart 的结果，他们证明了每一个连通的二部排列图 G=(U，W，E)的顶点可以嵌入到一个带形结构中。一旦获得这样的结构，我们证明了每一个破坏所有孔的最小顶点切割都位于孔的几个相邻顶点附近。这使我们能够检查我

们可以找到最小切割的所有可能性。最后，我们使用了一个用于寻找最大流（从而得到最小切割）的多项式算法。

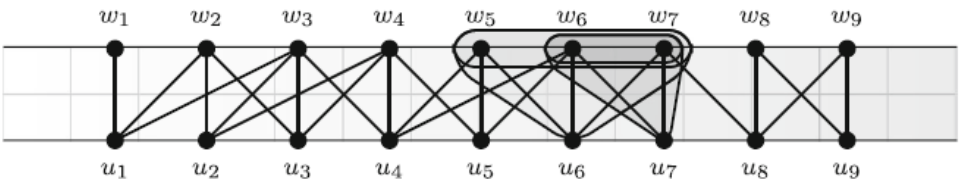



图 2 将二分置换图（U，W，E）嵌入满足邻接和外壳特性

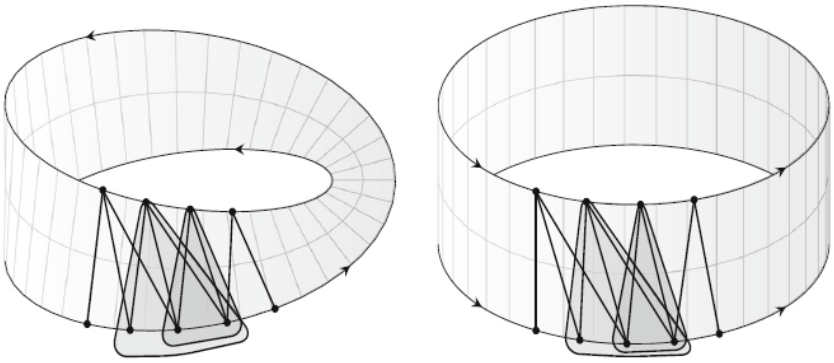


图 3 将连接的几乎二分置换图嵌入到圆柱体或莫比乌斯带中，局部满足邻接和封闭属性

用于证明定理 1 的方法可以略微修改以获得二部排列图顶点删除问题的 9 近似算法。我们证明了以下定理：

**定理 2**：存在一个多项式时间的 9-近似算法，适用于顶点删除生成二部排列图问题。

## 二．算法描述与分析

在证明定理前，作者先证明了很多个命题和引理，证明过程就不放在该论文中了。着重介绍定理的证明。

## 2.1 命题及引理的证明

**定义 1** 图 G = (V，E)是一个几乎二分置换图，如果 G 不包含 T2，X2，X3，K3，Ck（其中 k ∈ [5, 9]）作为诱导子图。假设 G =（V，E）是一个连通的几乎二分置换图。

**命题 1** G 中的每个洞都是一个支配集。

**命题 2** 对于 V 中的每个顶点 v，要么：（1）N（v）∩C = {ci}对于某个 i ∈ [m-1]，或者（2）N（v）∩C = {ci，ci+2}对于某个 i ∈ [m-1]。

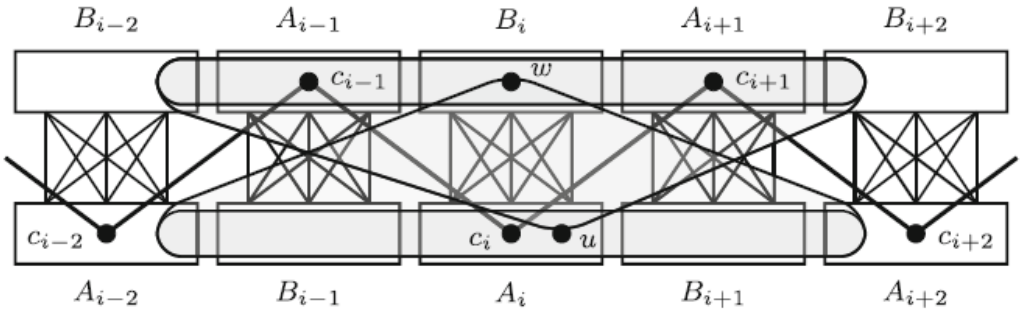



图 4 Ai 中 u 和 Bi 中 w 的可能邻域

**命题 3** 令 i ∈ [m -1]. 则：

(1) Ai 和 Bi 是独立集。

(2) 对于每个 u ∈ Ai 和每个 w ∈ Bi，我们有 uw ∈ E。

(3) 对于每个 u ∈ Ai，我们有 Bi ⊆ N(u) ⊆ B[i-2, i+2]。

(4) 对于每个 w ∈ Bi，我们有 Ai ⊆ N(w) ⊆ A[i-2, i+2]。

**命题 4** 对于每个 i∈[m-1],对于(i±2, i±1)∈{(i-2, i-1), (i+2, i+1)}，有以下内容：

对于每个 w，w ∈ Bi±2∪Ai±1，集合 N(w)∩Ai 和 N(w)∩Ai 是可比较的。此外，如果 w ∈ Bi±2 且 w ∈ Ai±1，则 N(w)鉴于命题 2 的给定，对于每个 i ∈ [m - 1] 我们可以设定 Part-3 章节如下所示:

对于每个 u，若 u ∈ A i±2 ∪ B i±1, N (u) ∩ Bi 和 N (u) ∩ Bi 是可比较的。此外，如果 u ∈ A i±2 而 u ∈ B i±1，则 N (u) ∩ Bi ⊆ N (u) ∩ Bi。

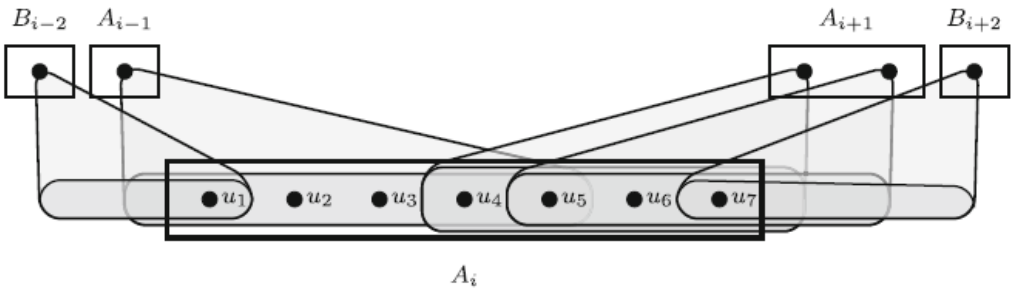



图 5 Bi-2 ∪ Ai-1 ∪ Ai+1 ∪ Bi+2 的顶点邻域仅限于 Ai。我们有

u1<Ai {u2, u3} <Ai u4 <Ai u5 <Ai u6 <Ai u7

**命题 5** 对于每一个 i ∈ [m-1] 都成立：

（1）(A_i, <_{A_i}) 是一个严格的偏序。而且，在(A_i,<_{A_i})中，u, u ∈ A_i 是不可比较的，当且仅当 N(u) = N(u')。

（2）(B_i, <_{B_i}) 是一个严格的偏序。而且，在(B_i, <_{B_i})中，w, w' ∈ B_i 是不可比较的，当且仅当 N(w) = N(w')。

**引理 1** 令 i , j 满足 i ≤ j , | j-i| = m - 3. 令 U = A[i , j ] 且 W = B[i , j ]。那么 G[U ∪ W] 是一个二分置换图。此外，(U,<U) 和 (W,<W) 满足邻接关系的严格线性阶和 G[U ∪ W] 中的包围属性

**命题 6** 假设 C 是 G 中大小为 t（t ≥ m）的一个孔。那么，C 的连续顶点可以按照 c 0 ，c 1 ，...，c t-1 标记，以满足以下条件（取模 k）： -ci ci+1 ∈ E 对于每个 i ∈ [t-1]， -ci < cl ci+2 对于每个 i ∈ [t-1]， -{c ∈ C：ci < cl c < cl ci+2}对于每个 i ∈ [t-1]不为空。

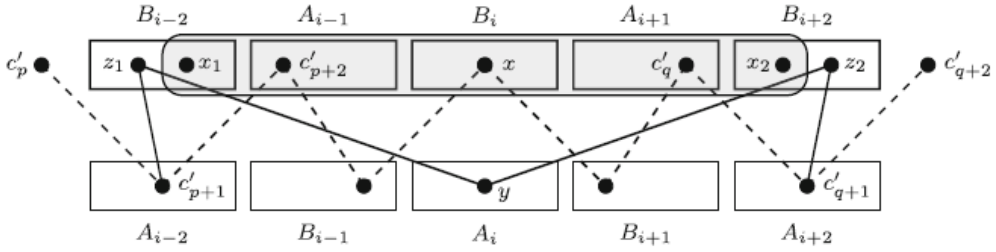



图 6 证明示意图：循环 C 用虚线标记。集合 X 有阴影

命题 7 G 中的每一个最小孔洞割集 X 都完全包含于某个 i∈[m-1]的 V[i-2，i+2]之中。

## 2.2 定理的证明

### 2.2.1 定理一的证明

这里利用前面的结构性结果提供对定理 1 的完整证明。让我们从展示在几乎二分排列图上，可以在多项式时间内解决“二分排列点删除问题”。证明：如果 G 是一个二分排列图，（G，k）是肯定实例，因此，我们在这种情况下完成了。如果 G 不是连通的，我们可以独立考虑每个连通分量，最后，我们将 k 与所有分量上删除的顶点的总数进行比较。设 G 是 G 的一个包含 r 个顶点的连通分量，使得 G 不是二分排列图（否则，显然不需要删除任何顶点）。设 C = {c0，...，cm-1}是 G 中的最短洞（它存在，因为 G 不是二分排列图）。我们可以通过以下方式在时间 O(r6)内找到它。我们遍历所有可能的 V(G)的四元素子集 S = {v1，v2，v3，v4}。对于这些 S，使得 G[S]是 induced P4,，具有连续顶点 v1，v2，v3，v4，我们构造一个图 G，通过删除 N(v2)∪N(v3)\{v1,v4}中的顶点（注意 v2 和 v3

也被删除)。然后在 G 中找到一个最短 v1-v4 路径，时间复杂度为 O(r2)。

根据命题 7，G 中每个极小洞切割 X 都包含在集合 V = VG[i-2, i+2]中，对于某个 i∈[m-1]。因此,我们可以检查包含极小切割的所有可能性。对于每个 i，我们运行以下有向图 Hi 中找最大流的算法。

有向图 Hi 的顶点集为 V×{in, out}∪{s, t}，边集包括：

-所有形式为(u, out)(v, in)的边，其中 uv 是 G[V]的边，

-s(v, in)，如果存在 u∈VG[i-4, i-3]，使得 uv 是 G 的边，

-(u, out)t，如果存在 v∈VG[i+3, i+4]，使得 uv 是 G 的边，

-(u, in)(u, out)，对于所有 u∈V。

将边(u, in)(u, out)的容量设为 1，将所有其他边的容量设为∞（实际上是|VG|)。很容易看出，所定义网络 Hi 中的最小(s, t)-割对应于 G[V]中的最小洞割（单位容量的边(u, in)(u, out)自然对应于 G 的顶点 u)。因此，仅需对每个 Hi 计算经典的最大流算法，并记住极小(s, t)-割的最小大小 kG。这可以在时间 O(m[11])内完成。最后，(G，k）是肯定实例当且仅当所有考虑的连通分量 G 上记住的 kG 的大小之和最多为 k。显然，总运行时间为 O(n6)。

$$\bullet\ (|V\ |\ +\ 2)\ \bullet\ (|E\ G\ [V\ ]\ |\ +\ 2|V\ |)\ 2\ )\ =\ O(r\ 6\ )$$

我们现在提出算法。给定一个 n 顶点图 G = (V, E) 和数字 k，我们想要回答 Bipartite Permutation Vertex Deletion 问题。我们称 (G, k) 为初始实例。我们将我们的算法分为两个部分。第一部分包括一个分支算法，用于将图删除到几乎二部排列图。第一部分的输出是一组实例 (G', k')，其中 G' 是一个几乎二部排列图，0 ≤ k' ≤ k （或者如果不存在这样的实例，则为无解），使得初始实例 (G, k) 是一个是实例，当且仅当至少一个这些实例是一个是实例。

在第二部分中，算法对第一阶段输出的每个实例 (G', k') 运行一个 O(n 6) 时间复杂度的 Bipartite Permutation Vertex Deletion 算法。

我们说 X ⊆ V 是一个禁用集合，如果 G[X] 同构于以下图之一：K3, T2, X2, X3, C5, C6, C7, C8, C9。我们定义了以下规则。

规则：给定实例 (G, k)，k ≥ 1，和一个最小的禁用集合 X，对于 X 中的每个 v ，分支成 |X| 个实例，(G - v, k - 1)。

从初始实例开始，算法会穷尽地应用规则。换句话说，算法形成一个分支树，叶子对应于实例 (G', k')，其中 k' = 0 或者 G' 是一个几乎二部排列图。很明显，由于至少必须从 G 的每个禁用集合移除一个顶点，初始实例是一个是实例当且仅当至少一个叶子是一个是实例。

只有 G 是一个几乎二部排列图（否则，该叶子是一个否实例）的叶子（G', k'），算法才继续第二部分。它运行描述在引理 2 中的算法，以找到是否可以通过最多 k 个顶点删除将 G 转化为二部排列图。它要么找到了是实例，要么在检查所有实例后得出结论，即初始实例是一个否实例。

我们注意到这种分支成有限个小实例的方法是标准技术，更多细节见[27]。我们现在分析整个算法的运行时间。首先，观察到分支树的深度最多为 k，并且具有最多 9k 个叶子节点，因为每当算法分支时 k 都会减少一，而所列出的禁止子图中每个均至多有九个顶点。因此，分支树中的总节点数为 O(9^k)。此外，在每个节点(G, k)中算法的时间复杂度为 O(n^9)，因为它检查 G 是否包含至多 9 个顶点的禁止集。因此，第一部分的时间复杂度为 O(9^k • n^9)。在第二部分，根据引理 2，算法在至多 123 • 9^k 个叶子节点中每个都需要 O(n^6)的工作。因此，第二部分的时间复杂度为 O(9^k • n^6)。由此，我们得出 Bipartite Permutation Vertex Deletion 算法的总运行时间为 O(9^k • n^9)。

### 2.2.2 定理二的证明

设 G = (V，E)是一个图，Y ⊆ V 是 G 的一个顶点子集，使得 G-Y 是一个二分排列图。我们希望在多项式时间内构造一个集合 Z ⊆ V，使得 G-Z 是一个二分排列图且|Z| ≤ 9|Y|。我们按照如下方式构造 Z。我们从 Z = ∅开始。然后，只要 G - Z 包含一个同构于 K3，T2，X2，X3，C5，C6，C7，C8，C9 中的一个子图 X，我们就把 X 中的所有顶点添加到 Z 中。注意到 Y ∩ X = ∅且|X| ≤ 9。

完成这一步后，G－Z 就是一个几乎二分排列图。注意到|Z| ≤ 9|Z ∩ Y|。我们在 G - Z 中找到一个最短的圈 C = {c0，...，cm-1}，并找到一个最小圈割 X，如第 4 节描述的那样。由于(Y－Z)是 G－Z 的一个圈割，我们有|X| ≤ |Y－Z|。我们将 X 添加到 Z 中。注意到此时 G - Z 是一个二分排列图。

由于 K3，T2，X2，X3，C5，C6，C7，C8，C9 最多有 9 个顶点，所以|Z| ≤ 9|Y|。这意味着上述算法是一个 9-近似算法。它可以在多项式时间内运行，因为可以在多项式时间内找到小的禁止子图，并且可以在几乎二分排列图中找到最小圈割。

## 2.3 算法举例

论文中并未给出任何代码的实现，因此这里先大致写一下两个定理所用算法的伪代码，然后举个简单的例子应用：

### 2.3.1 定理一：

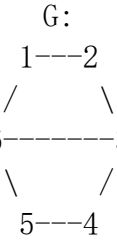
```
Algorithm: BipartitePermutationVertexDeletion
Input: G = (V, E) - 图, k - 顶点删除的最大数量
Output: 是否可以通过删除最多 k 个顶点将 G 转化为二分排列图

1. 判断 G 是否为连通图，若不是，则独立处理每个连通分量。
2. 对于每个连通分量 G'，计算其最短孔 C。
3. 对于每个四元素子集 S = {v1, v2, v3, v4}，其中 G'[S] 是 P4 induced，执行以下步骤：
   a. 构造图 G'，通过删除 N(v2) ∪ N(v3) \ {v1, v4} 中的顶点。
   b. 在 G' 中找到最短 v1-v4 路径。
4. 对于每个极小洞割 X，使得 X 包含在 G' 中的某个 V = VG[i-2, i+2] 中，执行以下步骤：
   a. 构造有向图 Hi，顶点集为 V × {in, out} ∪ {s, t}。
```

b. 添加边：(u, out)(v, in)，其中 uv 是 G'[V] 的边。
c. 添加边：s(v, in)，如果存在 u ∈ VG[i-4, i-3] 使得 uv 是 G' 的边。
d. 添加边：(u, out)t，如果存在 v ∈ VG[i+3, i+4] 使得 uv 是 G' 的边。
e. 添加边：(u, in)(u, out)，对于所有 u ∈ V。
f. 将边(u, in)(u, out)的容量设为 1，将所有其他边的容量设为 ∞。
g. 在 Hi 中找到最大流，得到极小(s, t)-割的最小大小 kG。
5. 判断是否所有考虑的连通分量 G' 上记住的 kG 的大小之和最多为 k。
6. 若是，则返回 true，否则返回 false。

算法举例：

考虑以下图 G，其中顶点集合为 V = {1, 2, 3, 4, 5, 6}，边集合为 E = {(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1)}。




我们的目标是通过删除最多 k 个顶点，使得 G 成为一个二分排列图。

步骤 1： 检查 G 是否为连通图，这里 G 是连通图。

步骤 2： 计算 G 的最短孔 C。在这个例子中，G 本身就是一个最短圈。

步骤 3： 对于每个四元素子集 S = {v1, v2, v3, v4}，其中 G[S] 是 P4 induced，执行以下步骤：

选择 S = {1, 2, 3, 4}，构造 G'，删除 N(2) ∪ N(3) \ {1, 4} 中的顶点。此时 G' 变为一个路径：1---2---3---4。

在 G' 中找到最短 1-4 路径：1---2---3---4。

对于其他可能的 S，执行相似的步骤。

步骤 4： 对于每个极小洞割 X，在这个例子中，G 中的唯一极小洞割是整个图 G，所以只有一个 X。

步骤 5： 检查是否所有连通分量 G' 上记住的 kG 的大小之和最多为 k。在这个例子中，只有一个连通分量 G'，且 kG = 0。

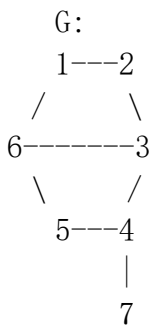
步骤 6： 返回 true，因为所有条件都满足，可以通过删除最多 k = 0 个顶点将 G 转化为二分排列图。

### 2.3.2 定理二：

```
Algorithm: BipartitePermutationSetConstruction
Input: G = (V, E) - 图，Y ⊆ V 使得 G - Y 是一个二分排列图
Output: Z ⊆ V，使得 G - Z 也是一个二分排列图且 |Z| ≤ 9|Y|
1. Initialize Z = ∅.
2. Repeat until G - Z 不再包含 K3，T2，X2，X3，C5，C6，C7，C8，C9 中的子图 X:
   a. For each X ∈ {K3, T2, X2, X3, C5, C6, C7, C8, C9}, do the following:
      i. Find all occurrences of X in G - Z.
      ii. Add all vertices of each occurrence of X to Z.
3. G - Z 变成一个几乎二分排列图。
4. Find the shortest cycle C = {c0, ..., cm-1} in G - Z.
5. Find a minimum cycle cut X as described in Section 4.
6. Add all vertices of X to Z.
7. G - Z 变成一个二分排列图。
8. Return Z.
```

算法举例：

考虑以下图 G，其中顶点集合为 V = {1, 2, 3, 4, 5, 6, 7}，边集合为 E = {(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1), (5, 7), (6, 7)}。




假设 Y = {1, 2, 3, 4}，我们知道 G - Y 是一个二分排列图。

步骤 1： 初始化 Z = ∅。

步骤 2： 重复以下步骤直到 G - Z 不再包含 K3，T2，X2，X3，C5，C6，C7，C8，C9 中的子图 X：

对于 X ∈ {K3, T2, X2, X3, C5, C6, C7, C8, C9}，检查并添加 X 中的所有顶点到 Z。

在这个例子中，可能会执行以下步骤：

检测到 G - Z 包含了 C5，将 {5, 6, 1, 3, 4} 添加到 Z。

步骤 3： G - Z 变成一个几乎二分排列图。

步骤 4： 在 G - Z 中找到最短圈 C，可能是 {1, 2, 3, 4, 5, 6}。

步骤 5： 找到最小圈割 X，可能是 {1, 2, 3, 4, 5, 6}。

步骤 6： 将 X = {1, 2, 3, 4, 5, 6} 添加到 Z 中。

步骤 7： G - Z 变成一个二分排列图。

步骤 8： 返回 Z = {1, 2, 3, 4, 5, 6}。

因此，通过从 Y = {1, 2, 3, 4} 开始，我们通过定理二的算法构造了 Z = {1, 2, 3, 4, 5, 6}，并且 G - Z 仍然是一个二分排列图。

# 三．讨论和分析

说实话，我真的看不懂这两个算法有什么不足还能怎么改进，下面的不足我也说的很笼统，只是觉得可能存在这些问题，而改进我则参考了网上的一些资料，改进后的也不一定正确

## 3.1 算法的不足

### 3.1.1 定理一：

定理一的算法可能存在的不足：

复杂性： 定理一的算法时间复杂度为 O(9^k • n^6)其中 k 是要删除的最大顶点数量。这是一个指数级别的复杂性，特别是当

k 较大时，算法可能变得非常慢。在实际应用中，指数级的复杂性可能使算法不适用于大型图。

确定性算法： 定理一的算法是一个确定性算法，这意味着在给定相同输入的情况下，它的输出总是相同的。这可能导致算法在处理某些图形时缺乏灵活性。对于某些图结构，可能存在更高效的随机化或近似算法。

特定情况： 定理一的算法是通用的，适用于任何二分排列图。然而，在某些情况下,可能存在更特定的算法,针对特定的图结构,能够更有效地解决问题。

常数因子： 复杂性中的常数因子可能对算法的实际性能产生影响。优化算法以减小常数因子可能是一种改进的方向。

未考虑其他性能度量： 算法的复杂性主要关注时间复杂性，但在实际应用中，还需要考虑其他性能度量，如空间复杂性和实际运行时间。

### 3.1.2 定理二：

上述定理二算法是一个有效的 9-近似算法，但它可能存在一些不足之处：

性能： 由于该算法涉及多项式时间的图操作，其运行时间可能在实际应用中不够高效，特别是在处理大规模图时。图算法的复杂性通常受到图的结构和大小的影响，因此可能需要进一步的优化。

近似度： 算法是一个 9-近似算法，这意味着删除的顶点数量可能是最优解的 9 倍。在某些情况下，这可能导致不必要的顶点删除，特别是当最优解的数量相对较少时。

特定情况： 算法的有效性可能受到图的具体结构的影响。对于某些类型的图，可能存在更有效的特定算法，而定理二的算法是一般性的。

图分割复杂性： 找到图中的最小圈割是一个 NP-难问题，算法在步骤 4 和步骤 5 中需要解决这个问题。对于某些图，这可能是计算上的瓶颈。

参数化复杂性：定理二的算法是一个参数化算法,其复杂性取决于参数 k。在某些情况下，参数化算法可能不够适用，尤其是当 k 较大时。

## 3.2 可能的改进

### 3.2.1 定理一算法改进

改进：优化图分割算法

原始算法中，图分割算法的复杂性对整体算法性能有重要影响。以下是一个优化

图分割算法的伪代码示例，其中采用更有效的最小割算法：

```
Algorithm: ImprovedBipartitePermutationVertexDeletion
Input: G = (V, E) - 图, k - 顶点删除的最大数量
Output: 是否可以通过删除最多 k 个顶点将 G 转化为二分排列图

1. 判断 G 是否为连通图，若不是，则独立处理每个连通分量。
2. 对于每个连通分量 G'，计算其最短孔 C。
3. 对于每个四元素子集 S = {v1, v2, v3, v4}，其中 G'[S] 是 P4 induced，执行以下步骤：
   a. 构造图 G'，通过删除 N(v2) ∪ N(v3) \ {v1, v4} 中的顶点。
   b. 在 G' 中找到最短 v1-v4 路径。
4. 对于每个极小洞割 X，使得 X 包含在 G' 中的某个 V = VG[i-2, i+2] 中，执行以下步骤：
   a. 优化的图分割算法，找到最小圈割 X。
5. 判断是否所有考虑的连通分量 G' 上记住的 kG 的大小之和最多为 k。
6. 若是，则返回 true，否则返回 false。
```

这个改进的例子主要集中在图分割算法上，采用了更优化的最小圈割算法。这样可以显著提高图分割的效率，减小整体算法的时间复杂度。

### 3.2.2 定理二算法改进

改进：基于贪婪的子图检测优化

原始算法中，子图检测可能是算法的一个瓶颈。以下是一个基于贪婪思想的优化伪代码示例：

```
Algorithm: GreedyBipartitePermutationSetConstruction
Input: G = (V, E) - 图，Y ⊆ V 使得 G - Y 是一个二分排列图
Output: Z ⊆ V，使得 G - Z 也是一个二分排列图且 |Z| ≤ 9|Y|

1. Initialize Z = ∅.
2. Repeat until G - Z 不再包含 K3，T2，X2，X3，C5，C6，C7，C8，C9 中的子图 X:
   a. For each X ∈ {K3, T2, X2, X3, C5, C6, C7, C8, C9}, do the following:
      i. Find all occurrences of X in G - Z.
      ii. Add all vertices of the first occurrence of X to Z.
3. G - Z 变成一个几乎二分排列图。
4. Find the shortest cycle C = {c0, ..., cm-1} in G - Z.
5. Find a minimum cycle cut X as described in Section 4.
6. Add all vertices of X to Z.
7. G - Z 变成一个二分排列图。
```

```
8. Return Z.
```

这个改进的例子采用了贪婪策略，每次只添加第一个发现的子图，而不是全部。这样可以减少子图检测的次数，提高算法的效率，特别是在图中包含大量子图的情况下。

## 3.3 作者留下的启发性的问题

问题 1：顶点删除问题对于排列图类和互可图类的参数化状态是什么？

问题 2：几乎排列图和几乎共可比图的结构是什么？

问题 3：二维偏序图的顶点删除问题的参数化状态是什么？

问题 4：几乎二维偏序集的结构是什么？

问题 5：对于二分置换顶点删除问题存在多项式核吗？

# 参考文献

[1]. Baker, K.A., Fishburn, P.C., Roberts, F.S.: Partial orders of dimension 2. Networks 2(1), 11–28 (1972)

[2]. Bo˙zyk, Ł, Derbisz, J., Krawczyk, T., Novotná, J., Okrasa, K.: Vertex Deletion into Bipartite Permutation Graphs. In: 15th International Symposium on Parameterized and Exact Computation (IPEC 2020),Leibniz International Proceedings in Informatics (LIPIcs), vol. 180, pp. 5:1–5:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020). https://doi.org/10.4230/LIPIcs.IPEC.2020.5. https://drops.dagstuhl.de/opus/volltexte/2020/13308

[3]. Brandstädt, A., Kratsch, D.: On the restriction of some NP-complete graph problems to permutation graphs. In: Fundamentals of Computation Theory, FCT ’85, Cottbus, GDR, September 9-13, 1985.Lecture Notes in Computer Science, vol. 199, pp. 53–62. Springer (1985). https://doi.org/10.1007/BFb0028791

[4]. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inf.Process. Lett. 58(4), 171–176 (1996)

[5]. Cao, Y.: Linear recognition of almost interval graphs. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12,2016, pp. 1096–1115. SIAM (2016). https://doi.org/10.1137/1.9781611974331.ch77

[6]. Cao, Y.: Unit interval editing is fixed-parameter tractable. Inf. Comput. 253, 109–126 (2017)

[7]. Cao, Y., Marx, D.: Chordal editing is fixed-parameter tractable. Algorithmica 75(1), 118–137 (2016)

[8]. Chudnovsky, M., Robertson, N., Seymour, P., Thomas, R.: The strong perfect graph theorem. Ann.Math. pp. 51–229 (2006)

[9]. Deogun, J.S., Steiner, G.: Polynomial algorithms for hamiltonian cycle in cocomparability graphs.SIAM J. Comput. 23(3), 520–552 (1994). https://doi.org/10.1137/S0097539791200375

[10]. Derbisz, J.: A polynomial kernel for vertex deletion into bipartite permutation graphs. CoRR arXiv:2111.14005 (2021)

[11]. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems.J. ACM (JACM) 19(2), 248–264 (1972)

[12]. Gallai, T.: Transitiv orientierbare graphen. Acta Mathematica Academiae Scientiarum Hungarica 18(1–2), 25–66 (1967)

[13]. Golumbic, M.C., Rotem, D., Urrutia, J.: Comparability graphs and intersection graphs. Discret. Math.43(1), 37–46 (1983). https://doi.org/10.1016/0012-365X(83)90019-5

[14]. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization.Algorithms and Combinatorics, vol. 2. Springer (1988). https://doi.org/10.1007/978-3-642-97881-4

[15]. Heggernes, P., van ’t Hof, P., Jansen, B.M.P., Kratsch, S., Villanger, Y.: Parameterized complexity ofvertex deletion into perfect graph classes. Theor. Comput. Sci 511, 172–180 (2013). https://doi.org/10.1016/j.tcs.2012.03.013

[16]. Heggernes, P., van ’t Hof, P., Lokshtanov, D., Nederlof, J.: Computing the cutwidth of bipartite permutationgraphs in linear time. SIAM J. Discret. Math. 26(3), 1008–1021 (2012). https://doi.org/10.1137/110830514

[17]. Kanesh, L., Madathil, J., Sahu, A., Saurabh, S., Verma, S.: A Polynomial Kernel for Bipartite

PermutationVertex Deletion. In: Golovach, P.A., Zehavi, M. (eds.) 16th International Symposium onParameterized and Exact Computation (IPEC 2021). Leibniz International Proceedings in Informatics(LIPIcs), vol. 214, pp. 23:1–23:18. SchlossDagstuhl – Leibniz-Zentrum für Informatik,Dagstuhl,Germany(2021).https://doi.org/10.4230/LIPIcs.IPEC.2021.23.
https://drops.dagstuhl.de/opus/volltexte/2021/15406
[18]. Kelly, D.: The 3-irreducible partially ordered sets. Can. J. Math. 29, 367–383 (1977)
[19]. Lekkeikerker, C., Boland, J.: Representation of a finite graph by a set of intervals on the real line.Fundamenta Mathematicae 51(1), 45–64 (1962)
[20]. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J.
Comput. Syst. Sci. 20(2), 219–230 (1980). https://doi.org/10.1016/0022-0000(80)90060-4
[21]. Lokshtanov, D.: Wheel-free deletion isW[2]-hard. In:Grohe,M., Niedermeier, R. (eds.) Parameterizedand Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16,2008. Proceedings. Lecture Notes in Computer Science, vol. 5018, pp. 141–147. Springer (2008).https://doi.org/10.1007/978-3-540-79723-4_14
[22]. Mancini, F.: Graph modification problems related to graph classes. PhD degree dissertation, Universityof Bergen Norway 2 (2008)
[23]. Marx, D.: Chordal deletion is fixed-parameter tractable. Algorithmica 57(4), 747–768 (2010). https://doi.org/10.1007/s00453-008-9233-8
[24]. Pnueli, A., Lempel, A., Even, S.: Transitive orientation of graphs and identification of permutationgraphs. Can. J. Math. 23(1), 160–175 (1971)
[25]. Spinrad, J.P., Brandstädt, A., Stewart, L.: Bipartite permutation graphs. Discret. Appl. Math. 18(3),279–292 (1987). https://doi.org/10.1016/S0166-218X(87)80003-3
[26]. Trotter, W.T., Jr., Moore, J.I., Jr.: Characterization problems for graphs, partially ordered sets, lattices,and families of sets. Discret. Math. 16(4), 361–381 (1976). https://doi.org/10.1016/S0012-365X(76)80011-8
[27]. van ’t Hof, P., Villanger, Y.: Proper interval vertex deletion. Algorithmica 65(4), 845–867 (2013).https://doi.org/10.1007/s00453-012-9661-3
[28]. Villanger, Y., Heggernes, P., Paul, C., Telle, J.A.: Interval completion is fixed parameter tractable.SIAM J. Comput. 38(5), 2007–2020 (2009). https://doi.org/10.1137/070710913