平面图中直线形模块的算法设计与基于图的分类

Rohit Lohani、Ravi Suthar、Krishnendra Shekhawat

比尔拉科技与科学学院数学系，皮拉尼，皮拉尼校区，维迪亚维哈尔，印度拉贾斯坦邦 333031

摘要。本文提出了一种图论框架，用于构建能够容纳非矩形模块的平面图，重点关注 L 形和 T 形几何形状。与主要处理外边界的传统方法不同，本文提出的方法纳入了实现这些更复杂形状模块时产生的结构约束。本文探讨了如何运用算法图论工具，将此类模块嵌入由三角剖分图导出的矩形平面图表示中。

分析表明，并非每个三角剖分图都能支持这些指定的模块几何形状。为刻画可行实例，本文提出了一项形状保持约束，以防止模块几何形状因边界变形而发生改变。此类变化会增加相邻模块的组合复杂度，或破坏预期的邻接结构。

本文提出了一种基于优先规范序的线性时间算法，可在平面图中构造 L 形和 T形模块。该方法适用于至少包含一个内部 K_4，或包含两个满足特定存在条件的内部 K_4 子图的三角剖分图。本文详细介绍了构造过程，概述了实现所需模块形状的条件，并展示了如何在最终平面图中生成这些模块。该算法简单且易于直接实现，适合集成到实际的布局生成工作流程中。未来工作包括将该框架扩展至更多模块形状，并探索更广泛的支持图结构类别。

关键词：图论、图算法、复杂三角形、三角剖分图、矩形平面图、正交平面图

arXiv:2601.00539v1 [math.CO] 2 Jan 2026

1 引言

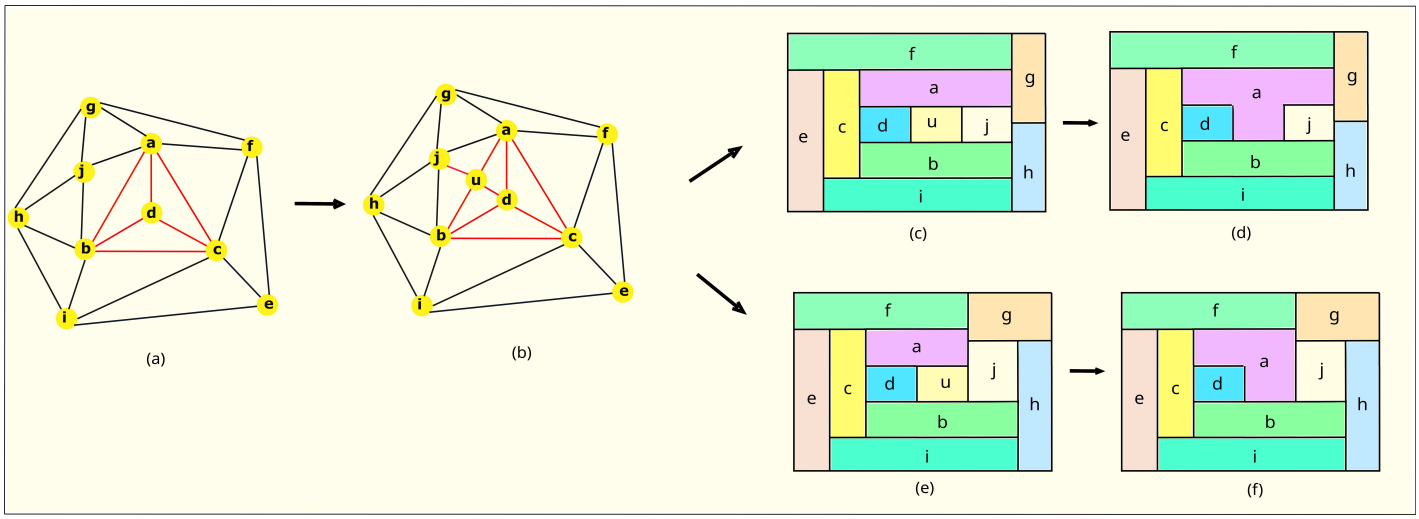


图 2：(a) 包含 K_4 （即 K_L）的输入图 G_L。(b) 通过添加附加节点 u消除复杂三角形后，由 G_L 得到的修改后的图。(c,e) 得到修改后的图的矩形平面图。(d,f) 将模块 u 合并至模块 a，从而在平面图中得到一个 L形模块和一个 T形模块。

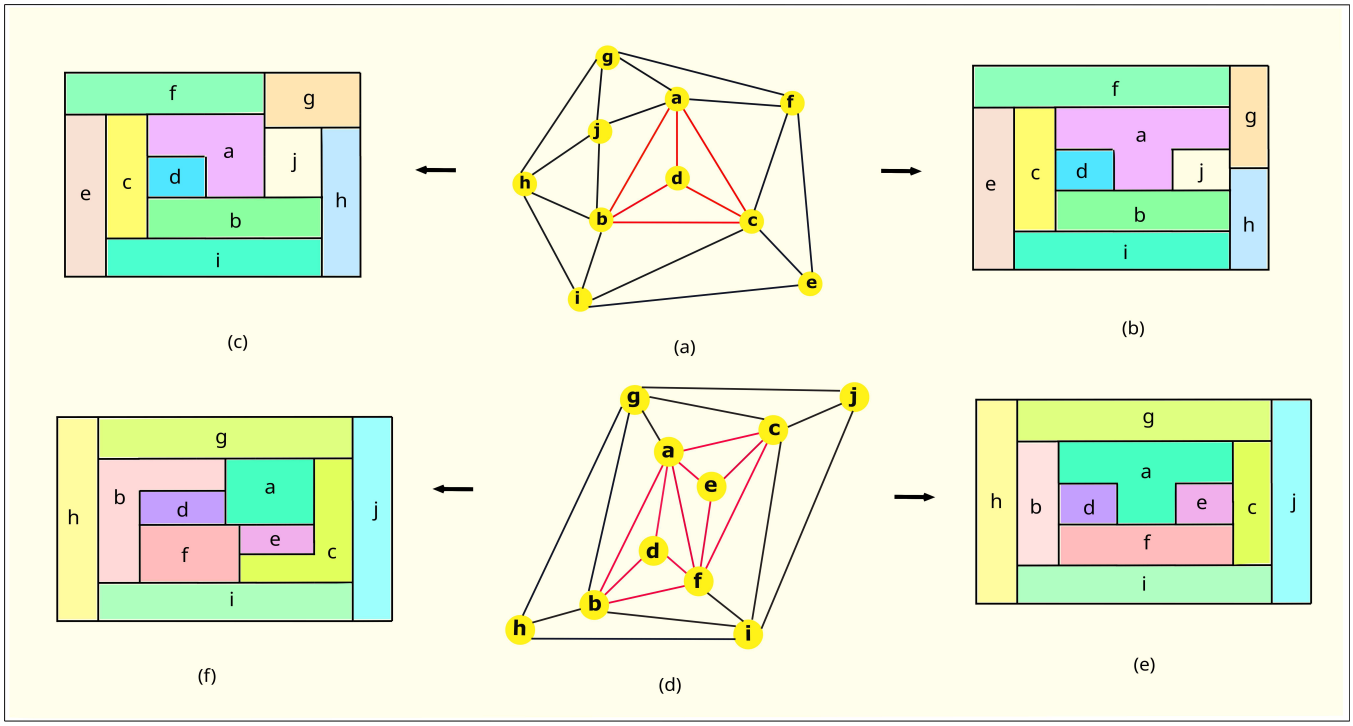


图1：(a) 包含 K_4（即 K_L）的输入图 G_L。(b,c) 包含与 G_L 对应的 L 和 T 形模块的平面图。(d) 包含两个共享一条公共边的 K_4（即 K_T）的输入图 G_T。(e,f) 包含一个 T 形模块和两个 L 形模块，分别对应 G_T 的平面图。

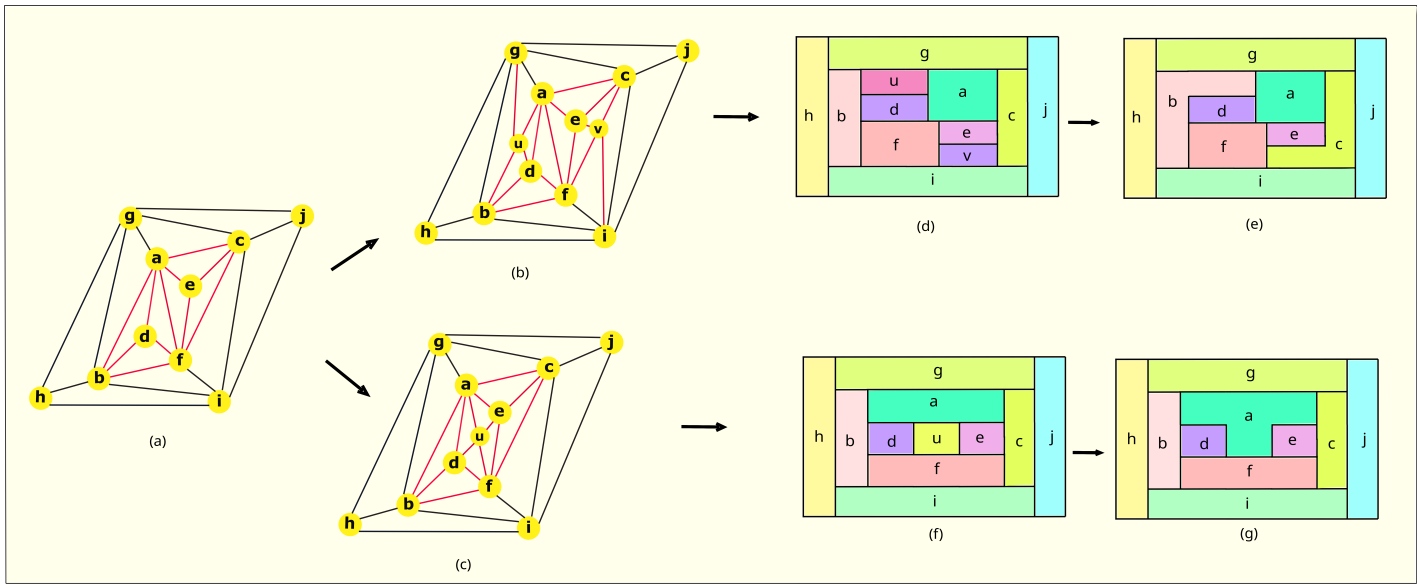


图3：(a) 包含两个共享公共边的 K_4（即 K_T）的输入图 G_T。(b, c) 展示了由 G_T 得到的两个修改后的图：通过引入一个节点（u）或两个额外节点（u 和 v），消除一个复杂三角形后得到。(d,f) 与修改后的图对应的矩形平面图。(e,g) 将模块 u 与模块 a 合并，得到一个 T 形模块；将模块 u 和 v 分别与模块 b 和 c 合并，在平面图中得到两个 L 形模块（b 和 c）。

平面图是指使用直线段将多边形空间划分为不同房间或模块。创建此类布局/平面图就像拼装一幅复杂的拼图，其中每个模块都是一块独特的拼片，而它们之间的空间关系则增加了复杂性。平面图设计长期以来一直是图论的重要应用，尤其是在超大规模集成电路芯片布局、建筑规划以及计算机科学的各个领域。在建筑领域，平面图对于确定施工场地的空间组织至关重要。尽管早期研究大多聚焦于由纯矩形模块构成的布局，但人们对包含非矩形形状的平面图日益感兴趣。这类平面图称为正交平面图。虽然已有多种生成此类布局的技术，但要开发出能够稳定生成特定非矩形模块的算法，仍是一项数学上复杂且要求很高的任务。

不含具有不超过四条角点指示路径的复杂三角形的三角剖分图 [1] 可以用矩形平面图表示。然而，如果一个图包含复杂三角形，那么由它生成的任何平面图都至少包含一个非矩形模块 [2]。因此，要在平面图中创建一个 L形模块，给定的图必须包含一个内部复杂三角形（见第5.2节），称为 K_L （如图1a所示，其中 $\triangle abc$ 是复杂三角形）。类似地，要构造一个 T形模块，输入图必须包含一种特定的内部构型，即两个共享一条公共边的复杂三角形（见第5.5节），称为 K_T（如图1b所示，由顶点 a、b、c、d、e和 f 诱导的子图即为 K_T）。其次， [3] 中提出的现有方法有一个显著局限：无法控制所生成模块的具体形状，也就是说，破坏一个复杂三角形并不一定会形成用户定义的目标 L形或 T形模块（见图2d、2f、3e、3g）。相比之下，本文提出的算法解决了这一问题，只需对输入图提出最低限度的要求（即存在 K_L 或 K_T），便可确保在平面图中有针对性地生成 L 或 T 模块，见图2f、3g。这是通过采用规范序实现的；规范序是一种成熟的方法，可用于对4-连通三角剖分图的顶点进行排序。

要在平面图中生成一个 L 形模块，首先通过添加辅助顶点和边，将输入图转换为 4-连通三角剖分结构。随后，针对修改后的 K_L 子图（参见第2节）中的顶点应用规范序，并依据预先定义的分类优先级方案（A–F 类，详见第5.3节）逐类考察，优先处理相应顶点。然后，利用这种基于优先级的规范排序构造矩形平面图，之后整合辅助模块，得到包含 L 形模块的最终平面图。类似地，要构造一个 T 形模块，需将输入图同样转换为 4-连通三角剖分图。应用规范序并构造矩形平面图后，再合并添加的模块，形成包含 T 形模块的最终平面图。

因此，我们提出两种线性时间算法，每种算法的复杂度均为 $O(n)$（其中 n 表示顶点数），即算法1和算法3。这些算法旨在针对给定输入图，在平面图中生成 L 形或 T 形模块；该输入图至少包含一个内部子图，类型为 K_L 或 K_T。
本文的结构安排如下：第2节介绍作为本研究基础的基本定义和符号说明。第3节详细回顾相关文献，第4节则分析现有研究中的空白。第5节（方法论）概述输入图为生成所需模块 L 和 T 必须满足的条件，并介绍通过所提出算法构造这些模块的相应过程。第6节和第7节全面分析各算法的正确性和计算复杂度。最后，第8节总结本文，并讨论本研究的局限性及未来工作的潜在方向。

2 术语

本节定义本文使用的关键术语和符号说明，以确保后续关于基于图和平面图相关概念的讨论清晰且一致。

图 [4] G 由两个部分组成：一个顶点集 $V(G)$ 和一个边集 $E(G)$。顶点集是一个有限且非空的节点集合。与之相对，边集包含基数恰为 2 的子集 $V_i \subset P(V(G))$，用于表示顶点之间的连接。如果一个图可以绘制在二维平面上，且边之间没有交点，则称其为平面图。以这种方式绘制的平面图称为平面图 [4], ，这种表示会将二维平面划分为称为面的区域。无界区域称为外面，其他所有区域则称为内面。

定义 1. k-连通图：如果任意一对顶点之间都存在路径，则称该图为连通图。这个概念还可以进一步推广为 k-连通图。具体而言，阶大于 k 的图在移除少于 k 个顶点后仍保持连通，则称为 k-连通图（见图4）。

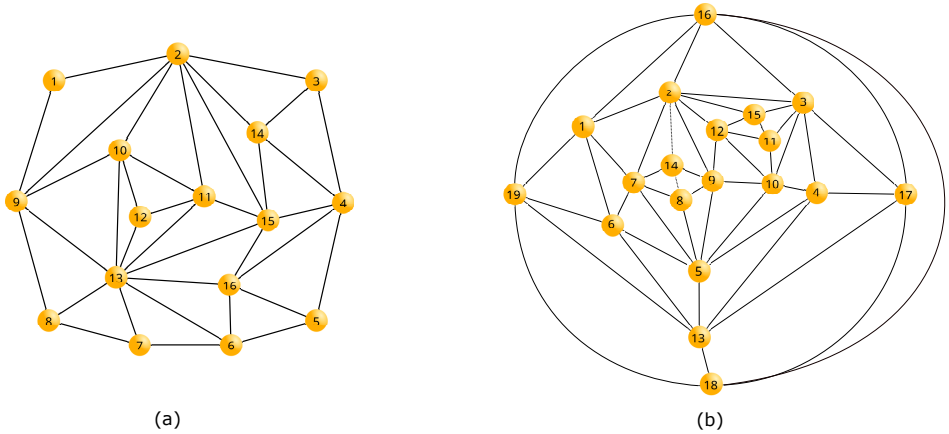


图4：（a）一个3-连通图。（b）一个4-连通图。

定义2. 平面三角剖分图 *[PTG] [1]*：如果一个双连通图的每个面（外部面除外）都是三角形，则称其为PTG（见图5a）。

定义3. 复杂三角形 *[CT]*：CT是一个长度为三的圈，且其内部至少包含一个顶点（见图5a）。

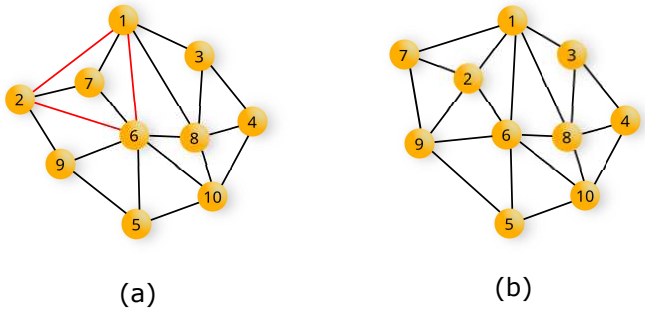


图5：（a）一个含有复杂三角形(1, 6, 2)的PTG。（b）PTPG。

如果平面三角剖分图（PTG，见定义2）不含复杂三角形，则称其为正规平面三角剖分图（PTPG）（见图5b）。

定义4. 规范序 *[1]*：设 G 是一个四连通平面三角剖分图（PTG），其中包含三个不同的外部顶点，分别记为 N、S 和 W。G 的规范序是一个顶点排序(v_n、v_{n-1}、...、v_1)，满足以下条件：

(i) 顶点 v_1、v_2 和 v_n 必须分别对应于 W、S 和 N。对于 $4 \leq j \leq n$，由 $\{v_1, v_2, ..., v_{j-1}\}$ 诱导的子图 G_{j-1} 是双连通的，其外部面形成一个包含边 (W, S) 的圈 C_{j-1}。(ii) 顶点 v_j 严格位于 G_{j-1}、($i.e., v_j \notin V(G_{j-1})$ 的外部。此外，v_j 在 G_{j-1} 内的所有邻点都必须沿路径 $C_{j-1} \setminus (W, S)$ 连续排列，且此类邻点至少有两个。

(iii) 对于 $j \leq n-2$，必须确保 v_j 在图 $G \setminus G_{j-1}$ 中至少有两个邻点，从而在排序过程中保证渐进连通性。（参见图6，其中由输入图 G 生成了一个规范有序图）。

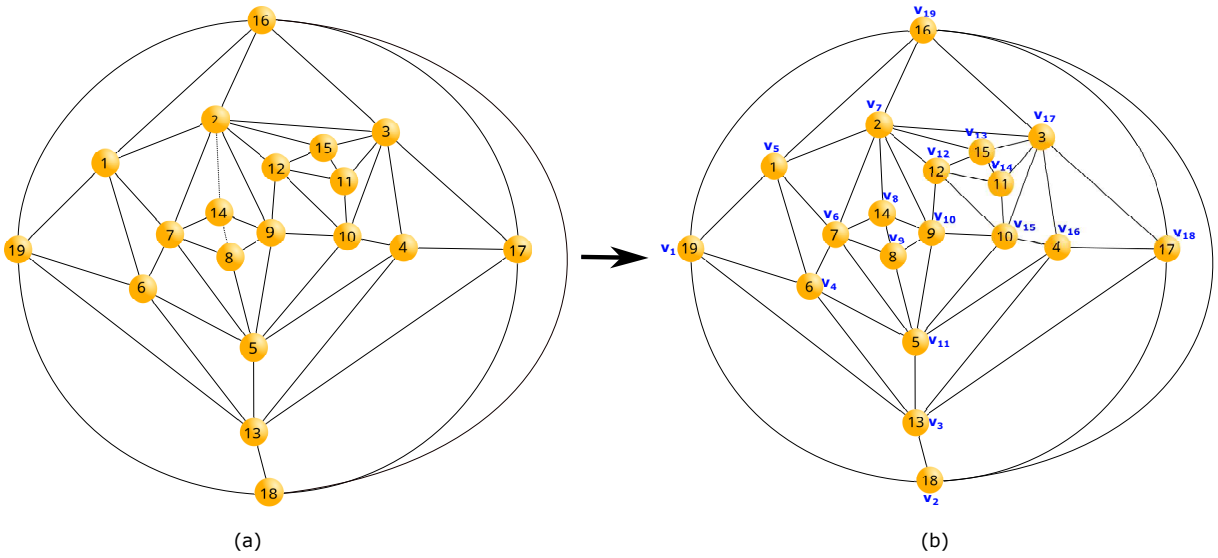


图 6：（a）一个 4-连通平面图 G。（b） G的规范有序图。

定义 5。正则边标记 *[REL] [5]*：对于具有四个外部顶点 N、W、S、E（按逆时针顺序排列）的双连通 PTPG G，正则边标记构成其内部边的划分和定向，将其划分为两个不相交的子集 T_1 和 T_2，并满足以下条件：

（I）对于每个内部顶点 v，关联边围绕 v 按逆时针方向排列，顺序如下：
- 指向 v：入边。
- 背离 v：出边
- 背离 v：出边
- 指向 v：入射的 T_2 有向边

(ii) 与顶点 N 相接的边属于 T_1，且指向 N。相反，与 W 相接的边属于 T_2，且背离 W。与 S 相接的边包含在 T_1 中，且背离 S；而与 E 相接的边属于集合 T_2，且指向 E。

见图 7 (a-b)，对于输入图 G，可生成正则边标记。对于 PTPG G 的每一种正则边标记（REL），都存在一个矩形平面图 $\mathcal{F}$（见图 7 (b-c)）。在此情境下，G中的每个顶点都对应于 $\mathcal{F}$中的一个矩形模块。此外，子集 T_1 中的有向边表示模块之间的垂直墙共享，这些模块沿 $y-axis$对齐；而子集 T_2 中的有向边表示模块之间的水平墙共享，这些模块沿 $x-axis$对齐。

定义 6。平面图（$\mathcal{F}$） *[[6]]*：平面图通过直线段将多边形分解为较小的组成多边形。外部多边形称为平面图的边界，而较小的组成多边形称为模块。如果两个模块共享一段墙段，则称它们相邻；仅有点接触（四个接点）不构成相邻关系。平面图中的一个特殊类别是矩形平面图（RFP），其中边界和所有模块均为矩形（见图 7c）。另一种推广形式是正交平面图（OFP）。在 OFP 中，边界为矩形，但与矩形平面图不同，OFP 中的模块可以是直角线形状，例如 L 形或 T 形多边形，前提是所有边均保持轴对齐（见图 10b ）。

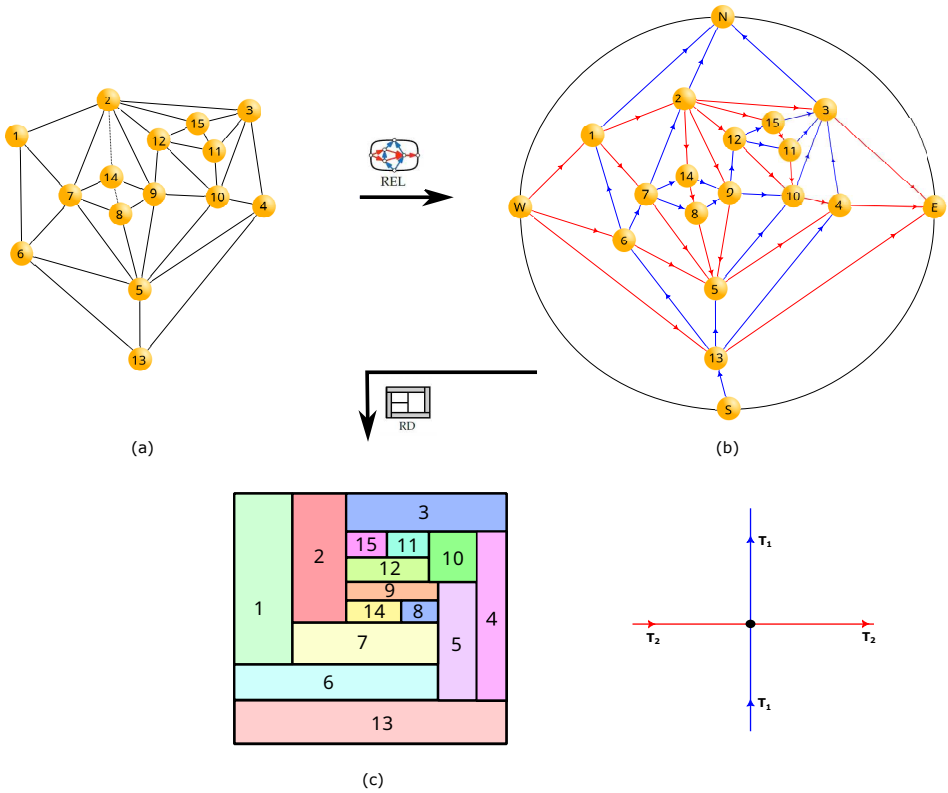


图7: （a）一个 PTPG G。（b ）图的正则边标记。（c）使用该图对应的正则边标记生成的平面图 G.

符号说明:

1. G_L: 输入三角剖分平面图至少包含一个内部复杂三角形（K_L），且外部面的长度为 ≥ 3 （见图2a）。2. G_T: 输入三角剖分平面图至少包含一个子图（K_T），且外部面的长度为 ≥ 3 （见图3a）。3. n: 输入图中的节点总数为 G_a （可以是 L 或 T）。4. ST_a （可以是 L 或 T）：包含一组复杂三角形的集合 s_i，其数量为 G_a。5. M（l、m、F_a）：在平面图 F_a 中，将模块 l 与模块 m 合并。6. K_L: 复杂三角形 K_4 （ K_L 的顶点按以下顺序排列：首先是 a，然后是 b，接着是 c，最后是d，在 G_L 中按逆时针顺序排列）（见图2a）。7. K_T: 共享一条边的两个复杂三角形（K_4）（ K_T 的顶点按以下顺序排列：首先是 a，然后是 b，接着是c、 d、 e，最后是 f，在 G_T 中按逆时针顺序排列）（见图3a）。8. $modified\ K_L$: 复杂三角形 K_L 的外部边（a、b）通过插入新顶点 u 进行细分，并引入额外的边以保持三角剖分，从而得到 $modified\ K_L$ （图2b）。9. $modified\ K_T$: K_T 的内部边（a、f）通过插入新顶点 u 进行细分，并引入额外的边以保持三角剖分，从而得到 $modified\ K_T$ （图3c）。10. G_a^1 （其中 a 是 L 或 T）：通过向 G_a 添加附加顶点和边而形成的 4-连通图。11. $G_{a_r}^1$: 由集合 v_1、 v_2、 ...、 v_r 定义的 G_a^1 的顶点诱导子图。12. C_r^a: 表示 G_{ar}^1 的外部面边界的一个圈，由边（ v_1、 v_2）组成。13. G_a^2: 从生成图 G_a^1 导出的规范序图。14. F_a'（其中 a 是 L或 T）：结构上表示图 G_a^2 的矩形平面图。15. F_a: 包含 a形模块、对应于输入图 G_a 的正交平面图。16. S_a: 从 G_a 中移除复杂三角形所需的边的子集，但不包括 K_a 子图，即 S $= \{$（a_1、b_1）、.....（a_i、 b_i）$\}$。

17. $Enodes_a$ （其中 a 是 L 或 T）：为移除 G_a 中的复杂三角形（不包括 K_a 子图）而引入 G_a 的一组额外顶点，即 $extra\ nodes = \{u_1, .., u_i\}$；其中，每个 u_i 都添加在 G_a中，对应于 S_a的每一对（a_i， b_i）。

18. $PLabel$（L）：按降序为集合 L （$L = \{p_1, p_2, p_3, p_4\}$）中的顶点赋予规范序，从 p_1开始，然后是 p_2，接着是 p_3，最后是 p_4。

3 文献综述

1. 建筑图论的起源：20世纪末，Paul 和 Grason 使用图来表示建筑平面图。

参考文献	贡献者	方法	洞见
[7]	Paul Levin（$1960s$）	图论方法	本文介绍了用于 建筑学中空间邻接关系的对偶图，作为表征方法。 将房间表示为顶点，并将相邻关系表示为 边。Levin 的框架将 抽象 平面 图转换为连通 图。然而，理论框架 缺乏计算实现。
[8]	John Grason（1970年）	图论方法	Grason 将自动平面图 综合形式化为使用对偶图。他将其研 究发现通过实验加以验证 tal 计算机辅助设计工具 GRAMPA。但它仅限于 轴对齐矩形模块。

2. 基础理论：这一方向的一项重要进展发生在20世纪80年代，当时研究人员开发了根据抽象邻接图自动生成矩形平面图的方法。

[4]	Kozminski 和 Kinnen $(1984 - 1985)$	图论方法	他们提出了矩形对偶所需且充分的 条件，其中包括平面三角剖分以及 排除复杂三角形。 他们提出了一种用于验证三角剖分并生 成二次时间复杂度的 $O(n^2)$ 算法 的算法。 矩形平面图的生成。
[9]	Bhasker and Sahni （1988年）	图论方法	作者开发了一种算法，能够 在构建矩形对偶和检查 构造矩形对偶并进行检查 然而，矩形对偶（平面图）的坐标 由所提出的算法生成， 是实数，且与结构 是真实存在的，与结构无关。 输入平面图的结构。
[1]	Xin He（1995年）	图论方法	作者讨论了一种并行算法， 用于构造矩形对偶 针对 $O(log^2 n)$ 中的平面三角图 在 CRCW 上使用 $O(n)$ 个处理器所需的时间 PRAM
[5]	Goos Kant, Xin He （2019）	图论方法	本文提出了两种线性时间 $O(n)$ 算法，即边收缩和 规范序， 用于构 造正则边标记（REL），以适用于 4-连通三角剖分平面图。 这些算法确保所构造矩形对偶的坐标 所构造矩形对偶的坐标 为整数，从而解决了一个显著的局限—— —— [9]的局限。

3. 矩形平面图（RFP）的最新进展：多年来，平面图生成逐步从优化技术和基于规则的图变换发展到先进的机器学习和深度学习方法。早期方法包括利用图算法和数学优化来复现平面图。随后，混合方法整合了进化算法和贪心算法，

通过强化学习实现更好的约束处理。最近，深度学习和图神经网络能够生成逼真且符合约束条件的平面布局，标志着设计系统正转向数据驱动的智能化方向。

[10]	X. Wang *et al.* (2018)	图变换	作者提出了一种图形化的设计生成方法（GADG）基于现有的传统平面图生成矩形平面图采用 PTPG 对偶图的平面图。这种方法采用矩形对偶查找技术，自动重新生成一组新的平面图，这些图可以进一步优化和定制。
[11]	M. Nisztuk, P. Myszkowski（2019）	基于贪心 and 进化式 ap-方法	作者提出了一种混合框架，将进化算法与贪心算法相结合，用于自动生成满足邻接和尺寸约束的平面图。进化算法优化房间序列、缩放比例和轴变换，而贪心方法则根据这些参数逐步放置房间，确保邻接关系且互不重叠。
[12]	Feng Shi *et al.*（2020）	图论方法与强化-化学习	作者提出了蒙特卡洛树搜索方法，该方法基于强化学习算法。要使用此方法，需要构建一棵决策树，将矩形平面图的布局和约束定义为输入，并将优化后的布局定义为输出。
[13]	N. Upasani *et al.* (2020)	图论与数学优化化	这项研究介绍了一种计算方法，可在保留现有矩形布局所体现的邻接关系的同时，生成尺寸确定的矩形平面图。作者采用垂直和水平流网络上的线性优化技术，以满足用户定义的尺寸约束，并获得满足给定邻接关系的最小面积可行解。然而，文中未讨论该解是否最优。
[14]	Ruizhen Hu *et al.* (2020)	机器学习方法与深度神经网络	作者提出了一种学习框架，用于自动生成平面图，并纳入用户定义约束（边界、房间数量和邻接关系），以生成从 RPLAN 训练数据集中提取的布局图。随后，这些布局图会根据输入边界进行调整。基于图神经网络（GNN）的 Graph2Plan 模型会生成相应的平面图。
[15]	Lufeng Wang *et al.* (2023)	深度学习和图论技-术	作者提出了一种融合深度学习与图算法的框架，用于自动生成建筑布局。该算法基于独特的 GeLayout标注数据集进行训练。系统采用欧氏距离、Dice系数和力导向图算法来优化布局选择。

[16]	J. Liu *et al.* (2024)	图论方法 学习技术	and Deep	本文提出了一种框架，该框架 采用图约束生成对抗 网络（GC-GAN）专门—— 地生成 模块化住宅 与居住建筑（MHRB）平面图 。该 GC-GAN 包含知识 图，以确保生成的 平面图切实可行。它还融入了 评估一种图像转矢量转换算法 与扁平化设计 标准化库的兼容性。一个重要方 面是自动推进 遵循模块化标准的建筑信息模型 以实现高效构建。

4. 向非矩形模块的过渡（20世纪90年代至2020年代）：从矩形平面图（1993年）向OFP的转变，借助图论的发展纳入了非矩形房间。早期研究聚焦于为矩形平面图剪除复杂三角形，以及验证 L形模块。沿袭这些方法，研究者开发了线性时间算法，从而纳入了 T形和 I形模块（1999年至2003年），并实现了包括生成树在内的优化（2003年）。对直线多边形（2011）和六边形镶嵌（2012）的探索进一步提升了几何灵活性。近年来，相关方法从障碍消除发展到模块合并，最终演进为拓扑操作；在适应不规则轮廓和用户定义几何形状的同时，时间复杂度也从二次降至线性。

[17]	S. Tsukiyama *et al.*（1993年）	图论方法	本文证明，某些 平面三角剖分图（PTG）无法转化为 标准矩形平面图（RFP） 由于嵌入了复杂三角形。为 解决这一局限，他们提出了 一种算法，用于移除这些阻碍性 子结构，并在修剪后的图上重建 矩形平面图。他们的算法 的运行时间为二次复杂度 $O(n^2)$。
[2]	Y. Sun *et al.*（1993年）	图论方法	本文提出一种算法 for 给定的图是否存在 L 形 对偶，其判定复杂度为 nation being $O(n^{\frac{3}{2}})$。如果一个 L 形模块 若存在模块，则可以构造矩形 平面图，其中包含一个呈 L 形的模块 ，时间复杂度为二次 $O(n^2)$。
[18]	X. He（1999）	图论方法	该论文提出了一种线性时间算法 构建平面图的算法 仅使用1矩形和2矩形 模块。研究结果表明 相较于以往研究取得了明显进展 [2] 由 Yeap 和 Sarrafzadeh 开展， 该研究证明 PTG 可以 用 1、2 和 3 个矩形表示 模块。
[19]	M. Kurowski（2003年）	图论方法	本文提出了一种算法，可在线性 $O(n)$ 时间内 计算平面图。该理论采用由两个矩形合并形成 的模块： $T-$、 $L-$ 或 I 形。生成的平面图 尺寸最多为 $n \times n - 1$。
[20]	CC Liao *et al.*（2003年）	图论方法	本文介绍了一种算法，该算法基于有序生成树， 将规范序扩展到无需三角剖分的平面图。这种 方法避开了复杂的矩形对偶阶段，并能在线性 时间内计算出一个有序对。

[21]	MJ Alam *et al.* (2011)	图论方法	本文研究了使用直线多边形且不浪费面积的比例接触表示。作者提出了一种新算法，可确保生成的直线多边形有10条边，并以线性 O(n) 时间运行。这些结果改进了先前的研究；先前的研究声称能在时间复杂度为 $O(nlogn)$ 的情况下生成有12条边的直线多边形。此外，他们还提出了一种线性时间算法，用于对具有8条边的直线多边形的平面3树进行比例接触表示，并证明这是最优的。
[22]	CA Duncan *et al.* (2012)	图论方法	The authors present a study demonstrat-明六边形是描绘.....所必需且充分的.....平面图，而这些图是五边形无法表示的。可以在.....上构造相互接触的六边形表示图 G 的过程可在线性时间内完成，使用具有凸区域的 $O(n) \times O(n)$ 网格。
[23]	K. Shekhawat *et al.* （2017）	图论方法	作者提出了一种基于图论的框架，用于生成直线形平面图中的非矩形轮廓，同时满足房间邻接关系和尺寸约束。构建 upon 先验矩形模型，支持复杂的布局，例如医院和办公室，通过多边形边界和用户定义的邻接关系。局限性包括缺乏实时适应性，以及有限的多层整合。
[3]	K. Shekhawat *et al.* （2023）	图论方法	作者提出了一种创新的算法 用于生成直角形平面图。这项研究推进了先前聚焦于正交平面图，且未考虑特定的房间形状。本研究提出的框架利用复杂三角形来生成 L形、 T形、 C形 F形、阶梯形和十字形（十字形）房间。然而，该研究所提出的算法不具备生成特定形状的能力，无法为给定的 PTPG 生成

4 文献空白的比较综述

基于图论方法的自动平面图生成早期研究 [[4], [9], [1], [5]] 主要集中于创建矩形平面图。随着该领域的发展，[[22], [3]] 出现了用于生成更复杂非矩形模块的算法，包括利用带有复杂三角形的图分类来生成 L形、 T形、 C形、 F形、阶梯形和加号形（十字形）房间。这些框架主要通过首先识别图中的复杂三角形，依据特定规则细分其边，然后合并所得模块来形成非矩形模块。然而，这些方法存在一个显著局限：对于给定的输入图，它们无法保证生成特定的非矩形形状；对于同一个图和细分过程，可能会不可预测地产生不同的非矩形形状。因此，这些框架无法对最终的模块形状进行精确控制，不适用于需要特定房间几何形状的应用。

相比之下，我们的工作着重于通过整合矩形组件，系统地生成 L 和 T形模块。我们提出了更精细的图分类，并通过结合复杂三角形的操作与基于优先级的规范排序，确立了构造 L 和 T 形状的必要条件。我们的方法认识到，L 和 T 形状都可能源自同一种图分类；然而，要实现预期结果，必须按照特定顺序放置与复杂三角形相关的模块。通过引入正则边标记（REL）并优先采用规范排序，我们确保生成过程兼具确定性和高效性。

与以往研究相比，我们的研究取得了一项显著进展 [[3], [23]]： 大幅降低了计算复杂度。我们提出的方法能够线性时间生成 L 形和 T形房间。这不仅保证了

对于给定的图，所需模块形状的存在性，也使该方法能够扩展并适用于实际建筑设计应用。 L 形和 T 形归为同一类基础图，它们的区别仅在于使用正则边标记（REL）和优先规范排序进行进一步分类。因此，有必要针对 L 形和 T 形采用单独的算法方法。

5 方法论

本节概述一种线性时间算法，该算法以外部面至少有三条边的三角剖分平面图为起点，在平面图中构造 L形或 T形模块。该算法依赖于一个或多个内部复杂三角形（即与 K_4同构的子图）的存在，具体取决于要生成的模块。在接下来的章节中，我们将分别介绍用于生成各类模块（即 L 形或 T 形模块）的算法。需要强调的是，只有当图中包含可引导构造过程的必要复杂三角形时，才能生成此类模块。

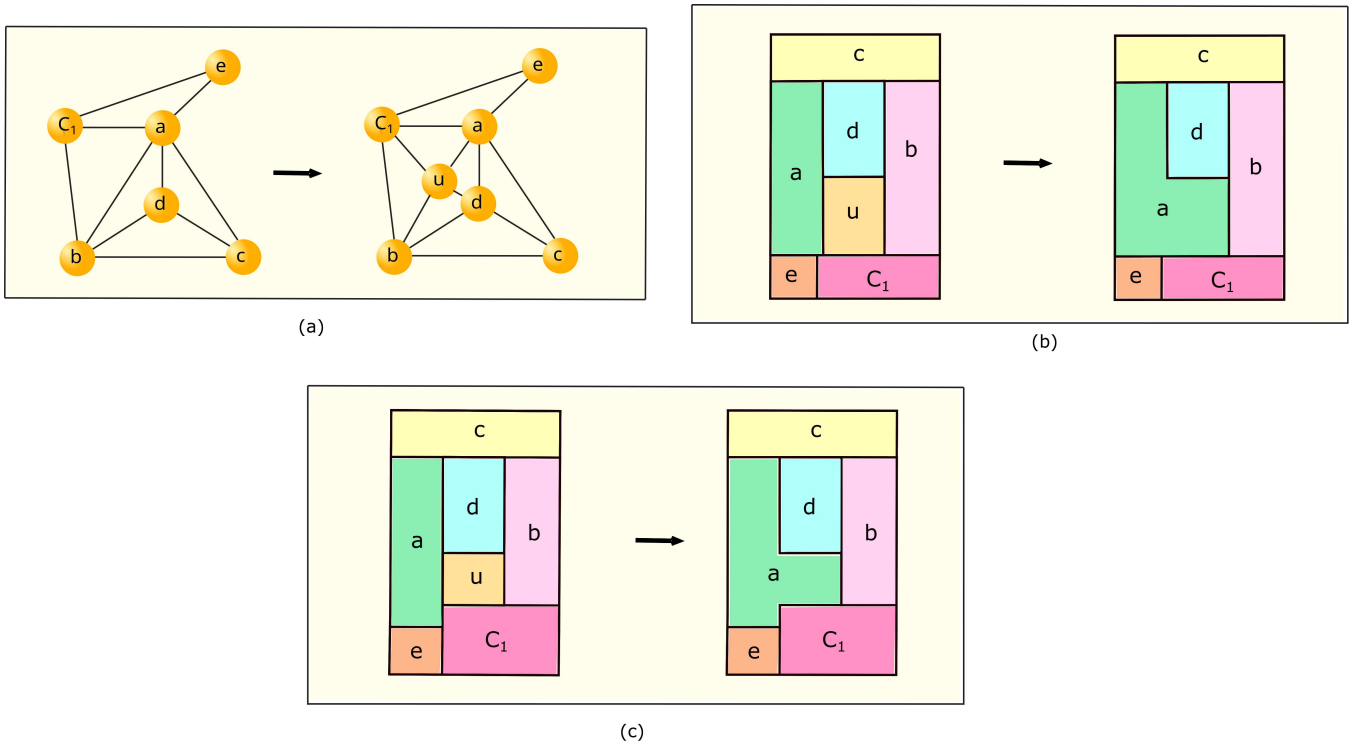
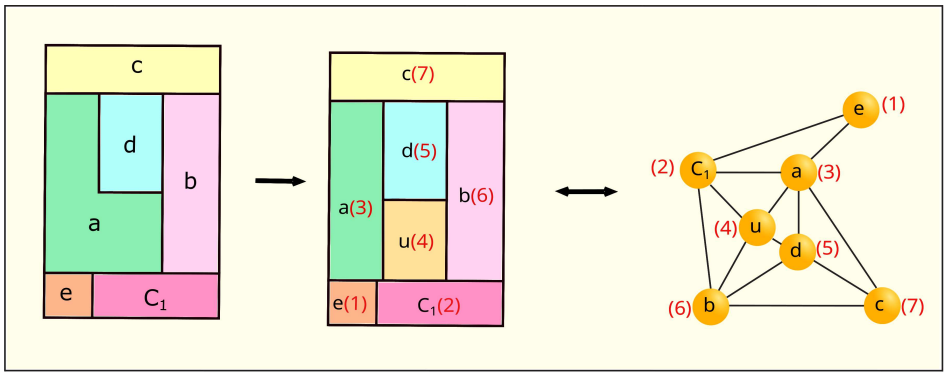


图8：（a）在输入图 K_L 中插入一个附加顶点 u，以消除复杂三角形。（b–c）在所得平面图中生成一个 L形模块，同时为相同的 $modified\ K_L$ 图创建一个平凡的 T形模块。

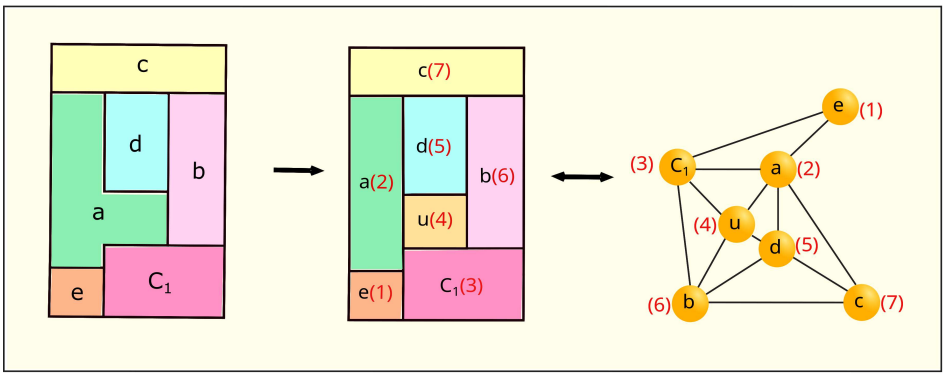
5.1 我们提出的 L形模块生成工作的概述

通过按照 [5] 中描述的方法细分 K_4 子图的一条外边，可以生成各种矩形平面图。合并由消除复杂三角形 K_4 而产生的额外模块 u 后，便可生成同时包含 L 形模块和普通 T 形模块的各种平面图（见图8）。这表明，仅打断复杂三角形的一条边，并不一定会在平面图中形成 L 形模块。这意味着，当图包含作为子图的 K_4 时，需要额外的步骤或方法，才能确保最终平面图中生成特定的 L 形模块。

此外，我们发现，[5] 中描述的规范顶点序，在表示包含 L 形模块的平面图的图与表示包含普通 T 形模块的平面图的图之间有所不同（见图9）。这表明



(a)



(b)

图9：（a-b）利用 [5], 中的算法从平面图生成的规范有序图，其中包括 L形模块和普通 T形模块。

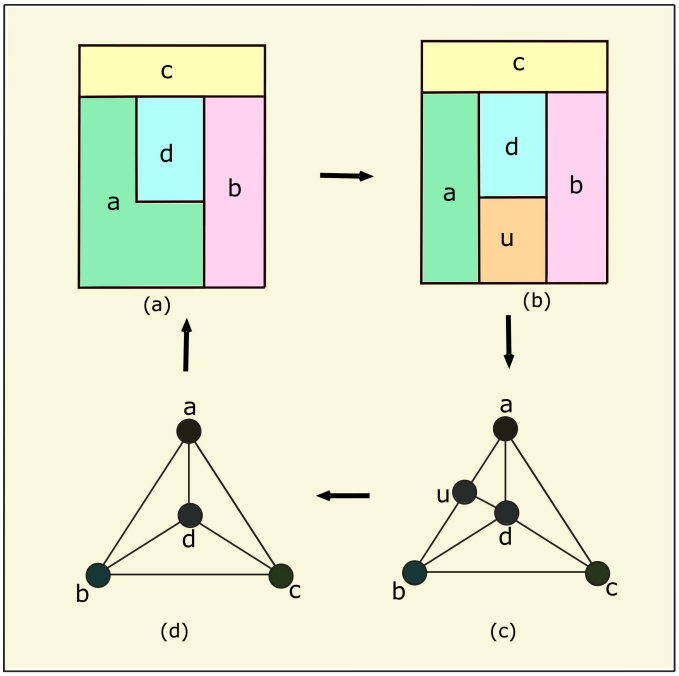
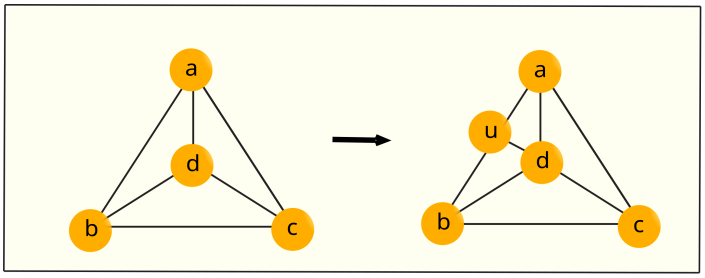
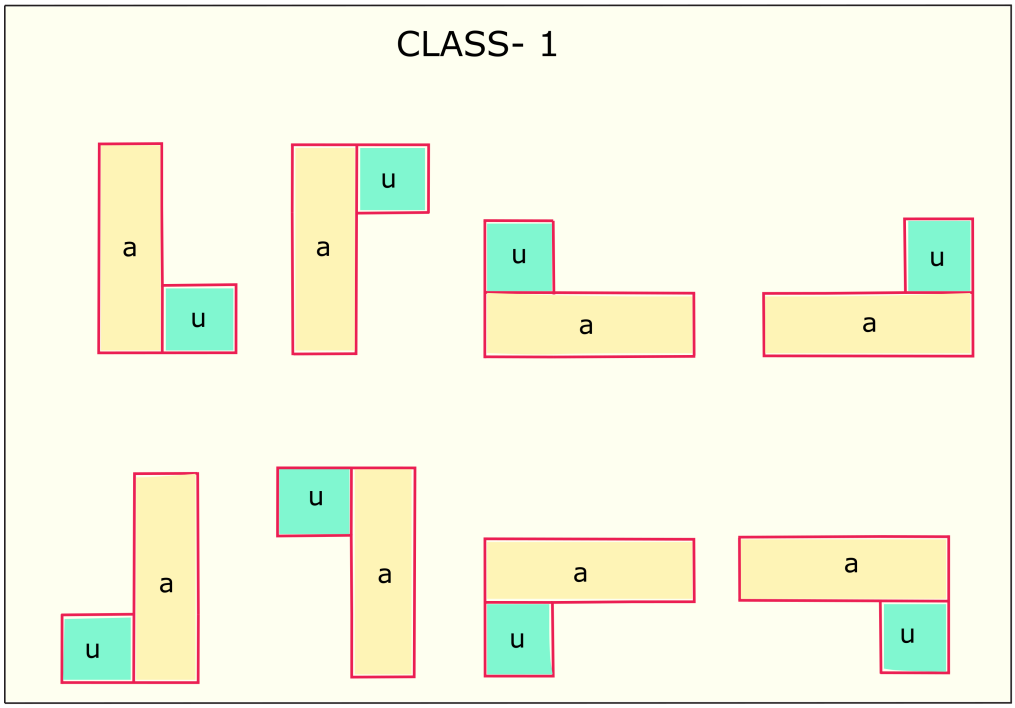


图10：（a-d）生成 L形模块所需的内部复杂三角形 K_L

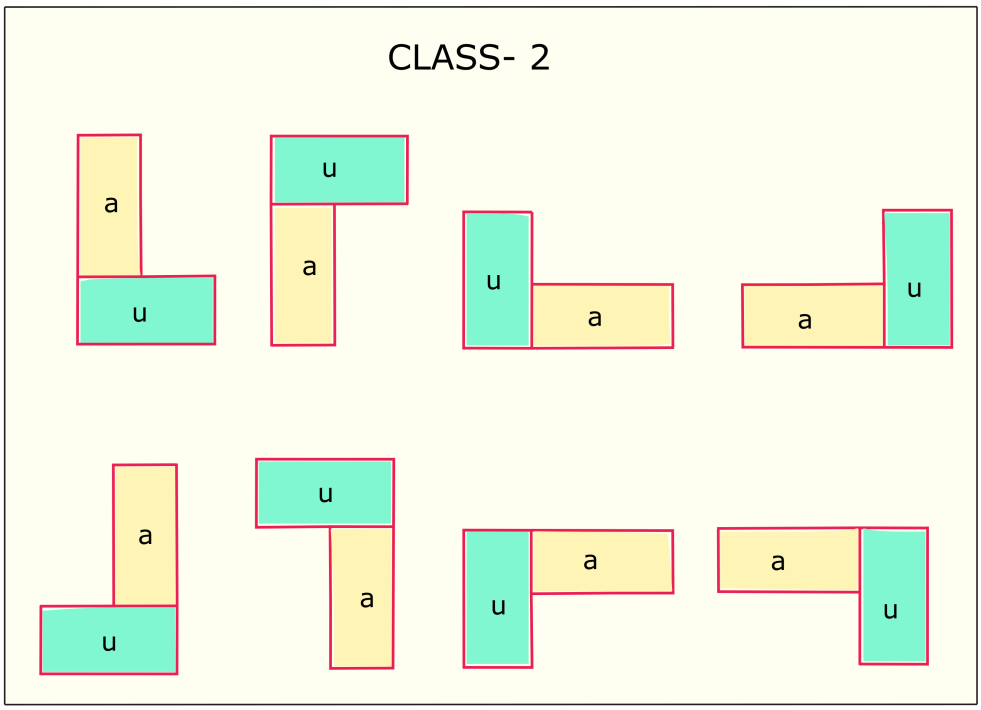
le.



(a)



(b)



(c)

图11：（a-c）将模块 u 与模块 a合并的几种方法。

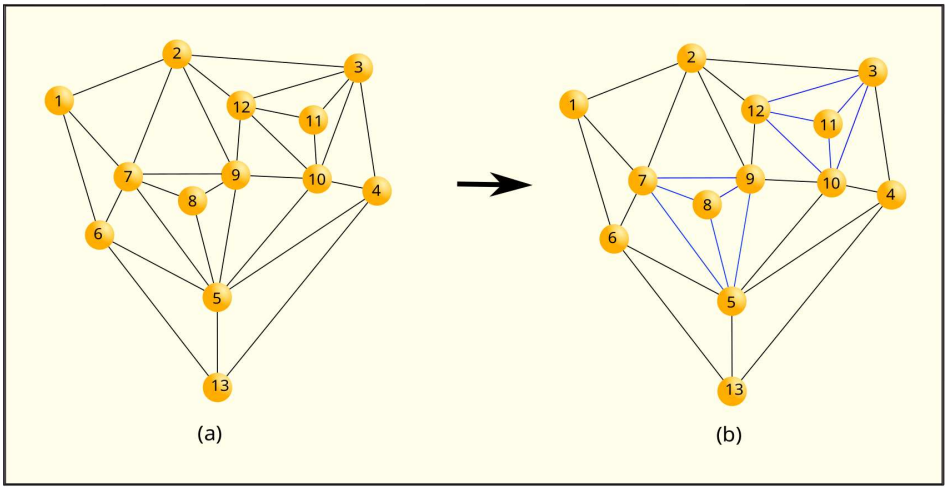


图12：（a-b）在 G_L^1中识别复杂三角形（即 K_L 及其他部分）。

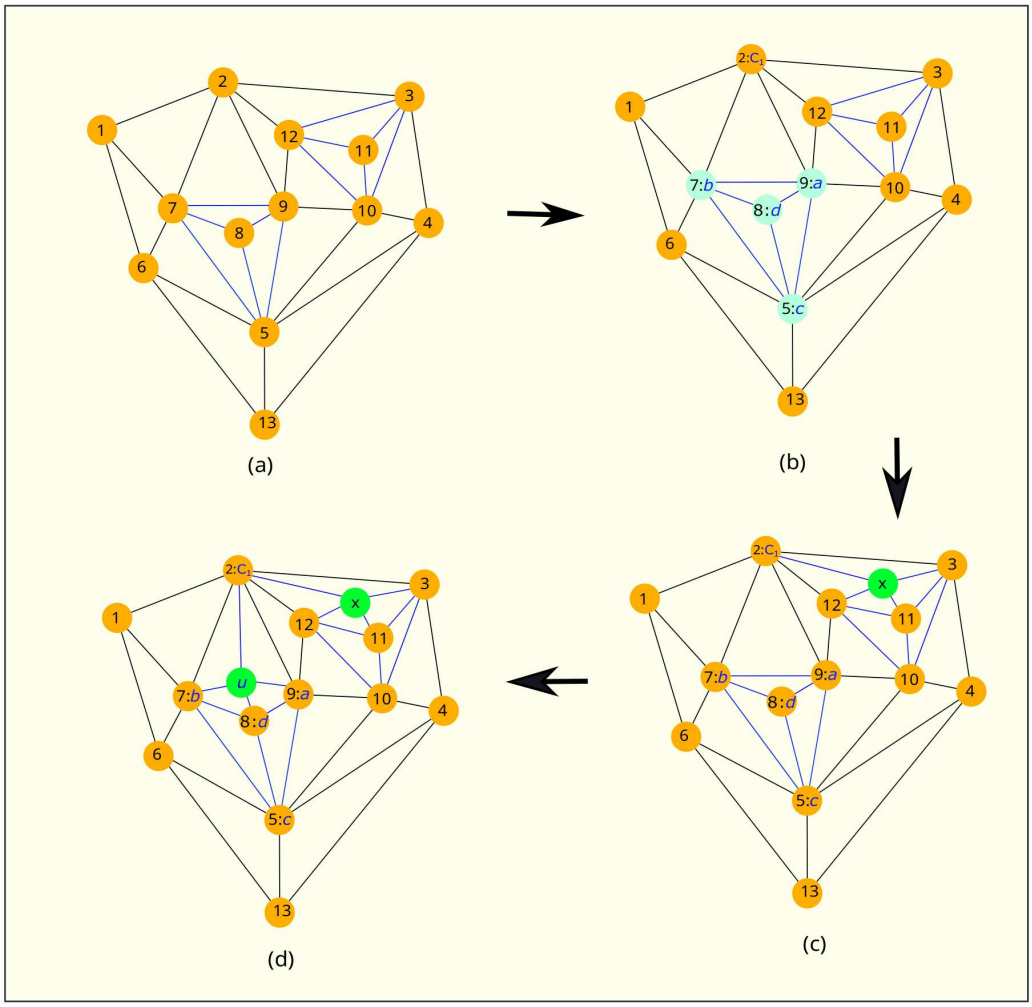


图13：（a-d）标记复杂三角形 K_L（即按逆时针顺序排列的 a、 b、 c和 d），并拆分 G_L^1中的复杂三角形。

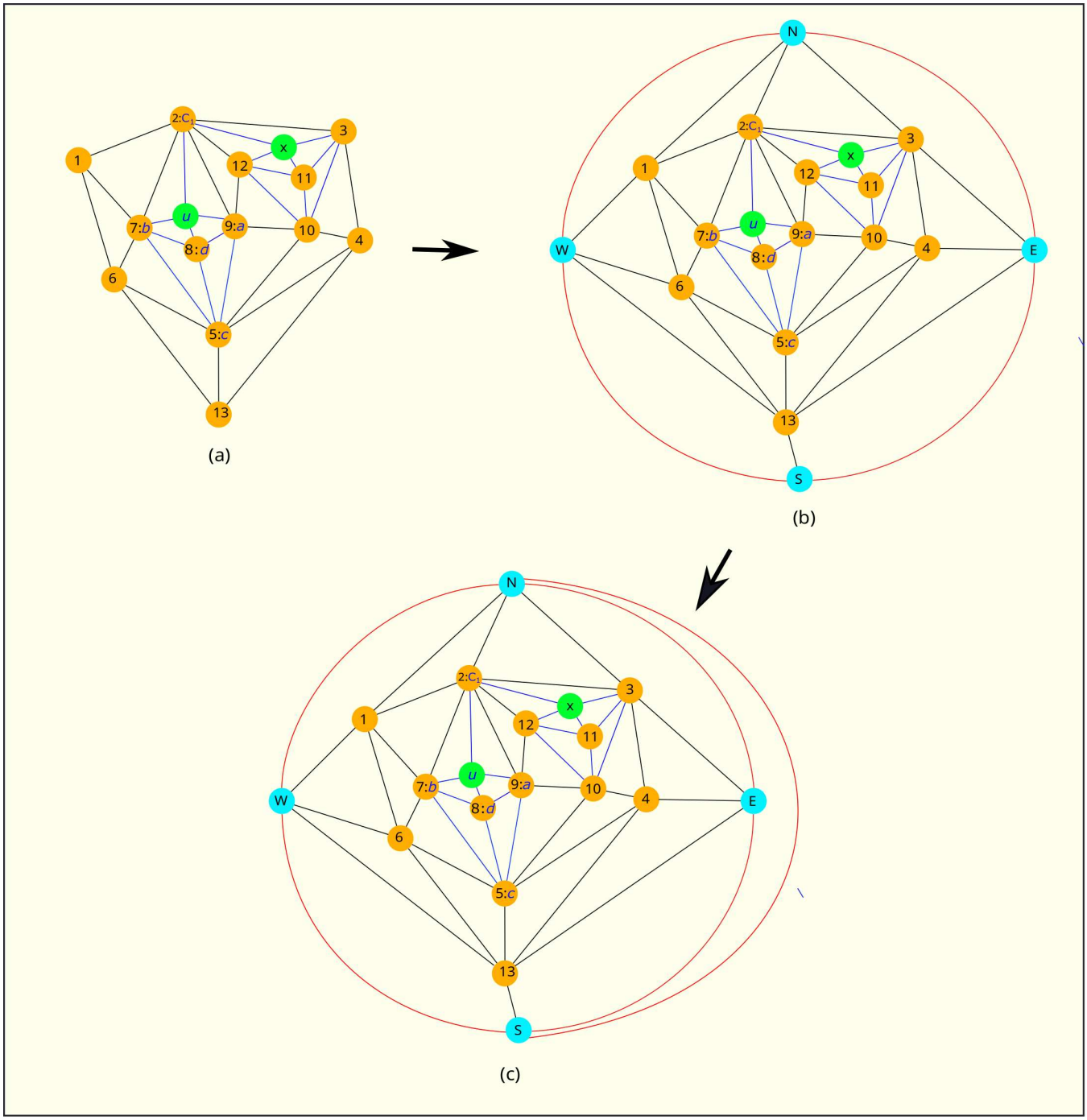


图14：(a-c) 对 G_L^1 应用四完备并添加额外的边，以构建4-连通三角剖分图。

为了在平面图中成功创建一个 L形模块，有必要在输入图 G_L的排序过程中，优先对修改后的 K_L 子图进行规范序排序。

因此，我们旨在开发一种算法，按照预定的优先级序列，为输入图 G_L 的顶点和修改后的 K_L 子图分配规范序（参见图17、图18、图19：其中给出了用于生成 L形模块的10种可能的规范序），从而确保生成的平面图包含一个 L形模块。相应地， $Algorithm\ 1$ 为任何包含至少一个内部复杂三角形 K_L的PTGG_L 构造一个 L形模块，其中该模块位于平面图 F_L 内。

5.2 图 G_L中内部复杂三角形 K_L 的要求

复杂三角形在平面图中的结构作用，是决定后续平面图特征的关键因素。在分析不含复杂三角形且最多允许四条角点指示路径的三角剖分平面图时，可以得出这样的结论：此类平面图能够表示为矩形平面图。相反，引入复杂三角形会带来几何复杂性，使得无法构建完全由矩形组成的平面图。这一现象是由于复杂三角形确保了矩形平面图 [2]中存在非矩形模块。

为了设计一个包含 L形模块的平面图（F_L），必须包含一个位于 G_L^1 内部的子图 K_L （见图 10）。因此，输入图中必须存在子图 K_L，才能在平面图内成功构建形状为 L 的模块。这一结果揭示了在平面图中构建 L形模块的基本要求。因此，首先要在输入图 G_L中识别一个区域 K_L，其顶点按逆时针顺序排列为 a、 b、 c 和 d 。

要将模块 L 整合到平面图中，可根据顶点 u 的度，将额外模块 u 与模块 a合并为两种类别（见图 11）。这些配置涉及 u 与 a共享水平墙或垂直墙。每种类别都有八种将模块 u 与模块 a合并的方式；本研究仅针对类别1，在平面图中生成 L形模块。

基于类别1中的类型，我们推导出十种类别的 L 形模块，它们可以出现在最终平面图中。对于这十种类别（1–10）中的每一种，都可以在平面图中构造一个 L 形模块（见图17、18、19）。然而，所提出的算法1必须根据优先级为每种类别定义规范序。总共有六种不同的情况，其中一些情况具有相同的规范序，但正则边标记（REL）不同，后文将对此进行说明。

本研究着重于在平面图中构建一个 L 形模块，关注两种可能情况中的一种（见图 11）。正确性部分正式证明，对于给定图中的类别1，十种类别中的任意一种都存在这样的 L 形模块。此外，对于每种类别，模块u 的度必须为4，以确保其包含在最终平面图中（见图 11）。这一要求表明，图 G_L 的修改后的子图 K_L 中的顶点 u 必须作为内部顶点放置。因此，我们仍然着重于根据定义的十种类别构造一个 L 形模块，同时考虑 K_L的内部构型。

5.3 L形模块在平面图 F_L中的生成

本节通过一个示例说明我们提出的算法1：在内部子图为 K_L的输入图 G_L 中，于平面图 F_L 内生成一个 L形模块。

1. 步骤 [1t o 4] 算法1：复杂三角形识别与移除（除 K_L外） :
Roy 等人 [24] 描述的方法提供了一种识别并移除图中复杂三角形的方式。在该方法中，识别出复杂三角形后，会为每个复杂三角形选取一条边，并通过引入一个新顶点将其细分。由此，复杂三角形被消除并转化为一个4-圈。原始复杂三角形内部的顶点，具体而言，内部节点，现在构成一个矩形子平面图（如图10b所示，其中模块 d 展示了该子平面图）。四个相邻节点则转化为矩形模块，环绕并包含该子平面图。细分过程中引入的顶点对应一个模块，最终将与原始复杂三角形的某个相邻节点合并。要消除图中的所有复杂三角形，首先必须识别一个边的子集（记为 S_L），使每个复杂三角形至少包含该子集中的一条边。然后，在 S_L中的每条边上插入新顶点，从而将这些边细分。这种系统性的修改确保了

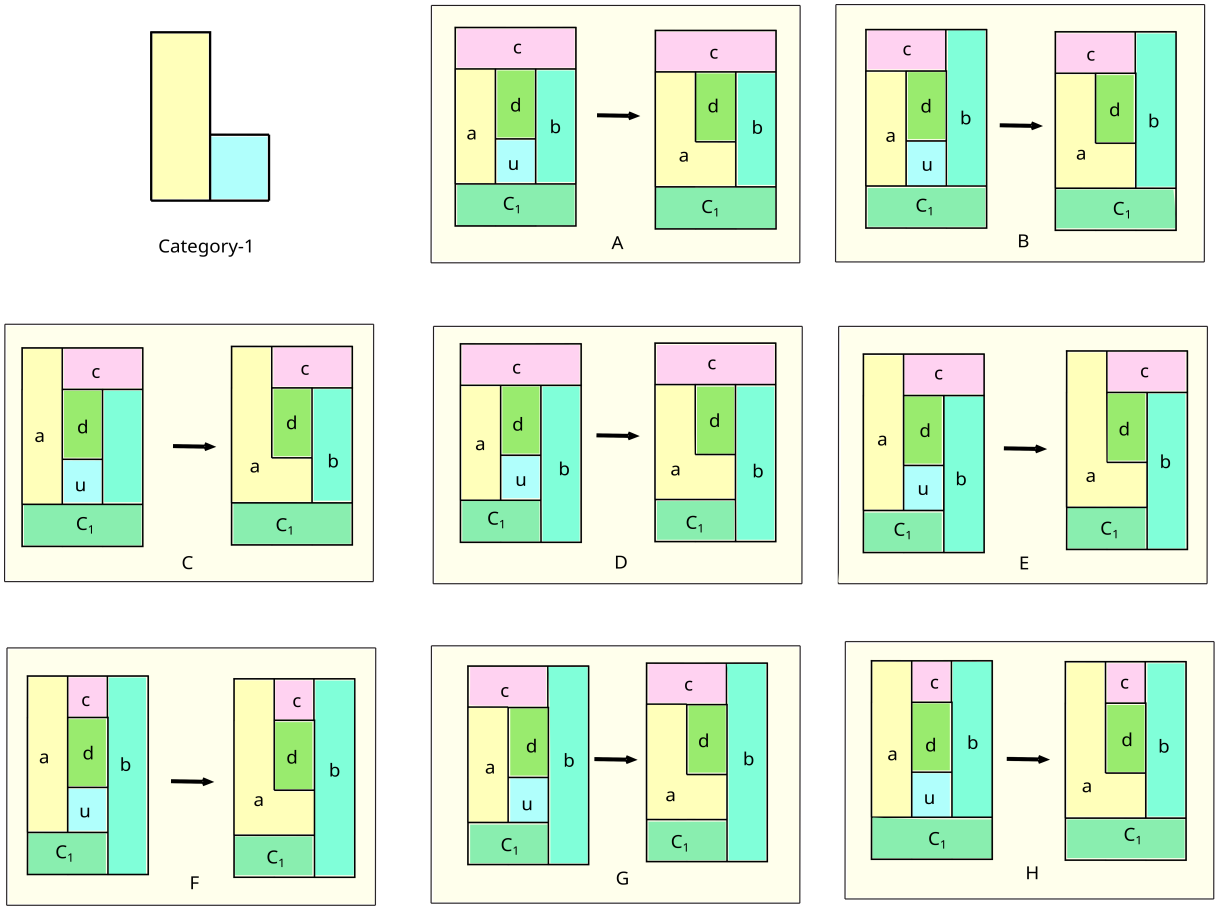


图15：平面图中与类别1相关的八种可能的 L 形配置类型（A–E）（见图17）。

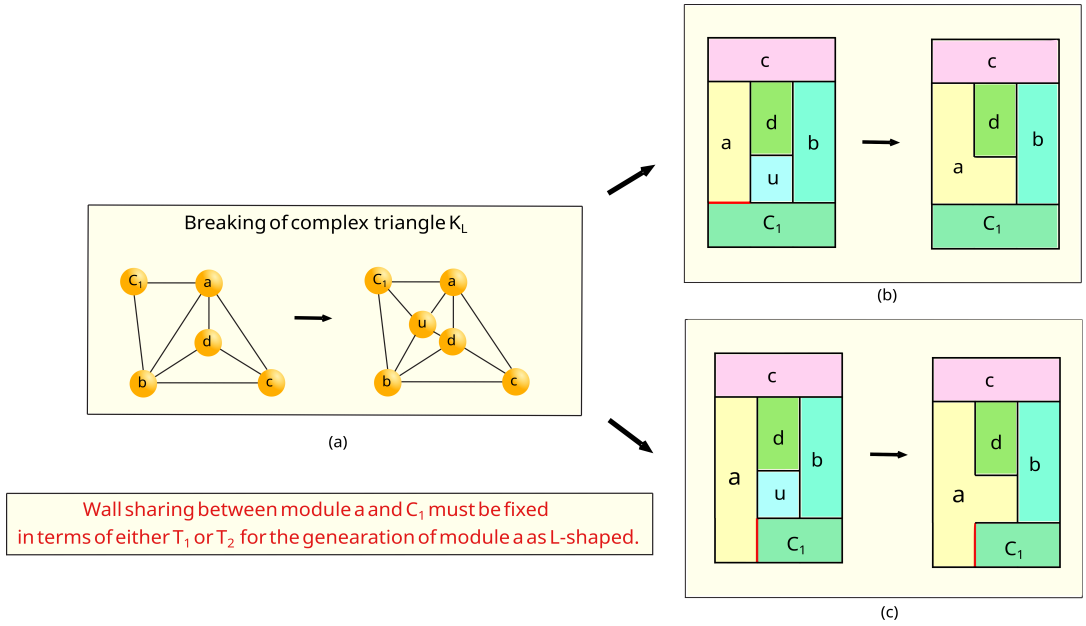
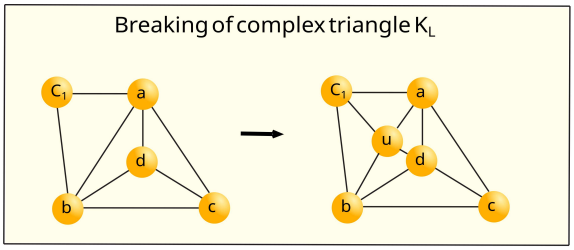
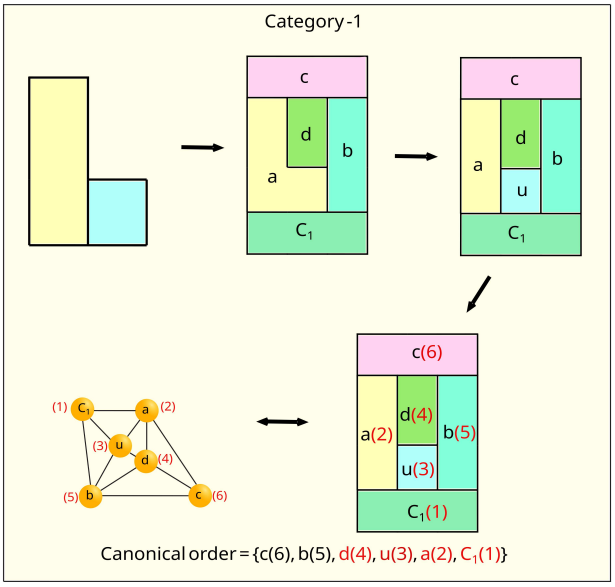


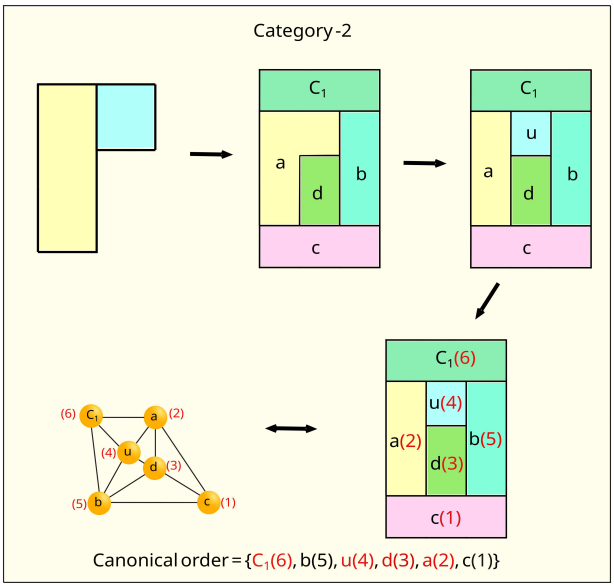
图16：（a）修改后的图 K_L. （b-c）模块 a 和 C_1 之间共用一面墙可能会导致最终平面图。



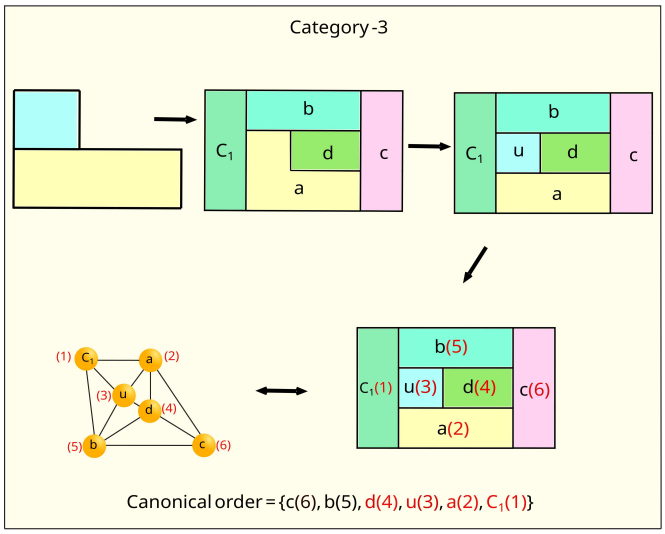
(a)



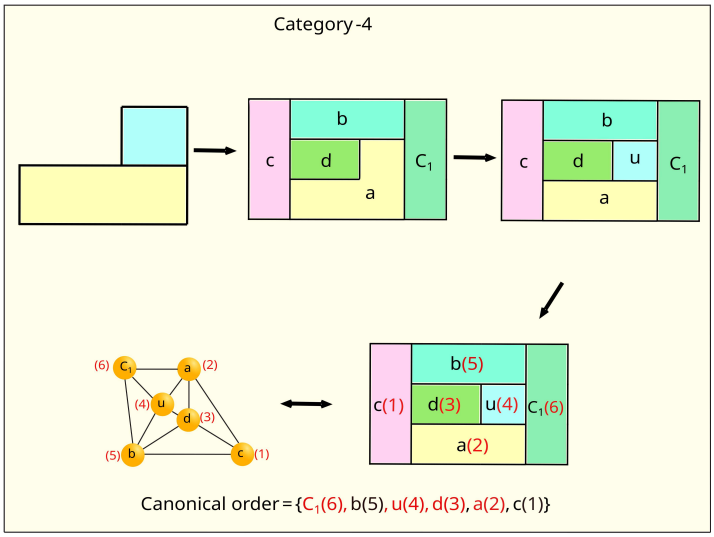
(b)



(c)

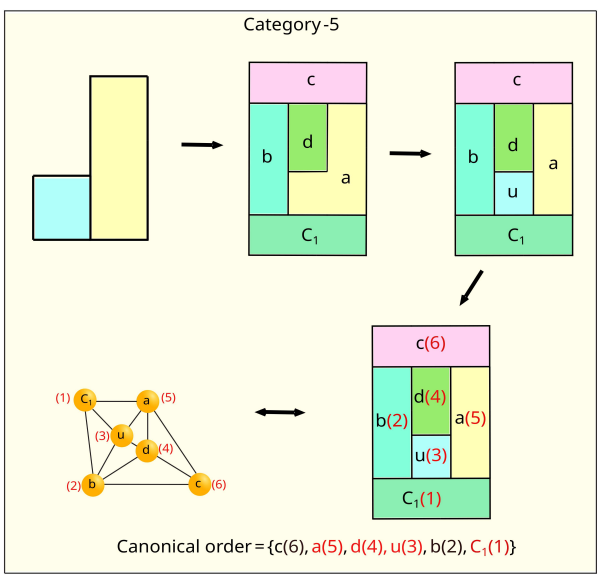


(d)

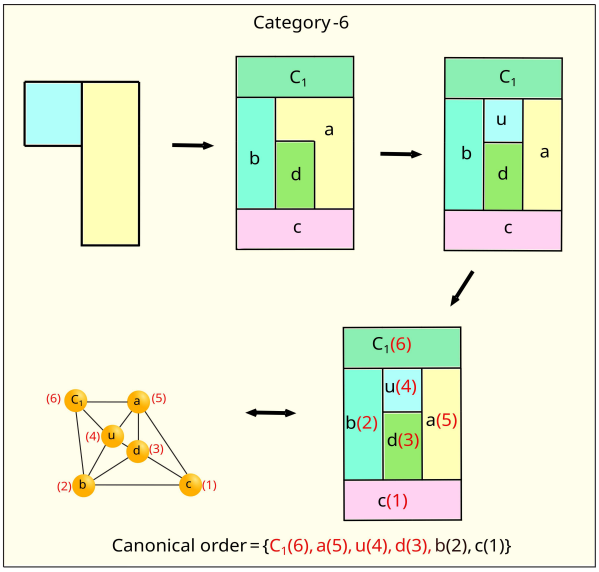


(e)

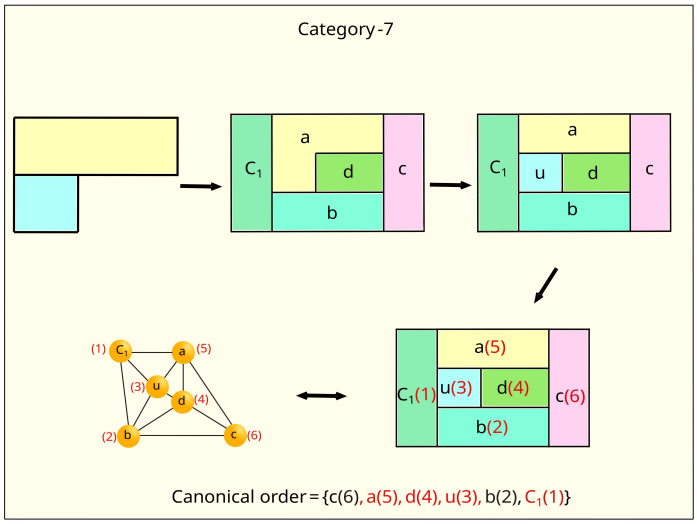
图17：展示了与 $modified$ 图 K_L 相关的规范序（a–e），对应不同类别（类别1–4）；各类别均根据相对于类别1的 L形模块定义。



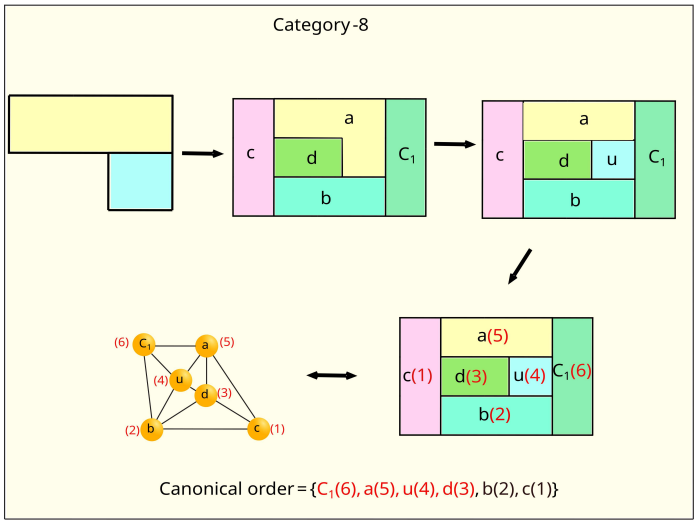
(f)



(g)



(h)



(di

图18: （f-i）展示了与 $modified$ 图 K_L 相关的规范序，分别对应不同类别（类别5–8）；这些类别均根据相对于类别1的 L形模块定义。

算法1：给定一个带有内部子图 K_L 的 G_L （三角剖分图），可以在相应平面图中构造一个 L 形模块，构造于相应平面图中。

输入：一个三角剖分图 $G_L(V,E)$，其中至少有一个与 K_L 同构的内部子图（见图）。

输出：一个正交平面图 F_L，其中包含一个与内部 K_L 相关联的 L 形模块（见图）。

如果 $\{\exists s_i \in ST_L/\{K_L\}\}$ 则
    调用 $Complex\ Triangle\ Removal\,(G_L(V,E))$ 算法 [24]。
否则
    转到第 5 行。
$\{C_1\} \leftarrow \{nbd\ (a)\ \cap nbd\ (b)\ \} - \{c,d\}$，$V \leftarrow V + \{u\}$，$E \leftarrow E - \{(a,b)\}$，$E \leftarrow E + \{(u,C_1),(u,d),(u,a),(u,b)\}$，$E_1 = \{a,b,c,d,u,C_1\}$，$G_L^1(V^1,E^1) = G_L(V,E)$. ▷ $nbd(x)$ = vertices adjacent to x
调用 $Four\text{-}Completion\,(G_L^1(V,E))$ 算法 [5]，$G_L^1(V^1,E^1) = E \leftarrow E + \{(N,S)\}$
$ch\ (v) = 0$，$vi\ (v) = 0$，$St\ (v) = F\ \forall v \in V^1$。
$W = v_1$，$S = v_2$，$St(W) = T$，$vi(N) = 2$ 和 $vi(E) = 1$，$St(S) = T$，$L = \emptyset$，$i = n$ （ G_L^1中的顶点数）。
函数 $CanonLabel(G_L^1(V^1,E^1)\ PLabel(L)\ r)$
    $i = r$。
    对 $r \leftarrow i$ 至 3 执行
        如果 $|L| > 0$，则
            如果 $\{\exists v \in L$，满足以下条件：$St(v) = F$，$ch(v) = 0$，$vi(v) \geq 2$，并满足优先级顺序$(L)\}$，则
                $v = v_r$；$St(v) = T$。
            else
                如果 $\{\exists v \in V^1 \setminus L$，满足以下条件：$St(v) = F$，$ch(v) = 0$，$vi(v) \geq 2\}$，则
                    $v = v_r$，$St(v) = T$。
                else
                    转到第 28 行（跳出 for 循环）。
        else
            如果 $\{\exists v \in V^1 \setminus E_1$ s.t. $St(v) = F, ch(v) = 0, vi(v) \geq 2\}$ 则
                $v = v_r, St(v) = T$。
            else
                转到第 28 行（跳出 for 循环）。
        令 $\{w_p, ..., w_q\}$ 为 v_r 的邻居（按其围绕 v_r 的顺序排列），其中 $St(w_s) = F$ 用于 $p \leq s \leq q$。
        将 $vi(w_s)$ 的值针对 $p \leq s \leq q$ 加 1。
        更新 $ch(v)$，以处理 $w_p, ..., w_q$ 及其邻点。
    返回 $(G_L^{cl}(V^1,E^1) = G_L^1(V^1,E^1),\ r)$。
函数 $Types\ of\ Priority\ label_L(G_L^{cl}(V^1,E^1),\ r)$
    $G_{L1}^{cl}(V,E) = G_L^{cl}(V^1,E^1),\ r^1 = r$。
    $(G^*, r^*) = CanonLabel(G_{L1}^{cl}(V,E)\ PLabel(\{C_1,u,d,a\})\ r^1)$.
    如果 $r^* == 3$，则 ▷ $r^* == 3$ implies each vertex of G_L^1 is canonical ordered.
        返回 $G^*(V,E)$，并转到第 53 行。
    else
        $(G^*, r^*) = CanonLabel(G_{L1}^{cl}(V,E)\ PLabel(\{d,u,a,C_1\})\ r^1)$.
        如果 $r^* == 3$，则
            返回 $G^*(V,E)$，并转到第 53 行。
        else
            $(G^*, r^*) = CanonLabel(G_{L1}^{cl}(V,E)\ PLabel(\{d,u,C_1,a\})\ r^1)$.
            如果 $r^* == 3$，则
                返回 $G^*(V,E)$ 并转到第 53 行。
            else
                $(G^*, r^*) = CanonLabel(G_{L1}^{cl}(V,E)\ PLabel(\{a,d,u,C_1\})\ r^1)$.
                如果 $r^* == 3$，则
                    返回 $G^*(V,E)$ 并转到第 53 行。
                else
                    $(G^*, r^*) = CanonLabel(G_{L1}^{cl}(V,E)\ PLabel(\{C_1,a,u,d\})\ r^1)$.
                    如果 $r^* == 3$，则
                        返回 $G^*(V,E)$ 并转到第 53 行。
                    else
                        $(G^*, r^*) = CanonLabel(G_{L1}^{cl}(V,E)\ PLabel(\{a,C_1,u,d\})\ r^1)$.
                        返回 $G^*(V,E)$ 并转到第 53 行。
$G_L^2(V^2,E^2) = G^*(V,E)$。
调用 $REL\ Formation\,(G_L^2(V^2,E^2))$：$Algorithm\ 2$。 ▷ return $G_L^3(V^3,E^3)$ m
调用 $Rectangular\ floorplan\,(G_L^3(V^3,E^3))$：算法 [24]。 ▷ return F_L'
函数 $Merge\ Rooms(G_L^2(V^2,E^2),\ F_L',\ Enodes_L,\ m)$
    对 $u_i \in Enodes_L$ 执行
        $M(x,a_i,F_L')$ 或 $M(x,b_i,F_L')$。 ▷ Since each $u_i = (a_i, b_i)$ of S_L.
    如果 $m == 1$，则
        $M(u,a,F_L')$
    else
        $M(u,b,F_L')$
    返回 $F_L = F_L'$

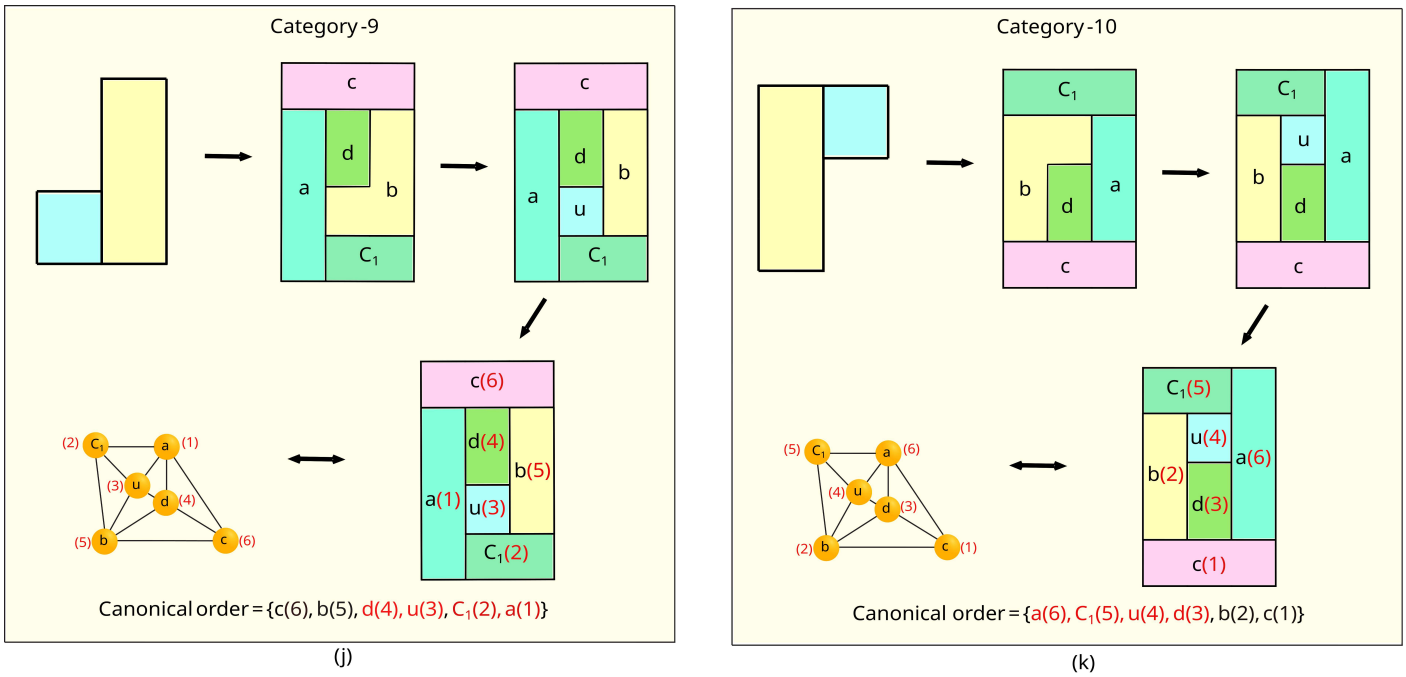


图19：(j-k) 展示了与 $modified$ 图 K_L 相关的规范序，适用于不同类别（类别9–10），每个类别均根据相对于类别1的 L形模块定义。

移除所有复杂三角形，同时保持图的三角剖分性。
因此，我们采用 [24] 中所述的复杂三角形识别与移除算法，首先识别 G_L 中的所有复杂三角形（见图12(a-b)），并从输入三角剖分图 G_L 中移除所有复杂三角形（如果存在），仅留下一个复杂三角形，记作 K_L （见图13(a-c)）。如果存在除 K_L 之外的任何复杂三角形，我们就通过添加一个新顶点 x 将其分割，并对输入图 G_L重新进行三角剖分。由此得到更新后的 G_L，其中除 K_L 外不含任何复杂三角形（见图13(a-c)）。

2. 步骤 [5至 6] 算法1：复杂三角形移除 K_L 与四向补全阶段 ：
为修改剩余的内部复杂三角形 K_L，我们选择边（a， b），并通过引入一个新顶点 u 将其消除（见图13d）。随后，我们插入新边（即 {（u， C_1）、（u， d）、（u， a）、（u， b）}），以保持图的三角剖分结构。所得图记作 G_L^1 （参见图13d）。经过此变换， G_L^1 不再包含任何复杂三角形。

完成复合三角形消除后，图 G_L^1 进入四向补全过程：Kant 和 He [5] 提出了一种方法，通过引入四个新顶点（分别代表东、南、西、北四个方向），从双连通 PTPG 中生成矩形平面图。该方法需要在图的外边界上识别四个角顶点，可通过应用 Bhasker 和 Sahni [9]提出的 CIP技术来确定这些顶点。矩形平面图通过选取 PTPG 的四条边界路径来构建。按照 [5], 所述的方法，首先在 G_L^1中识别四条路径 P_4、 P_3、 P_2、 P_1 ，然后将方向顶点（E、 W、 S、 N）插入各自对应的路径中。这些插入的顶点对应于定义平面图边界的矩形边界模块。如图14(a-b)所示，此过程构成四完备阶段。

完成四完备阶段后，将边（N, S）引入 G_L^1，从而得到一个4-连通三角剖分图（G_L^1），如图14c所示。随后，图 G_L^1 进入优先级规范序步骤。

3. 算法1的步骤 [7 至 53] ：优先级规范序：本节介绍算法1，该算法基于通过算法1的步骤7–53得到的4-连通三角剖分图 G^1，计算定义4（参见术语第2节）中定义的规范序图 G_L^2 。此优先规范序旨在支持在平面图 F_L中生成 L形模块，并以关联的PTGG_L为依据，其中至少包含一个内部复杂三角形 K_L。

算法2：从规范有序图构造正则边标记构造 $G_L^2(V^2, E^2)$.

符号说明:
在规范有序有向图 G中,
1. b_k：顶点 v_k 的 $basic\text{-}edge$ (b_k)定义为入边（v_b, v_k） $= b_k$ ，其中 $l < k$ 且取最小值（此处 $i \leq l \leq j$）。
2. C_k： v_k 的集合 $C_k = \{v_i, ..., v_j\}$ 定义为来自 v_i、.....、 v_j 并属于 C_{k-1} 的入边（外部面 G_{k-1}），沿着从 v_1 到 v_2 的路径按此顺序排列。
3. R_k： $R_k = \{v_m, ..., v_n\}$ 的 v_k 集合定义为从 v_m、.....、 v_n 在（$G - G_{k-1}$）中的出边，按逆时针方向环绕 v_k。
4. lp_k： $left\ point$ 的 v_k、 rp_k： $right\ point$ 的 v_k、 le_k： $left\ edge$ 的 v_k、 re_k： $right\ edge$的 v_k。

输入：G_L^2（V^2， E^2）。
输出： G_L^3（V^3， E^3）（正则边标记）， m。

1: $E^2 \leftarrow E^2 - \{$（N， S）$\}$、 $T_1 = \emptyset$、$T_2 = \emptyset$。
2: 对于每条边 $\{v_i, v_j)\} \in G_L^2(V^2, E^2)$: 直接 $v_i \to v_j$ 对于 i ＜j（除 (W, S)、(S, E)、(E, N)、(N, W) 外）。
3: 对于 $k = n - 2$ 至 3 执行 ▷ *see Figure 27*
4: 计算 b_k 的 v_k。
5: 在 $lp_k = v_i$ 和 $rp_k = v_j$的条件下，计算 $C_k = \{v_i, ..., v_j\}$ 的 v_k。
6: 在 $(v_i, v_m) = le_k$ 和 $(v_i, v_n) = re_k$的条件下，计算 $R_k = \{v_m, ..., v_n\}$ 的 v_k。
7: 对于 $k = 3$ 至 $n - 2$ 执行
8: le_k 的 $v_k \in T_1$。
9: re_k 的 $v_k \in T_2$。
10: $b_k =$ (v_b, v_k) 的顶点 $v_k \in T_2$ ，如果 $v_l = lp_k$ 且 $\in T_1$ ，如果 $v_l = rp_k$，否则， $\in T_1$ 或 T_2。
11: (a, u) $\in T_{x1}$， (a, C_1) $\in T_{x2}$， (b, u) $\in T_{x3}$， (b, C_1) $\in T_{x4}$。 ▷ $T_{x1}, T_{x2}, T_{x3}, T_{x4} \in \{T_1, T_2\}$ *i.e., x1, x2, x3, x4* $\in \{1, 2\}$
12: 如果 $x1 \neq x2$ 则
13: $m = 1$。
14: 否则，如果 $x3 \neq x4$ 则
15: $m = 2$
16: 否则
17: $\{x\} \leftarrow \{nbd(a) \cap nbd(C_1)\} - \{u\}$
18: $(C_1, x) \in T_{x5}$, $(a, x) \in T_{x6}$
19: 如果 $x2 == x5$ 则
20: $(a, C_1) \in T_{x6}$, $m = 1$。
21: **else**
22: $(x, C_1) \in T_{x2}$, $(a, C_1) \in T_{x6}$, $m = 1$.
23: 返回 $G_L^3(V^3, E^3) = G_L^2(V^2, E^2)$, m。

图17、图18和图19展示了十种不同类别（类别1–10），用于描述修改后的图 K_L的规范序；该图对应于类别1中的一个 L形模块（参见图 11）。这些类别将作为参考框架，用于在后续步骤中构建图G_L^1 的规范序。此外，我们发现，在与子图 K_L相关的最终平面图中，将模块 a 构建为 L形模块时，模块 a 与模块 c之间，以及模块 b 与模块 c、模块 C_1之间的墙体共享（即水平共享（T_1）和垂直共享（T_2））不会影响模块 a 形成 L形。换言之，模块 a 可以与模块 c共享 T_1 和 T_2，模块 b 可以与模块 c 和模块 C_1共享 T_1 和 T_2，而不会妨碍模块 a 形成 L形（如图15类别1所示，其中展示了八种有效的 L形配置）。然而，模块 a 与模块 C_1之间共享的墙必须固定。如图16所示，如果这面墙未固定，模块 a 可能会形成 T形配置，而非预期的 L形，从而导致布局不一致。因此，无需为顶点 b和 c指定固定的规范序。在顶点集合 $\{a, b, c, d, u, C_1\}$中，只有子集 $\{a, d, u, C_1\}$ 需要在图 G_L^1的规范排序过程中根据优先级指定顺序。

参见图 17、18、19：在使用类别1生成的十种类别（1–10）中（如图 11 所示，模块 u 与 a 合并），只有六种会为顶点 $\{a, u, d, C_1\}$ 产生唯一的规范序。在某些情况下，尽管正则边标记不同，规范序仍然相同。因此，这十种类别按规范序归为六个更大的类别，依次为 A 至 F：

1. 类别A：$\{C_1, u, d, a\}$–类别2的规范序与类别4相同。2. 类别B：$\{d, u, a, C_1\}$–类别1的规范序与类别3相同。3. 类别C：$\{d, u, C_1, a\}$–对应类别9。4. 类别D：$\{a, d, u, C_1\}$–类别5的规范序与类别7相同。5. 类别E：$\{C_1, a, u, d\}$–类别6的规范序与类别8相同。6. 类别F：$\{a, C_1, u, d\}$–对应类别10。

算法1旨在根据 A–F 类所规定的预定义优先级序列，为顶点（即 a、d、u、C_1）分配顺序。随后，通过执行算法1的第7步至第53步，构造规范有序图 G_L^2，具体如下所述。

（A）算法1的第7步）：对于 G_L^1 中的每个顶点 v，我们维护以下三个变量：
1. St (v)：一个用于指示 v 是否已排序的状态标志：(T) true 表示是，(F) false 表示否。
2. vi(v)：满足 St(u) $=$ T 的相邻顶点 u 的数量，即已完成排序的邻居。3. ch(v)：在由顶点集 V^1 构成的 G_L^1 子图中，与 v 相连的弦的数量；该顶点集不包括任何满足 St(u) $=$ T 的顶点 u。

使用算法1，我们通过检查顶点的邻居并评估预定义条件，迭代更新顶点属性。顶点按照特定顺序排列，每一步都会更新相关变量，并确定规范序中的下一个候选顶点。

图20-26 展示　　了 G 中顶点规范序排序过程的详细分解　　^1_L.

（B）算法1的步骤 [8 至 28]：开始时，在输入图 G_L^1 中，外边界顶点 N、W 和 E 分别按 v_{19}、v_1 和 v_2 的顺序进行规范排序。$G_{L_{18}}^1$ 的外面对应于圈 C_{18}^L（参见图20(ii)）。该过程首先识别圈 C_{18}^L 上顶点 v_{19} 尚未排序的邻居，即顶点 E、3、2 和 1。对于这些顶点，我们更新两个属性：ch(v) 和 vi(v)。更新后，顶点 E 满足规范排序所需的条件：ch(E) $= 0$ 且 vi(E) $= 2$。因此，E 按规范形式排序为 v_{18}，并相应调整其相邻顶点的相关属性（参见图20(iii)）。

接下来，我们对 G_L^1 中剩余的未标记顶点进行逐步规范排序（不包括集合 $\{a, b, c, d, u, C_1\}$ 中的顶点）。在每一步中，我们找出一个尚未排序的顶点 v，并且该顶点满足以下条件：$ch\,(v) = 0$（表示不存在内部弦），且 $vi\,(v) \geq 2$（即它至少有两个已排序的邻居）。如果有多个顶点满足这些条件，则选择其中一个，并将其规范序记为 v_r（即排序的第 r 步）。在对 v_r 排序后，我们会更新其邻居的属性，具体而言，更新当前边界圈 C_{k-1}^L 上弦的状态和访问状态。随后重复这一过程，寻找并排列下一个合适的顶点，v_{r-1}（参见图20至图22(ix)）。这一迭代排序持续进行，直到下一个可标记的顶点仅来自集合 $\{a, b, c, d, u, C_1\}$（见图22(x)）。此时，规范排序过程暂停，转入下一阶段（因为我们必须进入下一步，按照定义的优先级顺序排列剩余顶点）；所得的有序图记为 G_{L1}^{cl}（V^1，E^1）（见图22(x)）。

要点A：在算法1的第8至第28步中，主要目标是尽可能多地对 G_L^1 中的顶点进行规范排序，但不包括特定集合 $\{a, b, c, d, u, C_1\}$。然而，由于某些限制，例如计数值 (ch（v）) 为1，或某个顶点 (vi（v）) 小于1，集合外的某些顶点可能仍未排序，例如图22(ix)中的顶点1、6和13。因此，首要目标是对 G_L^1 中的大多数顶点进行规范排序，只留下指定集合 $\{a, b, c, d, u, C_1\}$ 中的顶点不处理。

要点 B：集合 $\{a, b, c, d, u, C_1\}$ 顶点的排序过程：对于一组 $\{a, b, c, d, u, C_1\}$ 顶点，通过调用算法1中的函数（$Types\ of\ Priority\ label_L$），采用一种不同的规范排序过程（见图22至图26）。这些顶点相对于六个类别（A-F类）中的任意一个进行规范排序。我们将尝试使用六个类别中的任意一个对集合 $\{a, d, u, C_1\}$ 顶点进行排序，同时逐一检查（因为无需为顶点 b 和 c指定固定的规范序：如上所述）。
假设可以根据任一已定义类别（A至F）对顶点 $\{a, d, u, C_1\}$ 进行有效排序。在这种情况下，我们将对其进行规范排序，并传递生成的图 G_L^2，以构造相应的正则边标记。

(C) 算法1的步骤 [29 至 53]： G_L^1 中的顶点排序（包括集合 a、b、c、d、u、C_1）：
给定先前获得的规范序 π，我们尝试按照六个预定义类别（A–E）依次对其子集 $\{a, d, u, C_1\}$ 进行排序（参见图22(xi)）。输入图不满足类别A所要求的结构条件：见图22 (xi)-(xiv)（不过，如果在另一个图中找到符合条件的匹配，则会进入正则边标记阶段）。针对类别B进行检验时也出现了类似情况（见图23(xv)-(xvii)）。随后，检验该图是否符合类别C，但结果同样不符合（见图23(xviii)-(xx)）。最后，按照类别D对其排序进行检验，结果满足所需条件（见图24至26）。随后将得到的规范序图指定为 G_L^2。

图20至26逐步展示了按照所提出的算法1中的步骤7至53，从初始输入图 G_L^1推导出规范有序图G_L^2 的过程。所得图现在已准备好进入后续的正则边标记阶段。

4. 算法1的步骤54：生成正则边标记：
为了生成 G_L^2 的矩形平面图，我们将应用算法2，根据其生成的规范序为G_L^2 构造正则边标记，方法论遵循 [5] 中所述。

算法2的步骤 [1-6] ：首先计算 G_L^2 的有向边（见图28(a-b)），然后生成基边列表 b_k，以及所有顶点对应的集合 C_k 和 R_k （见图 28c ）。

算法2的步骤 [7-10] ：利用基边 b_k 以及每个顶点对应的集合 C_k 和 R_k，为 G^2 构造由 T_1 和 T_2组成的正则边标记（REL）（见图28d）。

算法2的第 [11-23] 步：我们将检查模块a 与 C_1之间，以及模块 a 与 u之间共享的是哪种类型的墙，即 T_1 还是 T_2；并对模块 b 进行相同的检查（参见图28d）。为确保模块 a 形成 L形结构，它必须与 C_1 和 u共享不同类型的墙，也就是说，其中一个连接应属于 T_1，另一个应属于 T_2。有三种可能的情况需要考虑。我们将逐一评估每种情况，并确定 m 的值，该值表示模块 u 应与模块 a 还是模块 b合并。

情况1：如果模块 a 分别与 u 和 C_1共享类型相反的墙（T_1 和 T_2），则设置 $m = 1$，表示模块 u 应与模块 a 合并（参见图28d：对于输入，存在情况1）。
情况2：如果情况1不适用，但模块 b 与 u 和 C_1共享不同类型的墙，则设置 $m = 2$，表示模块 u 应与模块 b 合并（参见图19j）。

情况3：如果情况1和情况2均不满足，我们修改边（x， C_1）（情况A），或依次修改（x， C_1）和（a， C_1）（情况B）：通过翻转该边（T_1 或 T_2），其中 $x = \{\text{nbd}(a) \cap \text{nbd}(C_1)\} - \{u\}$。完成此调整后，设置 $m = 1$ （参见图44(e-h)）。

由此生成正则边标记 G_L^3，并返回 m 值（1 或 2）。

5. 算法1的步骤55：生成矩形平面图：
使用规范形式顺序生成正则边标记（G_L）后，我们便用该正则边标记为生成的图 G_L^2 构造矩形平面图（F_L'）。具体做法遵循 Bhasker 和 Sahni [9] 描述的方法（见图 29a）。

6. 算法1的步骤 [**56** 至 **63**] ：通过合并模块，在正交平面图中生成 L形模块：

$Merge\ Rooms$ 函数描述了将对应于移除复杂三角形时引入的额外顶点的矩形模块整合起来的方法。它需要四个输入：由图G_L^2导出的矩形平面图F_L'、规范有序图G_L^2 本身、额外节点集合 $Enodes_L$，以及参数m；该参数用于确定额外模块 u 应与模块 a 还是模块 b合并。最终得到的平面图中将生成一个 L形模块，其结构与 K_L一致。

参见图 29(b-c)（此处 S_L = (12,3)，且 $Enodes_L = x$：将模块 x 与3合并，并将模块 u 与 a合并）；我们利用 $function Merge\ Rooms$，从矩形平面图 F_L' 中得到带有 L形模块的平面图 F_L。

因此，给定一个带有内部子图 K_L 的 G_L （三角剖分图），我们可以使用所提出的算法1，在相应平面图 F_L 中构造一个 L 形模块。

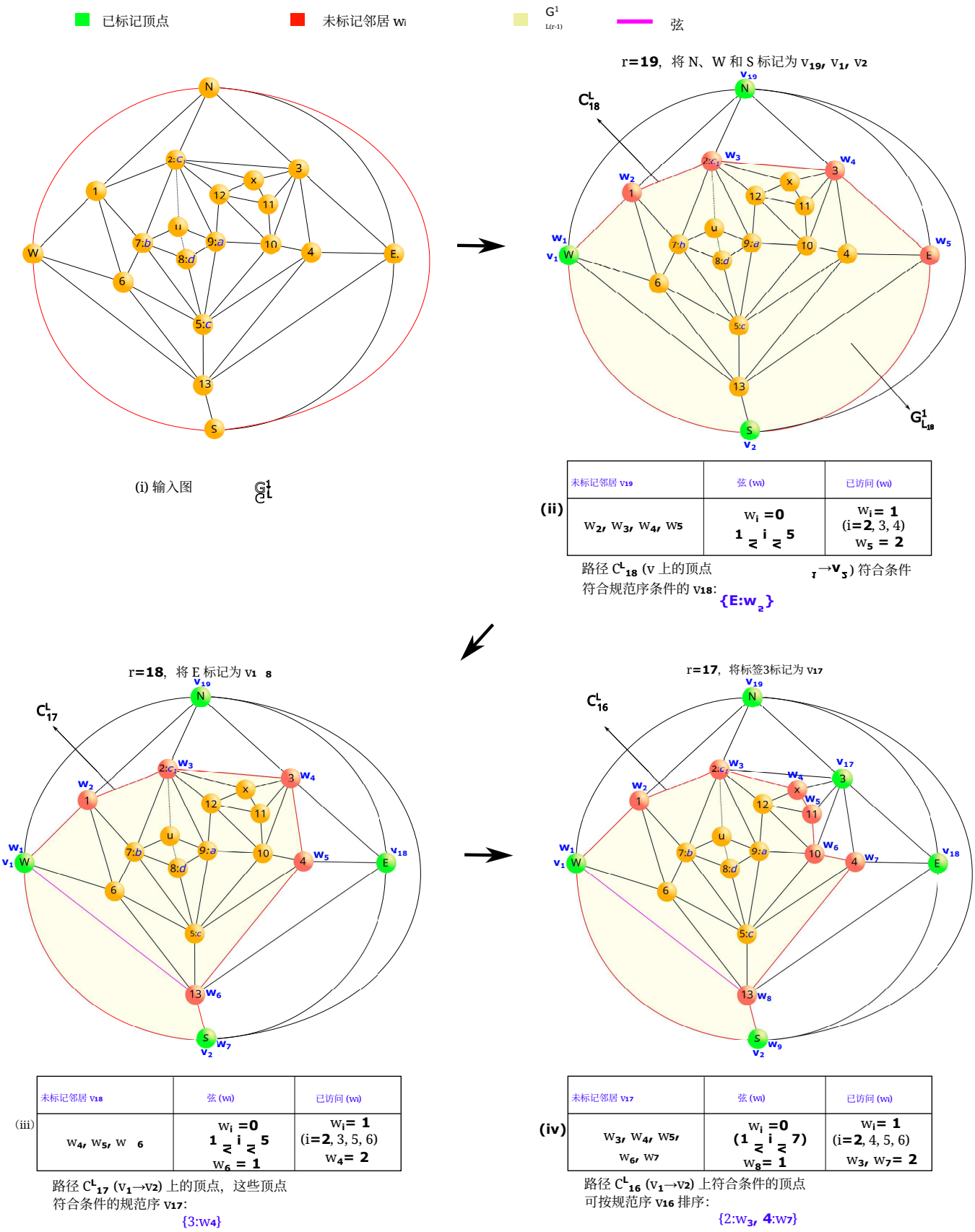
算法3：给定一个带有内部子图 K_T 的 G_T（三角剖分图），可以在相应平面图中构造一个 T 形模块，
构造于相应平面图中。

输入：一个带有内部子图 K_T 的 G_T （三角剖分图）（见图）。
输出：一个包含与内部 K_T 相关联的 T 形模块的正交平面图 F_T （见图）。

1：如果 $\{\exists s_i \in ST_T/\{K_T\}\}$ 则
2：　调用 $complex\ Triangle\ Removal(G_T(V,E))$ 算法 [24]。
3：否则
4：　转到第 5 行。
5：$E \leftarrow E-\{(a,c)\}$, $V \leftarrow V+\{u\}$, $E \leftarrow E+\{(u,e),(u,f),(u,a),(u,c)\}$, $G_T^1(V^1,E^1) = G_T(V,E)$。
6：调用 $Four\text{-}Completion(G_L^1(V,E))$ 算法 [5], $G_T^1(V^1,E^1) = E \leftarrow E+\{(N,S)\}$
7：$ch(v)=0$, $vi(v)=0$, $St(v)=F\ \forall v \in V^1$。
8：$W=v_1$、$S=v_2$、$St(W)=T$、$vi(N)=2$和 $vi(E)=1$、$St(S)=T$。
9：$t=n$ （G_T^1中的顶点数）。
10：函数 $CanonLabel_T(G_T^1(V^1, E^1)\ t)$
11：　对 $t \leftarrow i$ 到3执行
12：　　$\exists v \in V^1$ 使得 $St(v)=F$、$ch(v)=0$、$vi(v) \geq 2$ 则
13：　　$v=v_t$、$St(v)=T$。
14：　　令 $\{w_p,...,w_q\}$ 为 v_t 的邻居（按其围绕 v_t 的顺序排列），其中 $St(w_r)=F$ 对于 $p \leq r \leq q$。
15：　　将 $vi(w_r)$ 增加 1，对于 $p \leq r \leq q$。
16：　　更新 $ch(v)$，对于 w_p、...、w_q 及其邻居。
17：　返回 $G_T^2(V^2,E^2) = G_T^1(V^1,E^1)$
18：调用 $REL\ Formation(G_T^2(V^2,E^2))$：算法 [5]。　▷ $return\ G_T^3(V^3,E^3)$
19：调用 $Rectangular\ floorplan\ (G_L^3(V^3,E^3))$：算法 [5]。　▷ $return\ F_T'$
20：函数 $Merge\ Rooms(G_T^2(V^2,E^2),\ F_T',\ Enodes_T)$
21：　对 $u_i \in Enodes_T$ 执行
22：　　$M(x,a_i,F_T')$ 或 $M(x,b_i,F_T')$。　▷ $Since\ each\ u_i = (a_i,\ b_i)\ of\ S_T.$
23：　$M(u,a,F_T')$ 或 $M(u,c,F_T')$。
24：　返回 $F_T = F_T'$

5.4 我们提出的 $\boldsymbol{T}$ 形模块生成方法概述

通过细分边来改变 K_T 子图，从而构造出不同的矩形平面图（RFP），具体方法如 [5] 所述。一旦移除一个所引入的额外连通分量（即顶点），



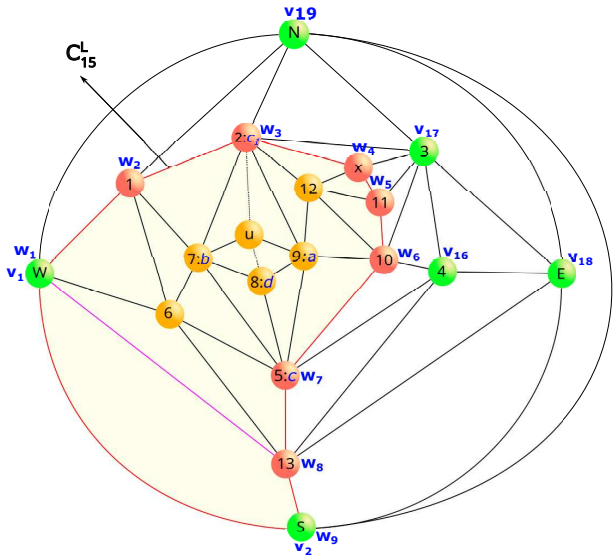
未标记邻居 v_{19}	弦 (w_i)	已访问 (w_i)
w_2, w_3, w_4, w_5	$w_i = 0$ $1 \le i \le 5$	$w_i = 1$ ($i = 2, 3, 4$) $w_5 = 2$

未标记邻居 v_{18}	弦 (w_i)	已访问 (w_i)
w_4, w_5, w_6	$w_i = 0$ $1 \le i \le 5$ $w_6 = 1$	$w_i = 1$ ($i = 2, 3, 5, 6$) $w_4 = 2$

未标记邻居 v_{17}	弦 (w_i)	已访问 (w_i)
w_3, w_4, w_5, w_6, w_7	$w_i = 0$ ($1 \le i \le 7$) $w_8 = 1$	$w_i = 1$ ($i = 2, 4, 5, 6$) $w_3, w_7 = 2$

图 20：（i-iv）输入图 G_L^1. 的逐步规范排序

r=16，将 4 标记为 v_{16}

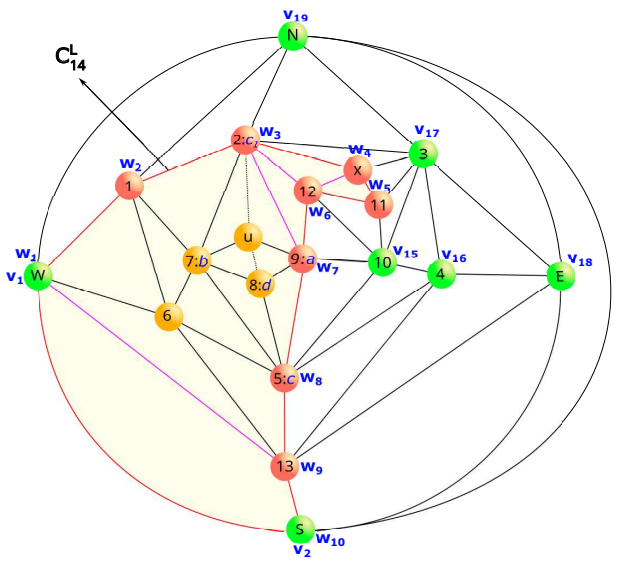


未标记邻居 v_{16}	弦 (w_i)	已访问 (w_i)
w_6, w_7, w_8	$w_i = 0$ $1 \le i \le 7$ $w_8 = 1$	$w_i = 1$ (i=2, 4, 5, 7) w_3, w_6, w_8=2

(v)

路径 C^L_{16} ($v_1 \to v_2$) 上的顶点
符合条件的规范序 v_{15}:
{2:w_3, 10:w_6}

r=15，将 10 标记为 v_{15}

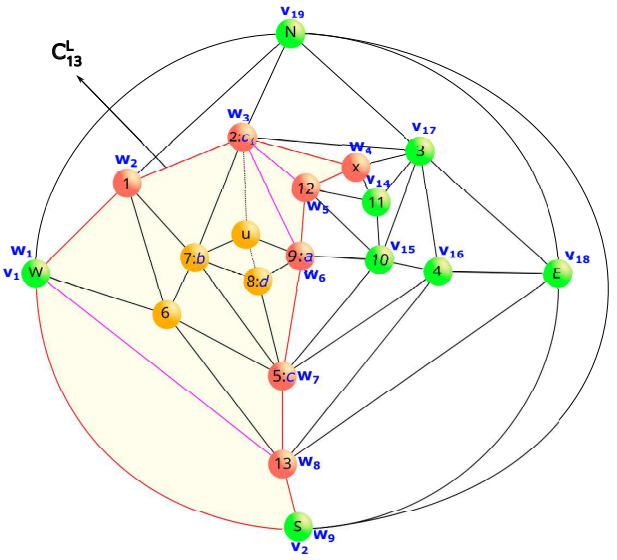


未标记邻居 v_{15}	弦 (w_i)	已访问 (w_i)
w_5, w_6, w_7, w_8	$w_i = 0$ (i= 2, 5, 8) (i=3, 5, 8, 9) $w_i = 1$	$w_i = 1$ (i=2, 4, 6, 7) (i= 3, 4, 5, 9)

(vi)

路径 C^L_{15} ($v_1 \to v_2$) 上符合条件的顶点
可按规范序 v_{14} 排列:
{5:w_7, 11:w_5}

r=14，将标签 11 记为 v_{14}

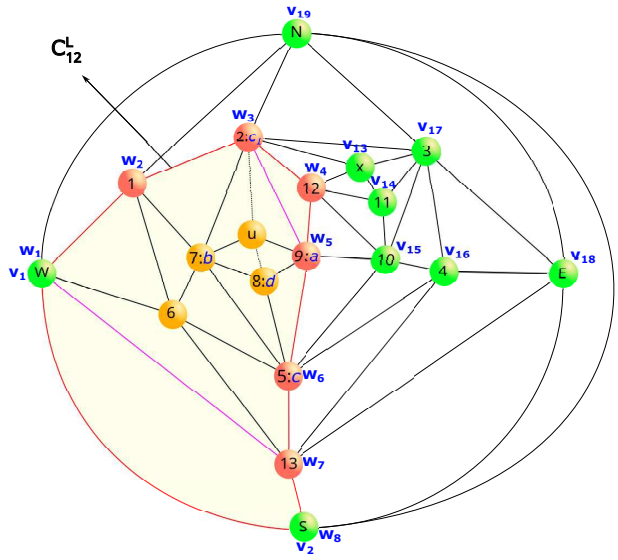


未标记邻居 v_{14}	弦 (w_i)	已访问 (w_i)
w_4, w_5	$w_i = 0$ (i = 2, 4, 7) (i=3, 4, 5, 7, 8) w_3, w_5, w_6, w_8= 1	$w_i = 1$ (i=2, 6) $w_i = 2$

(vii)

路径 C^L_{13} ($v_1 \to v_2$) 上的顶点
符合规范序 v_{13} 的条件:
{x:w_4, 5:w_7}

r=13，将 x 标记为 v_{13}



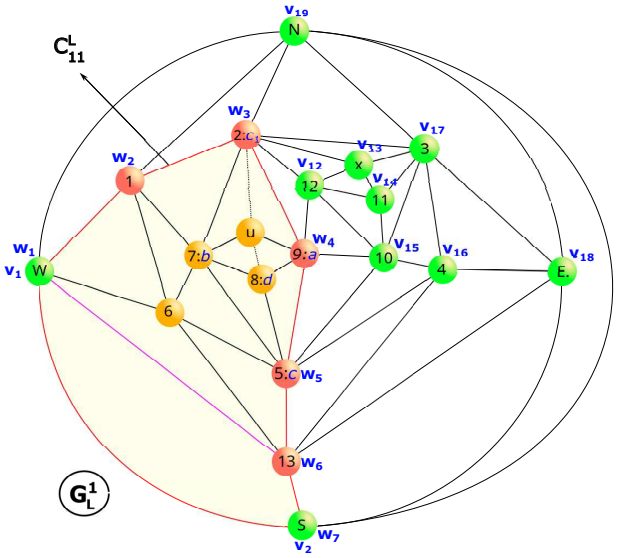
未标记邻居 v_{13}	弦（w_i）	已访问（w_i）
w_3, w_4	$w_i = 0$ (i= 1, 2, 4, 6) w_3, w_5, w_7 =1	w_2, w_5= 1 w_6, w_7= 2 w_3, w_4= 3

(viii)

路径 C^L_{12} ($v_1 \to v_2$) 上的顶点
符合规范序 v_{12} 的:
{12:w_4, 5:w_6}

图 21：（v-viii）输入图 G_L^1. 的逐步规范排序

r=12，将标签 12 设为 v_{12}



(ix)

未标记邻居 v_{12}	弦 (w_i)	已访问 (w_i)
w_3, w_4	$w_i = 0$ ($2 \le i \le 5$) w_1, $w_6 = 1$	$w_2 = 1$ w_4, w_5, $w_6 = 2$ $w_3 = 4$

路径 C^L_{11} ($v_1 \to v_2$) 上符合规范序 v_{11} 的条件：

{2:w_3, 5:w_5, 9:w_4}

(x)

现在，下一个可能的 (C_1, a, c) 顶点中选出下一个可能被标记的

集合 E_1 = {a, b, c, d, u, C_1} 中的 (C_1 = a, c) 顶点。

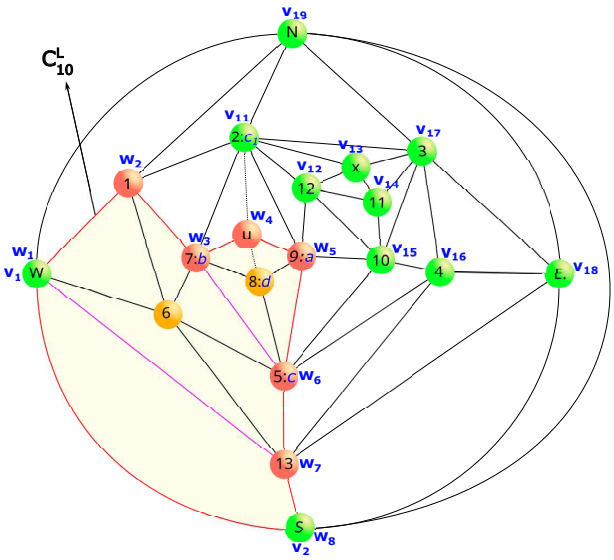
$G^{cl} = {}_{\Gamma_L} G^1{}_{,L}$ = 11

(xi)

$G^{cl}_{L1} = G^{cl}_L$ $r^1 = r$ = 11

Checking the existence of ordering with respect to category A: {C_1, u, d, a}.

r=11，标签 2：c_1 as v_{11}



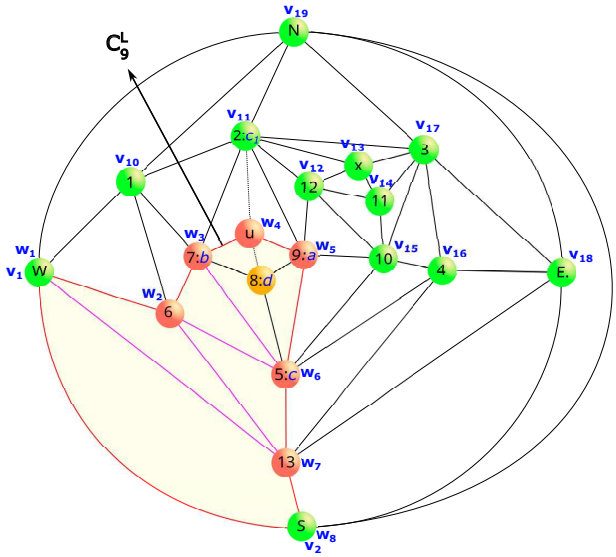
(xii)

未标记邻居 v_{11}	弦 (w_i)	已访问 (w_i)
w_3, w_4, w_5	w_2, w_4, $w_5 = 1$ w_3, w_6, $w_7 = 1$	w_3, $w_4 = 1$ w_2, w_6, $w_7 = 2$ $w_5 = 3$

路径 C^L_{10} ($v_1 \to v_2$) 上符合条件的顶点是规范序 v_{11}：

{1:w_2, 9:w_5}

r=10，将标签 1 设为 v_{10}



(xiii)

未标记邻居 v_{11}	弦 (w_i)	已访问 (w_i)
w_2, w_3	w_4, $w_5 = 0$ $w_i = 1$ ($i = 1, 2, 3, 6, 7$)	w_2, $w_4 = 1$ w_3, w_6, $w_7 = 2$ $w_5 = 3$

路径上的顶点 C^L_9 ($v_1 \to v_2$)，它们符合规范序 v_{11} 的：

{9:w_5}

(xiv)

类别A标记失败，因为顶点 '9:a' 不再符合将 C_1 标记为优先级层级后 v_{11}。由于顶点 'a' 是唯一剩下的顶点，但无法根据规则进行标记类别A。

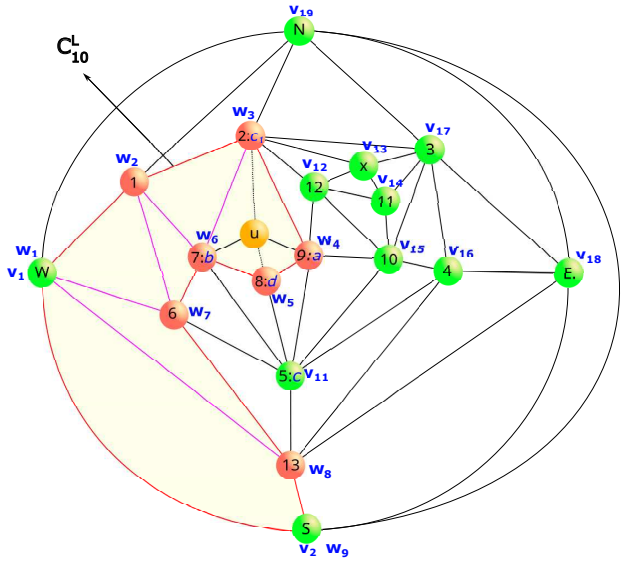
图 22：（ix-xiv）检查与类别A相关的规范序是否存在

(xv) ↳ $G^{cl}_{L1} = G^{cl}_{L}$, $r^1 = r = 11$

Checking the existence of ordering with respect to category B: {d, u, a, C_1}.

(xvii) **Since vertex d can not be labeled due to visited value being less than 2, therefore we can not assign vertices as in order associated with category B.**

r=11，将标签 5 标记为 v_{11}



未标记邻居 v_{11}	弦 (w_i)	已访问 (w_i)
w_4, w_5, w_6, w_7, w_8	w_4, w_5 =0 w_i =1 (i=1, 2, 3, 6, 7, 8)	w_2, w_5, w_6, w_7= 1 w_8= 2 w_4 = 3, w_3= 4

(xvi)

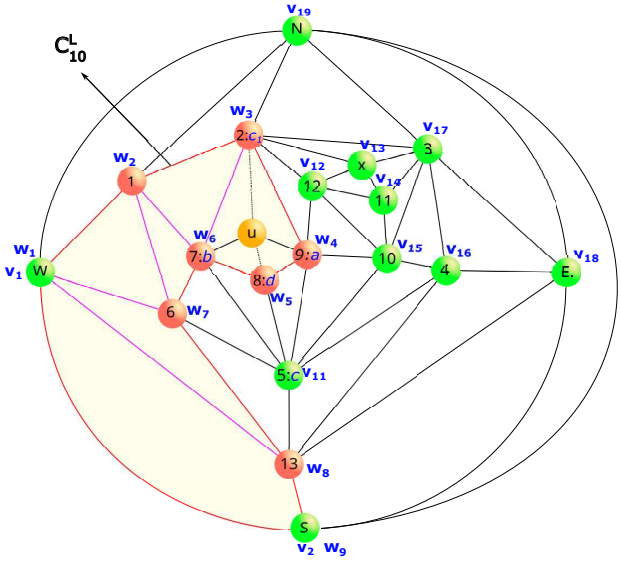
路径 C^L_{10} ($v_1 \rightarrow v_2$) 上的顶点是
符合条件的规范序 v_{10}:

{9:w_4}

(xviii) ↳ $G^{cl}_{L1} = G^{cl}_{L}$, $r^1 = r = 11$

Checking the existence of ordering with respect to category C: {d, u, C_1, a}.

r=11，将 5 标记为 v_{11}



未标记邻居 v_{11}	弦 (w_i)	已访问 (w_i)
w_4, w_5, w_6, w_7, w_8	w_4, w_5 =0 w_i =1 (i=1, 2, 3, 6, 7, 8)	w_2, w_5, w_6, w_7= 1 w_8= 2 w_4 = 3, w_3= 4

(xix)

路径 C^L_{10} ($v_1 \rightarrow v_2$) 上的顶点，它们
符合条件的规范序 v_{10}:

{9:w_4}

(xx) **Since vertex d can not be labeled due to visited value being less than 2, therefore we can not assign vertices as in order associated with category C.**

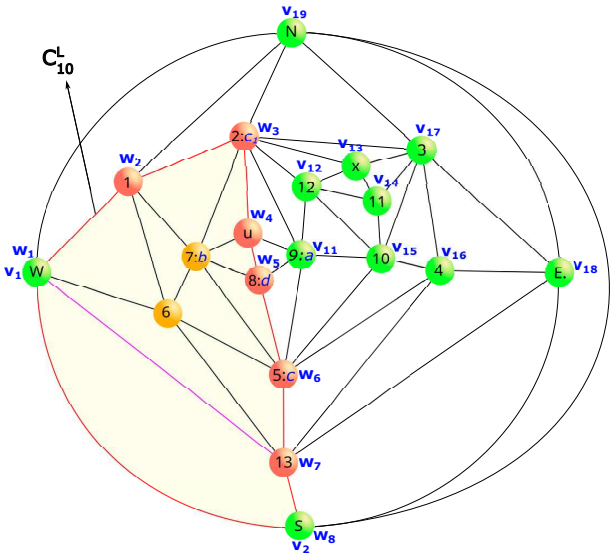
图 23：（xv-xvii）检查是否存在与类别B相关联的规范序。（xviii-xx）检查是否存在与类别C相关联的规范序。

(xxi)

$\hookrightarrow G^{cl}_{L1} = G^{cl}_{L}$, $r^1 = r = 11$

Checking the existence of ordering with respect to category D: {a, d, u, C_1}.

r=**11**, 标签 9:a 作为 v_{11}



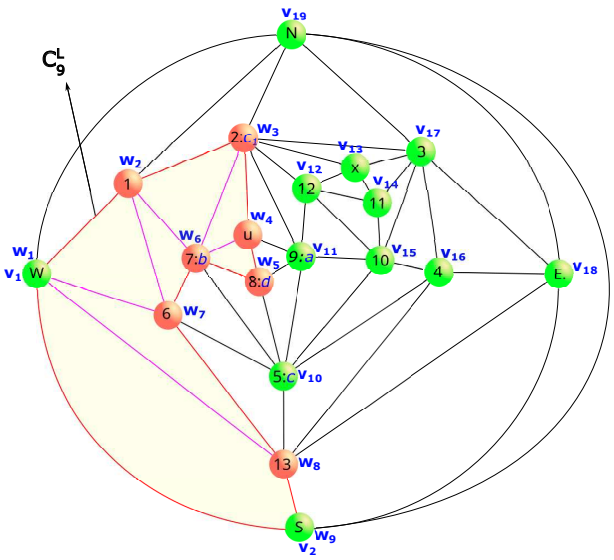
(xxii)

未标记邻居 v_{11}	弦 (w_i)	已访问 (w_i)
w_3, w_4, w_5, w_6	w_i **= 0** (**2 ≤ i ≤ 6**) w_7 **= 1**	**w_2, w_4,** w_5**= 1** w_6**= 3**, w_7**= 2** w_3**= 5**

路径 C^L_{10} ($v_1 \rightarrow v_2$) 上的顶点
符合规范序 v_{10} 的条件:

{C1:w_3, **5**:w_6}

r=**10**, 标签 5 作为 v_{10}



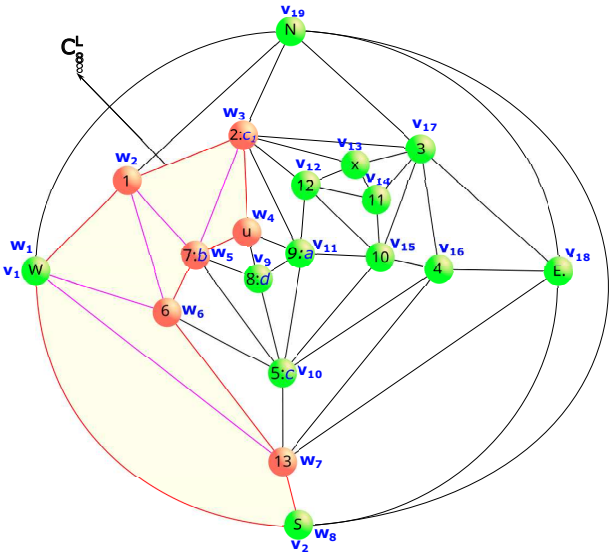
(xxiii)

未标记邻居 v_{10}	弦 (w_i)	已访问 (w_i)
w_5, w_6, w_7, w_8	w_5 **= 0** w_i **= 1** (**1 ≤i ≤8**, i≠ **5**)	w_i**= 1** (i= **2**, 4, 6, 7) w_5**= 2**, w_3**= 5**

路径 C^L_9 ($v_1 \rightarrow v_2$) 上的顶点，它们
符合规范序 v_9 的条件:

{C1:w_3, **8**:d}

r=**9**, 标签 8 作为 v_9



(xxiv)

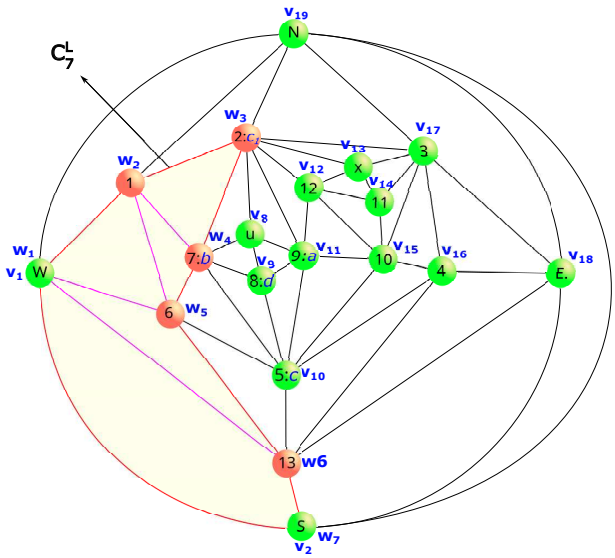
未标记邻居 v_9	弦 (w_i)	已访问 (w_i)
w_4, w_5	w_4**= 0** w_i **=1**, (i= **2**, 3, 5, 6, 7)	w_2, w_6**= 1** w_4, w_5 **= 2** w_7 **= 3** w_3 **= 5**

路径 C^L_8 ($v_1 \rightarrow v_2$) 上的顶点包括
符合规范序 v_8 的条件:

{u:w_4}

图 24: （xxi-xxiv）检查是否存在与类别D相关联的规范序。

r=8，将 u 标记为 v8



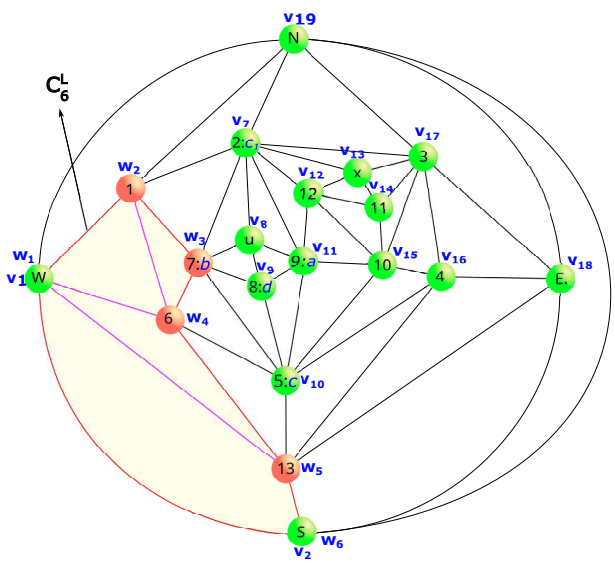
(xxv)

未标记邻居 v8	弦 (wi)	已访问 (wi)
w3, w4	w2, w4, w5, w6	w2, w5= 1 w4, w5 = 2 w3 = 6

路径 C^L_7 ($v_1 \to v_2$) 上符合条件的顶点
符合规范序 v7 的：

{2:w3}

r=7，将标签 2 设为 v7



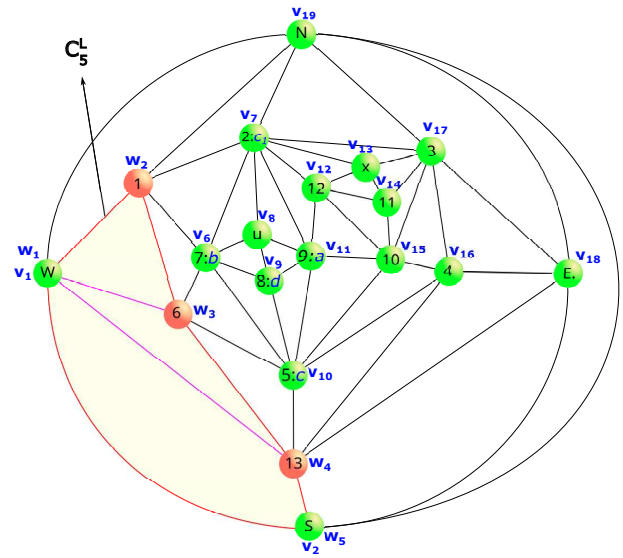
(xxvi)

未标记邻居 v7	弦 (wi)	已访问 (wi)
w2, w3	w2, w4, w5	w4 =1, w2 =2 w3, w5 = 3 w3 = 4

路径 C^L_6 ($v_1 \to v_2$) 上符合条件的顶点
可按规范序排列 v6：

{7:w3}

r=6，将标签 7 标记为 v6



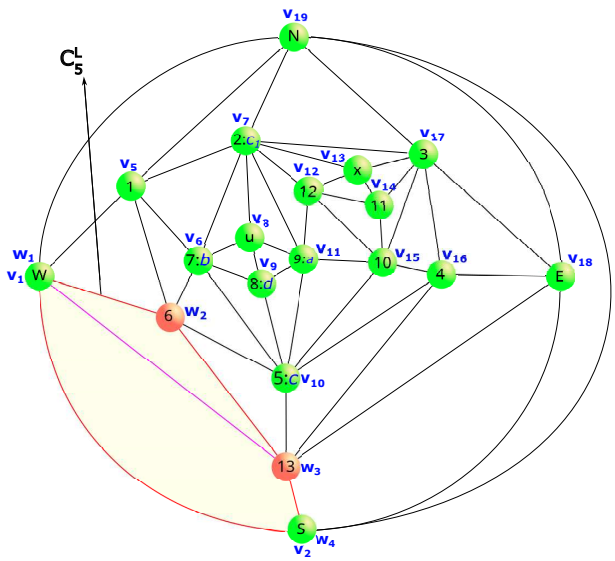
(xxvii)

未标记邻居 v6	弦 (wi)	已访问 (wi)
w2, w3	w3, w4	w2, w4 =3 w3 = 2

路径 C^L_5 ($v_1 \to v_2$) 上符合条件的顶点
可用于规范序 v5：

{1:w2}

r=5，将 1 标记为 v5



(xxviii)

未标记邻居 v5	弦 (wi)	已访问 (wi)
w2	w2= 0 w3= 1	w2, w3 =3

路径 C^L_4 ($v_1 \to v_2$) 上的顶点，它们
符合规范序 v4 的条件：

{6:w2}

图25：（xxv-xxviii）根据类别D为顶点按优先级排序。

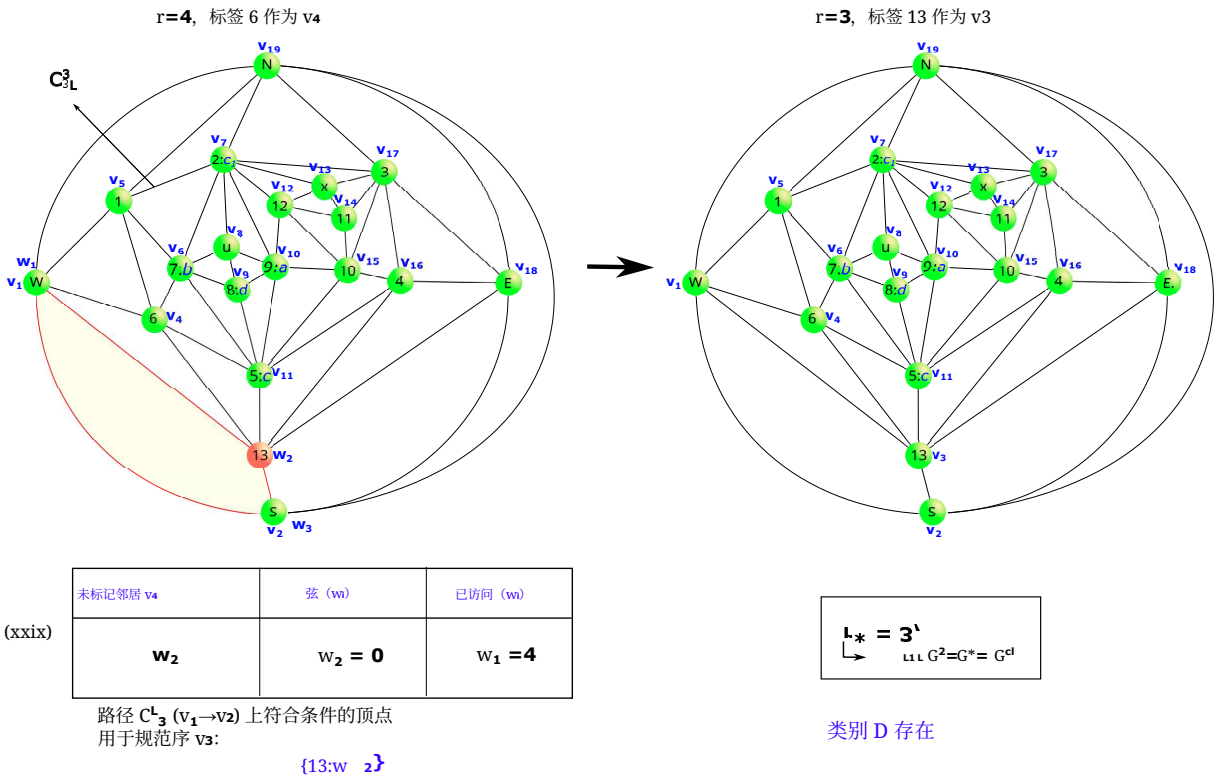


图26：存在类别D。因此得到一个规范有序图 G_L^2 。

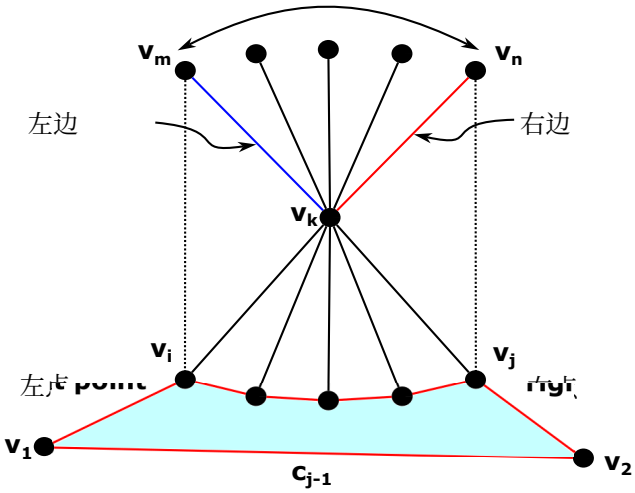
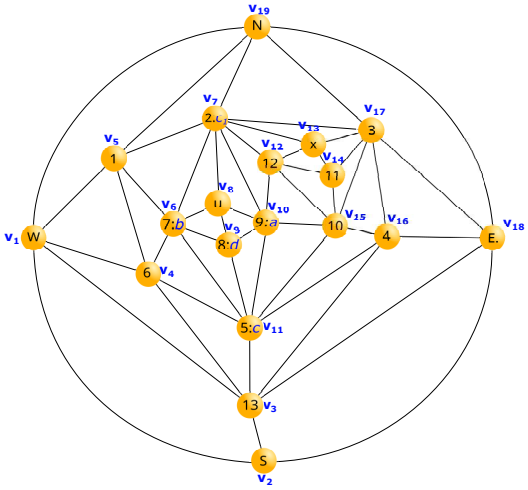
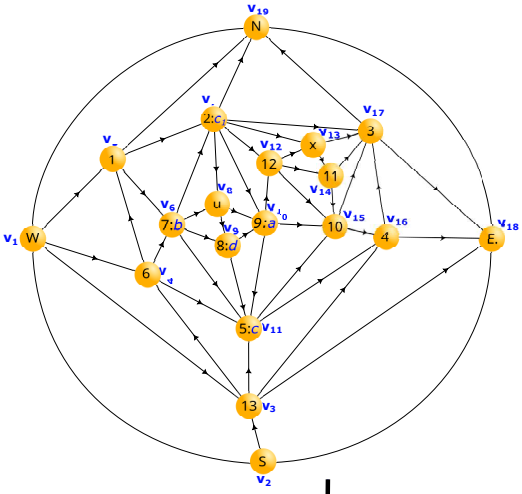


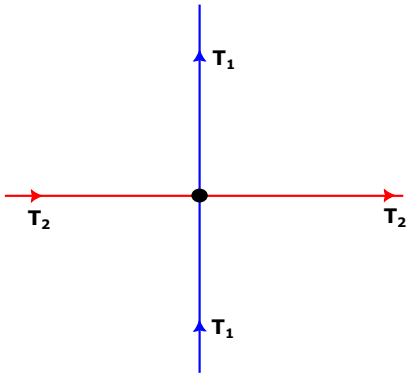
图 27：对于每个 j， G_{j-1} 的外部面边界及 v_j的相邻顶点。



(a)



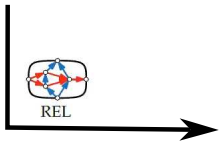
(b)

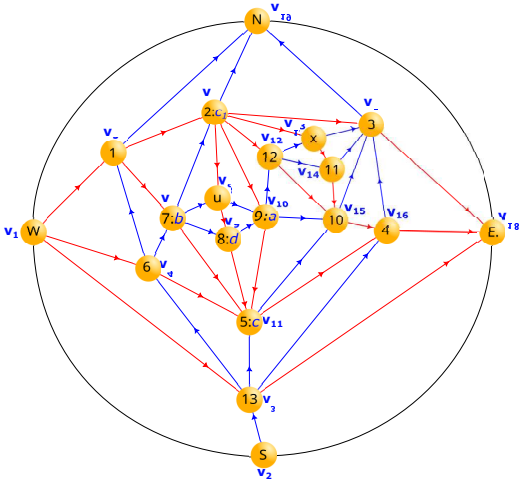


左点 右点	顶点 v^j v_i .... v_k		Left edge	右侧 edge
v_{13}	v_1, v_{13}, v_{14}, v_{12},	(v_1 v_{13})	(v_{13} v_{14})	(v_{13} v_{18})
v_{16}	v_{12}, v_{11}, v_3	(v_3 v_{16})	(v_{16} v_{13})	(v_{16} v_{18})
v_{12}	v_{14}, v_{12}, v_{10}, v_{11}	(v_{10} v_{12})	(v_{12} v_{13})	(v_{12} v_{16})
v_{14}	v_3, v_{12}	(v_{12} v_{14})	(v_{14} v_{13})	(v_{14} v_{12})
v_{13}	v_1, v_{12}	(v_1 v_{13})	(v_{13} v_{13})	(v_{13} v_{14})
v_{12}	v_1, v_{10}	(v_1 v_{12})	(v_{12} v_{13})	(v_{12} v_{12})
v_{11}	v_{10}, v_9, v_8, v_4	(v_4 v_{11})	(v_{11} v_{12})	(v_{11} v_{16})
v_{10}	v_1, v_8, v_9	(v_1 v_{10})	(v_{10} v_{12}) (v_{10} v_{11})	
v_9	v_8, v_6	(v_6 v_9)	(v_9 v_{10})	(v_9 v_8)
v_8	v_1, v_6	(v_6 v_8)	(v_8 v_{10})	(v_8 v_9)
v_1	v_2, v_6	(v_2 v_1)	(v_1 v_{13})	(v_1 v_8)
v_6	v_2, v_4	(v_4 v_6)	(v_6 v_1)	(v_6 v_{11})
v_2	v_1, v_4	(v_1 v_2)	(v_2 v_{13})	(v_2 v_6)
v_4	v_1, v_3	(v_1 v_4)	(v_4 v_2)	(v_4 v_{11})
v_3	v_1, v_2	(v_1 v_3)	(v_3 v_4)	(v_3 v_{16})

(c)

1. 对于每个顶点 v ^{j} $^{1\prime}$ ，
左边属于 T，右边属于 T_2。2. 对于每个顶点 v_i，边 b_k=(v_k,v_i) 在 v_k 为右点时属于 T_1，在 v_k 为左点时属于 T_2。





(d)

图28：规范有序图 G_L^2（V^2， E^2）的 REL构造。

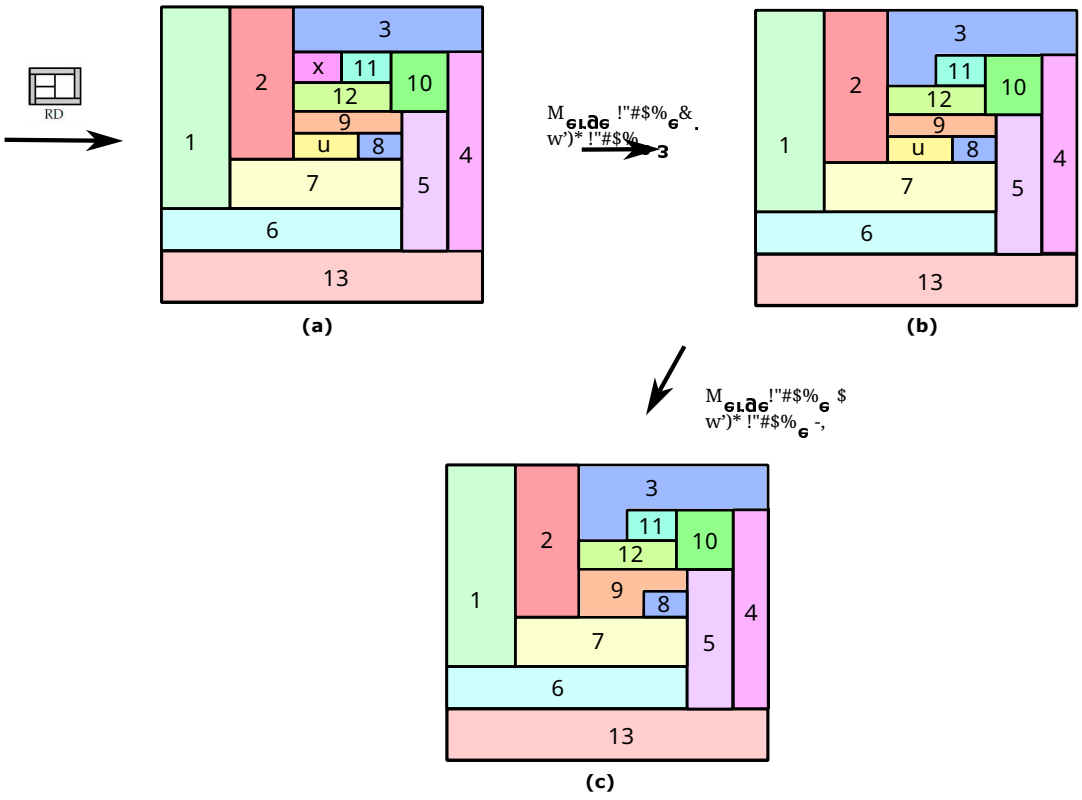


图29：获取与 $G_L^3(V^3, E^3)$对应的矩形对偶的坐标。

组合复杂三角形 K_T 后，会形成各种平面图，其中既有简单模块（平凡情况：墙收缩后成为 L形模块），也有 T 形模块（参见图30）。这一观察表明，仅仅拆分复杂三角形的任意一条边，并不能保证生成的平面图 F_T中会出现 T形模块。因此，如果图包含 K_T 作为子图，则需要进一步处理或采用其他技术，才能在最终设计中生成目标 T 形模块。

此外，我们注意到，通过对 K_T的某条特定内部边进行细分，可以在平面图 F_T中生成一个 T形模块。我们的目标是设计一种算法（即 $Algorithm\ 3$），为任何包含至少一个内部子图 K_T的三角剖分平面图 G_T生成一个T形模块。

5.5 图 G_T 中 KT 的要求

要设计一个包含 T 形模块的平面图（F_T），必须在 G_T^1 中包含一个 K_T 子图（参见图31）。因此，输入图中必须存在一个 K_T 子图，才能在平面图中成功构建 T 形模块。结果揭示了在平面图中构建 T 形模块的基本要求。因此，该过程首先要在输入图 G_T 中识别一个区域 K_T，其顶点按逆时针顺序排列为 a、b、c、d、e 和 f（参见图30）。

要将 T 形模块整合到最终平面图中，有两种类型（参见图32）。我们重点关注 B型的 T 形模块的生成；对于此类型，还可进一步分为四个类别，以便在最终平面图中生成 T 形模块（参见图33）。对于 B型的每个类别（即类别1–4），都可以在平面图中构建一个 T 形模块。本研究着重于在平面图中构建 T 形模块，关注两种可能情况中的一种（即 B型）（参见图32）。在正确性部分中，我们对给定图中存在属于四个类别（针对 B型）中任一类别的此类 T 形模块进行了形式化证明。因此，我们仍聚焦于利用四个已定义类别之一构建 T 形模块，并考虑子图 K_T。

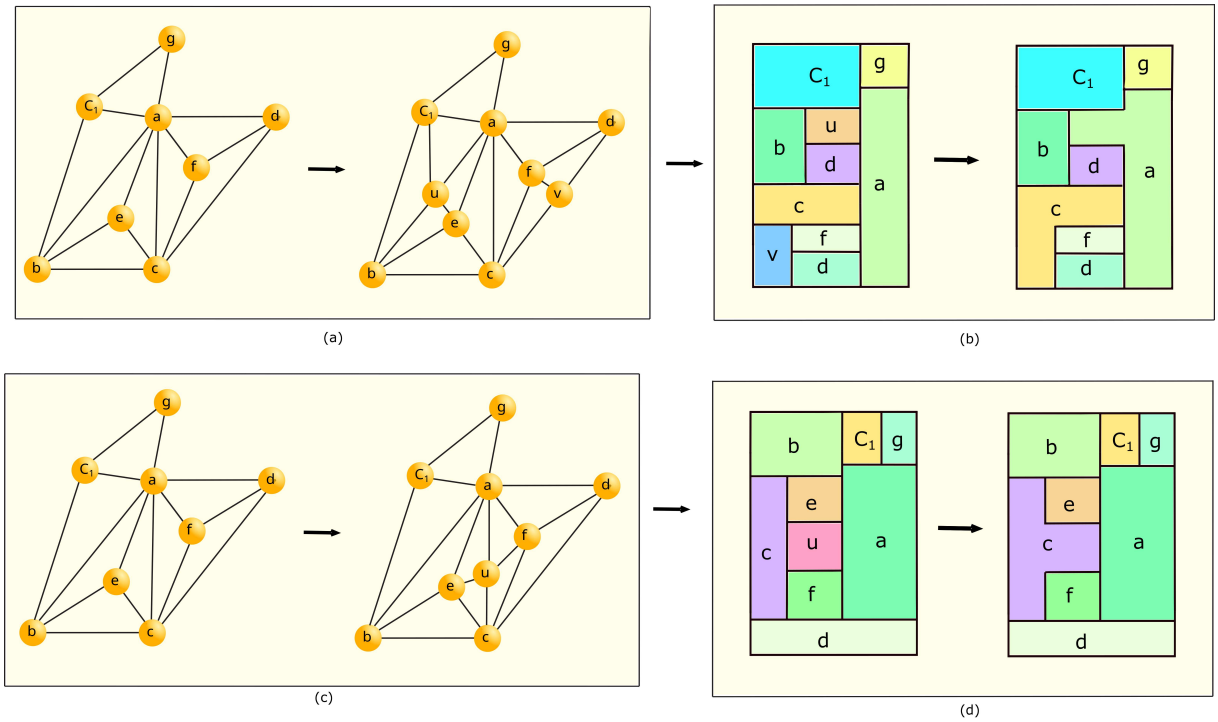
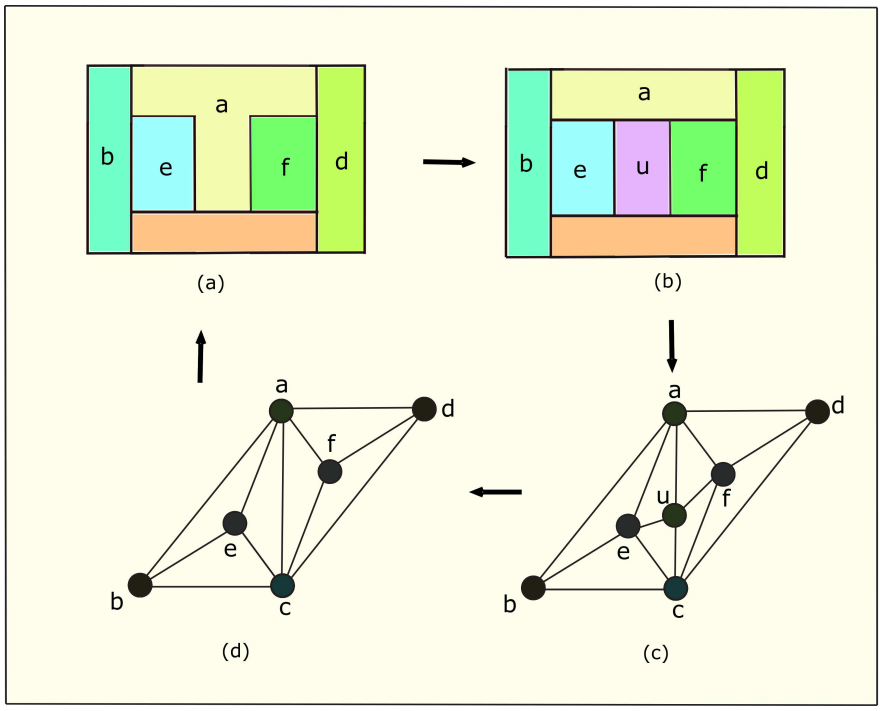


图30：（a）在输入图 K_T 中插入附加顶点 u 和 v，以消除复杂三角形。（b）生成的平面图中会生成一个平凡的 T 形模块和一个 L 形模块。（c）在 K_T 中插入一个附加顶点 u。（d）生成的平面图中会生成一个 T 形模块。



Fi图31：（a–d）在平面图中生成 T 形模块所需的子图 K_T .

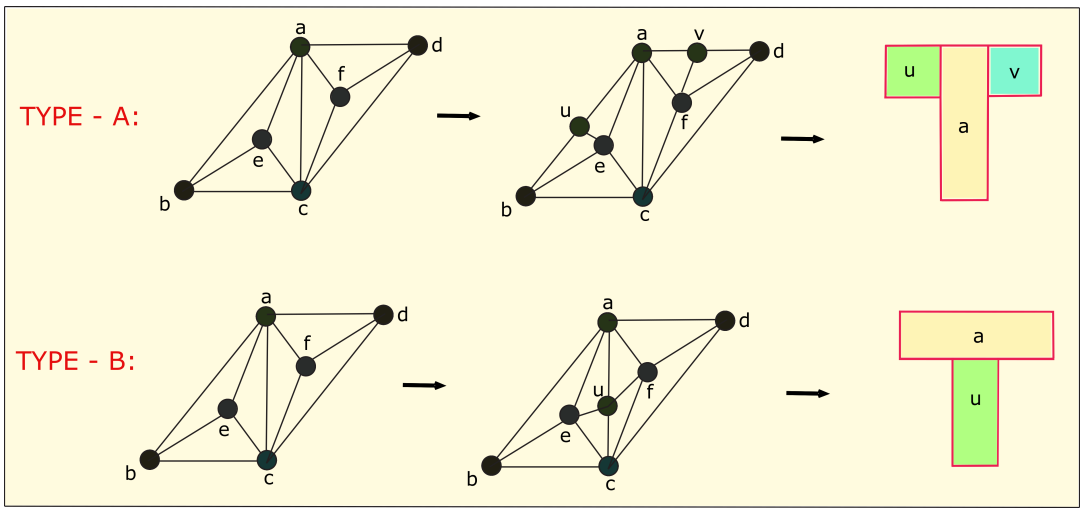
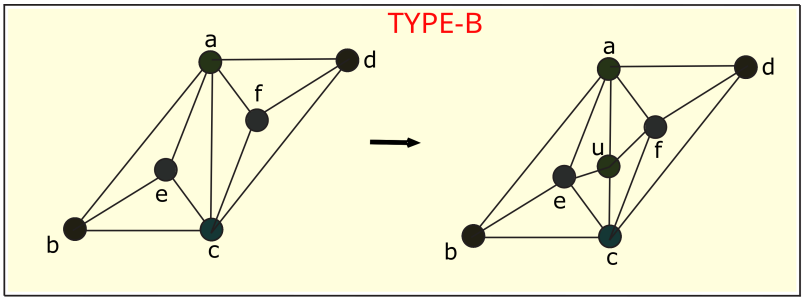
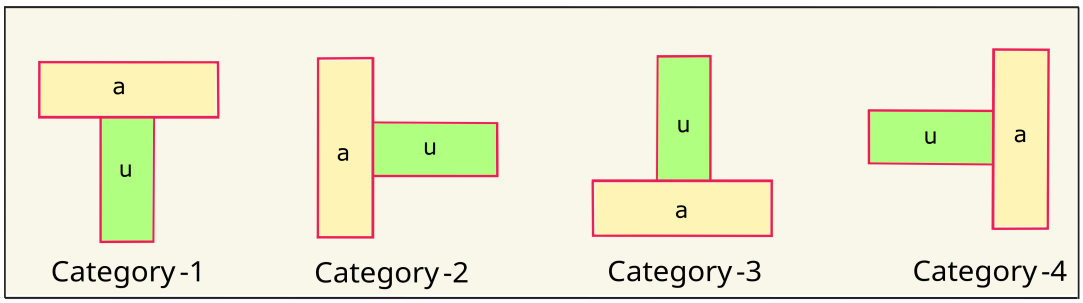


图32：通过两种不同方式生成 T形模块（A型和B型）。



(a)



(b)

图33：(a-b) 将 u 与 a 合并的几种方式（类别：1-4）。

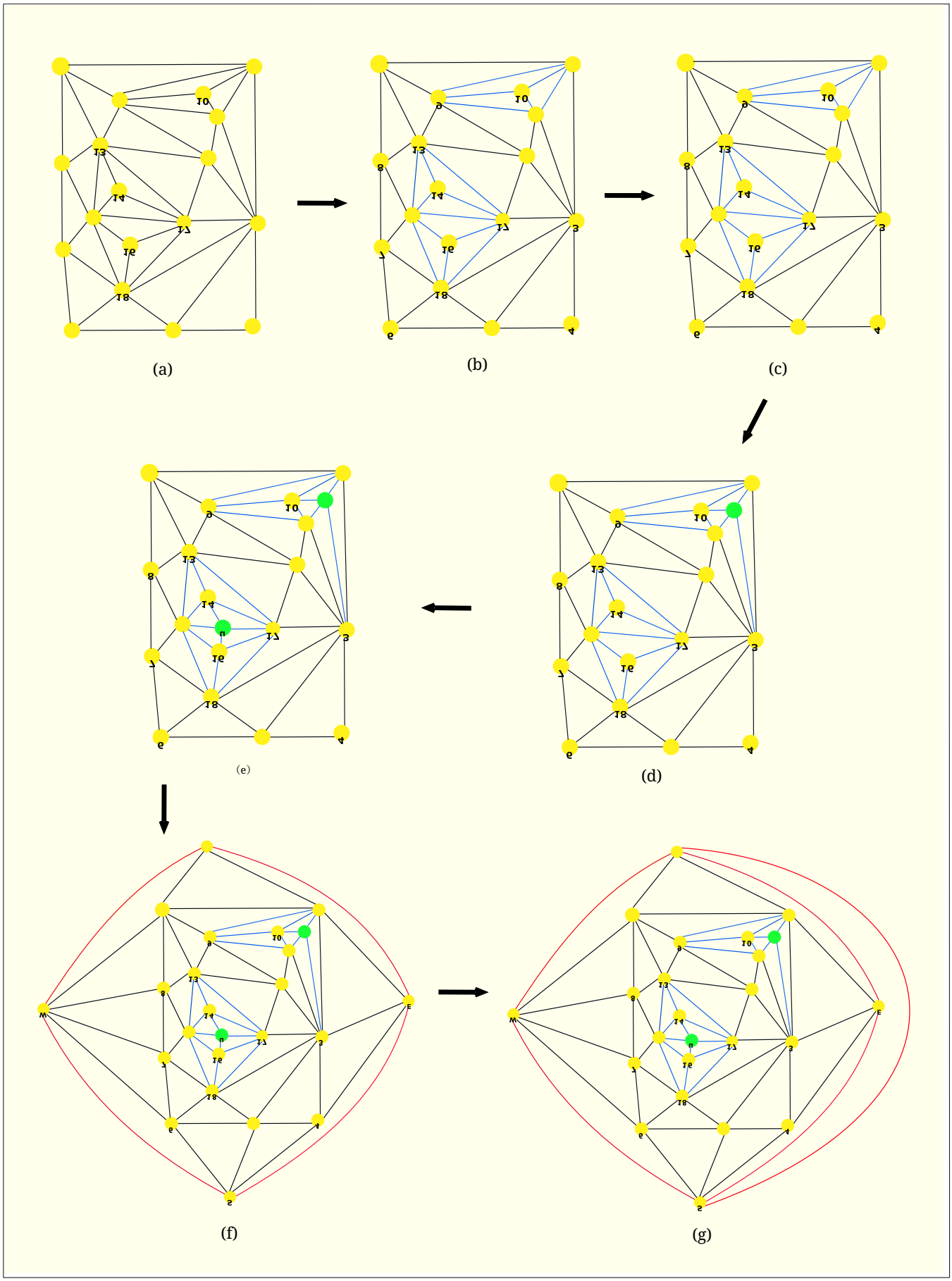


图34：（a）输入图 G_T。（b-c）识别分离三角形（即子图 K_T及其他部分）并进行标记。（d-e）通过引入新节点来打破分离三角形，并将更新后的图标记为 G_T^1。（f-g）应用四完备，并在 G_L^1 中添加一条额外的边，以构造4-连通三角剖分图。

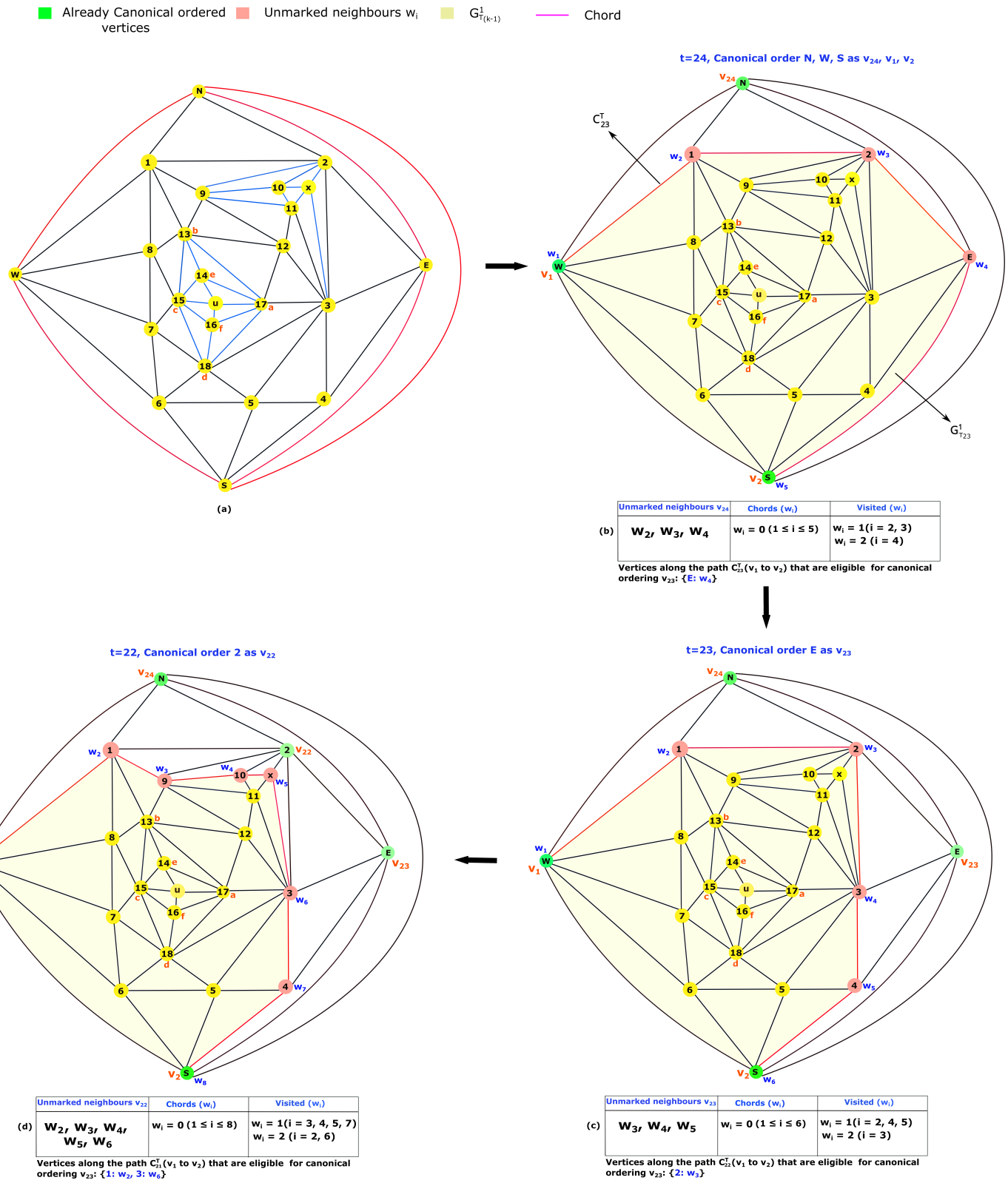


图35：（a-d）为输入图 G^1_T生成典范序图 G^2_T。

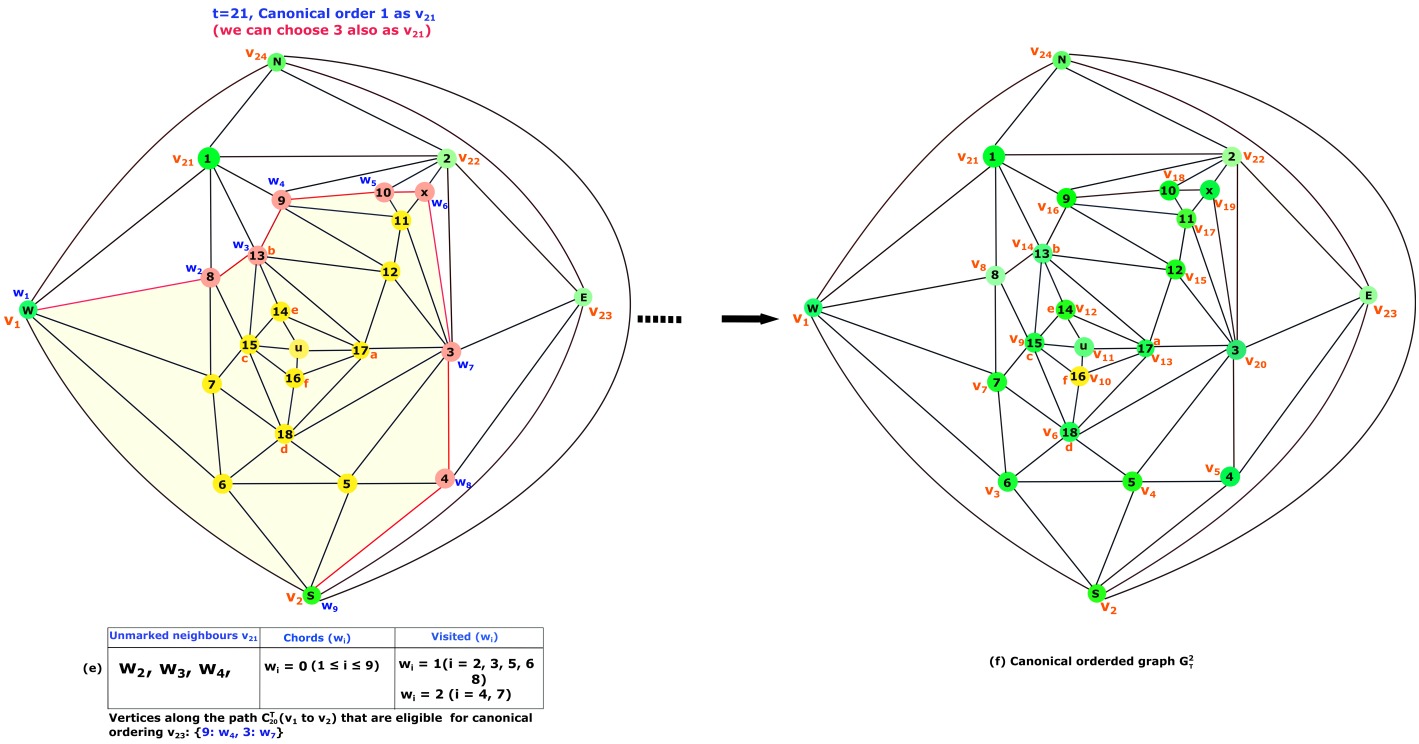


图36：(e-f) 为 in 生成典范序图 G_T^2

图 G_T^1生成典范序图 $G_{T^\circ}^1$

5.6 T形模块在平面图 F_T中的生成

本节通过一个示例说明所提出的算法3：针对包含一个与 K_T同构的内部子图的输入图 G_T，在平面图 F_T 中生成一个 T 形模块。

1. 算法3的步骤 [1 至 4]：复杂三角形识别与移除（子图K_T除外）：

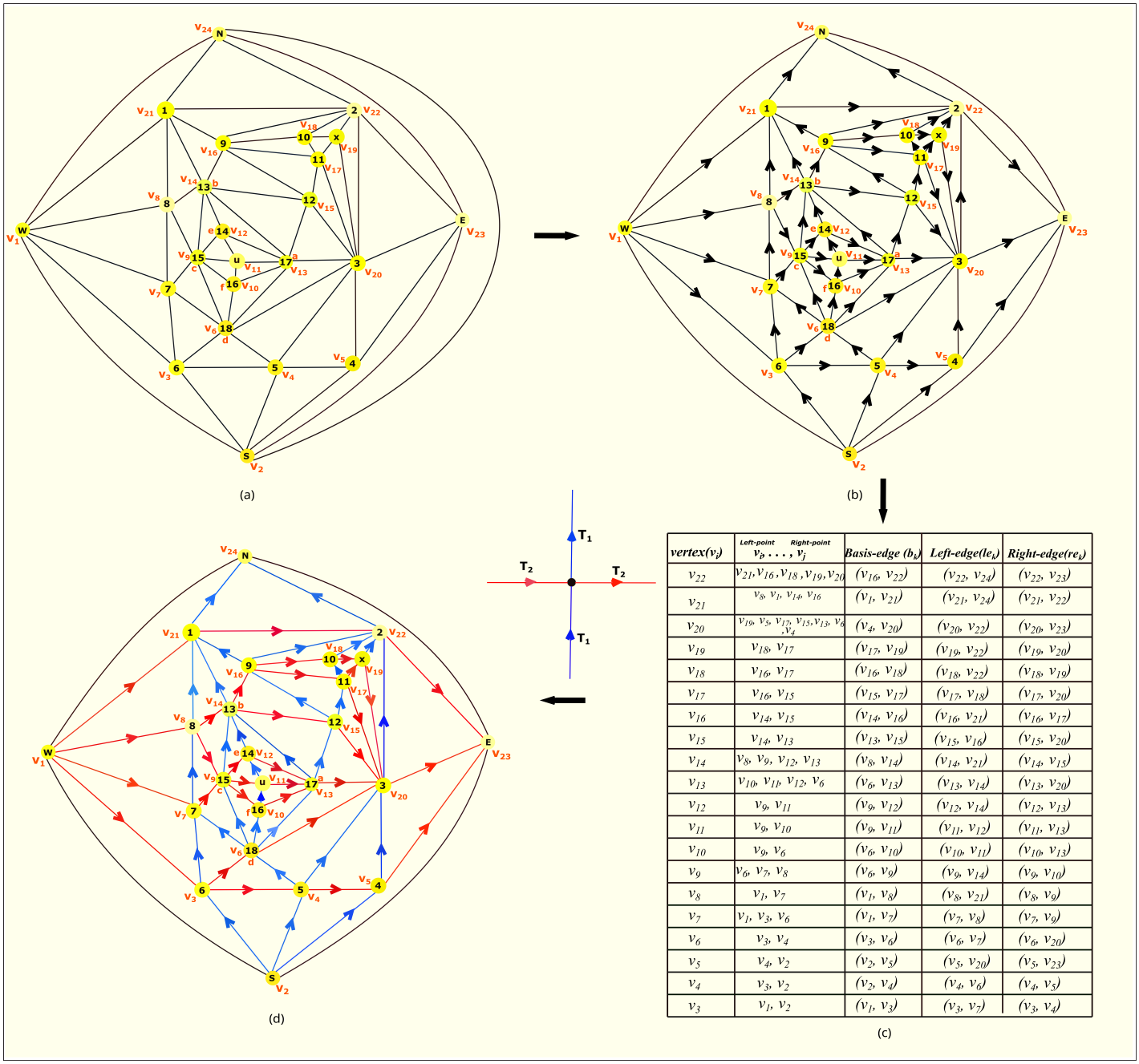
Roy 等人 [24] 描述的方法提供了一种识别并从图中移除复杂三角形的方式。要消除图中的所有复杂三角形，必须先识别一个边的子集（记为 S_L），使每个复杂三角形都至少包含该子集中的一条边。然后，通过在图中插入新顶点 G_T，将 S_T 中的每条边拆分。此系统性修改可确保移除所有复杂三角形，同时保留图的三角剖分性 G_T。

因此，我们采用 [24] 中所述的复杂三角形识别与移除算法，首先识别所有复杂三角形（见图34(a-b)），然后从输入三角剖分图 G_T 中移除所有复杂三角形（如果存在），仅留下 K_T 子图（见图34(c-d)）。如果存在子图 K_T 之外的任何复杂三角形，我们就添加一个新顶点 r 将其拆分，并重新对输入图 G_T进行三角剖分。这样会生成更新后的 G_T，其中不含复杂三角形，但子图 K_T 除外（见图 34(a-d) ）。

2. 步骤 [5至 6] 算法3：复杂三角形移除 K_T 与四向补全阶段 ：

为修改剩余子图 H_1，我们选择随 a 相关联的新顶点 r，将其移除。随后，我们插入新边（即 $\{(u, e), (a, f), (u, a), (u, c)\}$），以保持图的三角剖分结构。所得图记为 G_T^1（参见图34e）。经过这一变换后，G_T^1 不再包含任何复杂三角形。

复杂三角形消除完成后，图 G_T^1 进入四向补全过程：按照 [5], 所述的方法，首先在 G_T^1中确定四条路径 P_4、P_3、P_2、P_1，随后将方向顶点（E、W、S、N）插入各自对应的路径中。这些插入的顶点对应于定义平面图边界的矩形边界模块。此过程构成四向补全阶段，如图34(e-f)所示。完成四向补全阶段后，将边（N，S）引入 G_T^1，从而得到一个4-连通三角剖分图（G_T^1），如图34(f-g)所示。随后，图 G_T^1 进入规范形式



vertex(v_i)	Left-point Right-point $v_p, \ldots, v_j$	Basis-edge (b_k)	Left-edge(le_k)	Right-edge(re_k)
v_{22}	$v_{21}, v_{16}, v_{18}, v_{19}, v_{20}$	(v_{16}, v_{22})	(v_{22}, v_{24})	(v_{22}, v_{23})
v_{21}	v_8, v_1, v_{14}, v_{16}	(v_1, v_{21})	(v_{21}, v_{24})	(v_{21}, v_{22})
v_{20}	$v_{19}, v_5, v_{17}, v_{15}, v_{13}, v_6, v_4$	(v_4, v_{20})	(v_{20}, v_{22})	(v_{20}, v_{23})
v_{19}	v_{18}, v_{17}	(v_{17}, v_{19})	(v_{19}, v_{22})	(v_{19}, v_{20})
v_{18}	v_{16}, v_{17}	(v_{16}, v_{18})	(v_{18}, v_{22})	(v_{18}, v_{19})
v_{17}	v_{16}, v_{15}	(v_{15}, v_{17})	(v_{17}, v_{18})	(v_{17}, v_{20})
v_{16}	v_{14}, v_{15}	(v_{14}, v_{16})	(v_{16}, v_{21})	(v_{16}, v_{17})
v_{15}	v_{14}, v_{13}	(v_{13}, v_{15})	(v_{15}, v_{16})	(v_{15}, v_{20})
v_{14}	v_8, v_9, v_{12}, v_{13}	(v_8, v_{14})	(v_{14}, v_{21})	(v_{14}, v_{15})
v_{13}	$v_{10}, v_{11}, v_{12}, v_6$	(v_6, v_{13})	(v_{13}, v_{14})	(v_{13}, v_{20})
v_{12}	v_9, v_{11}	(v_9, v_{12})	(v_{12}, v_{14})	(v_{12}, v_{13})
v_{11}	v_9, v_{10}	(v_9, v_{11})	(v_{11}, v_{12})	(v_{11}, v_{13})
v_{10}	v_9, v_6	(v_6, v_{10})	(v_{10}, v_{11})	(v_{10}, v_{13})
v_9	v_6, v_7, v_8	(v_6, v_9)	(v_9, v_{14})	(v_9, v_{10})
v_8	v_1, v_7	(v_1, v_8)	(v_8, v_{21})	(v_8, v_9)
v_7	v_1, v_3, v_6	(v_1, v_7)	(v_7, v_8)	(v_7, v_9)
v_6	v_3, v_4	(v_3, v_6)	(v_6, v_7)	(v_6, v_{20})
v_5	v_4, v_2	(v_2, v_5)	(v_5, v_{20})	(v_5, v_{23})
v_4	v_3, v_2	(v_2, v_4)	(v_4, v_6)	(v_4, v_5)
v_3	v_1, v_2	(v_1, v_3)	(v_3, v_7)	(v_3, v_4)

(c)

图 37: （a）表示为 G_T^2 的一种可能的规范序。（b）根据该规范序构建有向图。（c）列出与每个顶点相关的基准边、左边和右边。（d）根据该规范序推导出正则边标记图 G_T^3。

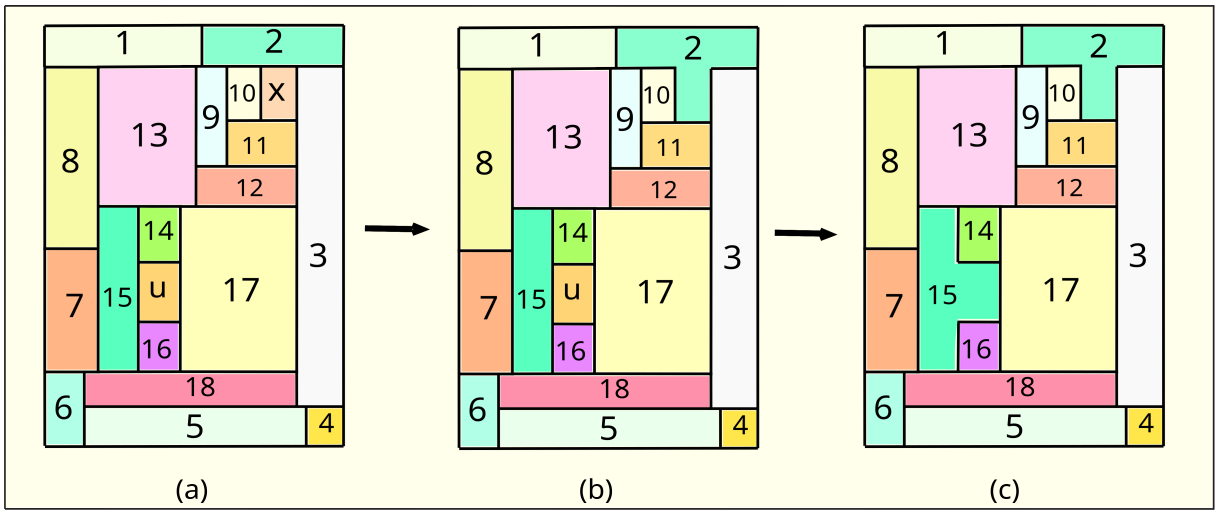


图38：(a) 使用正则边标记图 G_T^3 构建的 G_T^2 的平面图 F_T'；(b–c) 通过合并输入图 G_T 中的模块（将模块 x 与模块2合并，并将模块 u 与模块15合并）得到的最终平面图 F_T，其中包含一个 T 形模块（模块15）。

排序步骤。

3. 步骤7至17（算法3）：规范序：
本节介绍用于从给定的4-连通三角剖分图 G_T^1 生成规范序（G_T^2）的算法（算法 [5]），其定义见定义4。该过程使用算法3中的步骤7–17，并通过图35、36中的示例加以说明。此规范序对于基于初始PTG G_T 构建平面图 F_T 中的 T 形模块至关重要，该初始PTG至少包含一个内部子图 K_T。

参见图35、36：从先前得到的4-连通三角剖分图 G_T^1 开始，使用步骤7–9将每个顶点的 $chord$ 值设为0，并将 G_T^1 中每个顶点的 $status$ 值设为 $False$。将顶点 W 指定（按规范序排序）为 v_1，将顶点 S 指定为 v_2，并将顶点 E 的访问值设为1。随后，按照定义4（参见术语第2节），依据[5], 中所述的方法应用步骤10–17，逐一为 G_T^1 中所有其余顶点指定规范序。由此得到规范序图 G_T^2，从而确立规范顶点序。随后使用此排序（G_T^2）构建正则边标记 G_T^3。

4. 算法3的步骤18：生成正则边标记：
为了构造正则边标记（REL），我们采用 Kant [5] 使用规范序 G_T^2 所描述的方法。请参阅图37(a-d)，其中展示了将该过程应用于 G_T^2 的情况；在此过程中，我们确定基边（b_k）以及每个顶点对应的集合 C_k 和 R_k。随后利用这些连通分量生成正则边标记图 G_T^3（见图37d）。

5. 算法 m 3：矩形平面图的生成 图：
使用规范序概念生成正则边标记（G_T）后，我们利用该正则边标记为生成的图 G_T^2 构造矩形平面图（F_T'）。具体做法是遵循 Bhasker 和 Sahni [9] 所描述的方法（见图38a）。

6. 算法3的步骤 [**20** 至 **24**]：通过合并模块，在正交平面图中生成 T 形模块：

$Merge\ Rooms$ 函数描述了如何整合矩形模块，这些模块对应于移除 G_t 中的复杂三角形时引入的额外顶点。该函数需要三个输入：由图 G_T^2 导出的矩形平面图 F_T'、规范有序图 G_T^2 本身，以及额外节点集合 $Enodes_T$。输出是在平面图中生成一个 T 形模块，使其与 K_T 的结构相符。

参见图 38(b,c)（其中 S_T =(2, 11) 和 $Enodes_T$ 为 x），我们使用 $function\ Merge\ Rooms$，从矩形平面图 F_T' 中获得了带有 T 形模块的平面图 F_T。

因此，给定一个带有内部子图 K_T 的 G_T（三角剖分图），可以使用我们提出的算法3，在相应平面图 F_T 中构造一个 T形模块。

6 算法的正确性

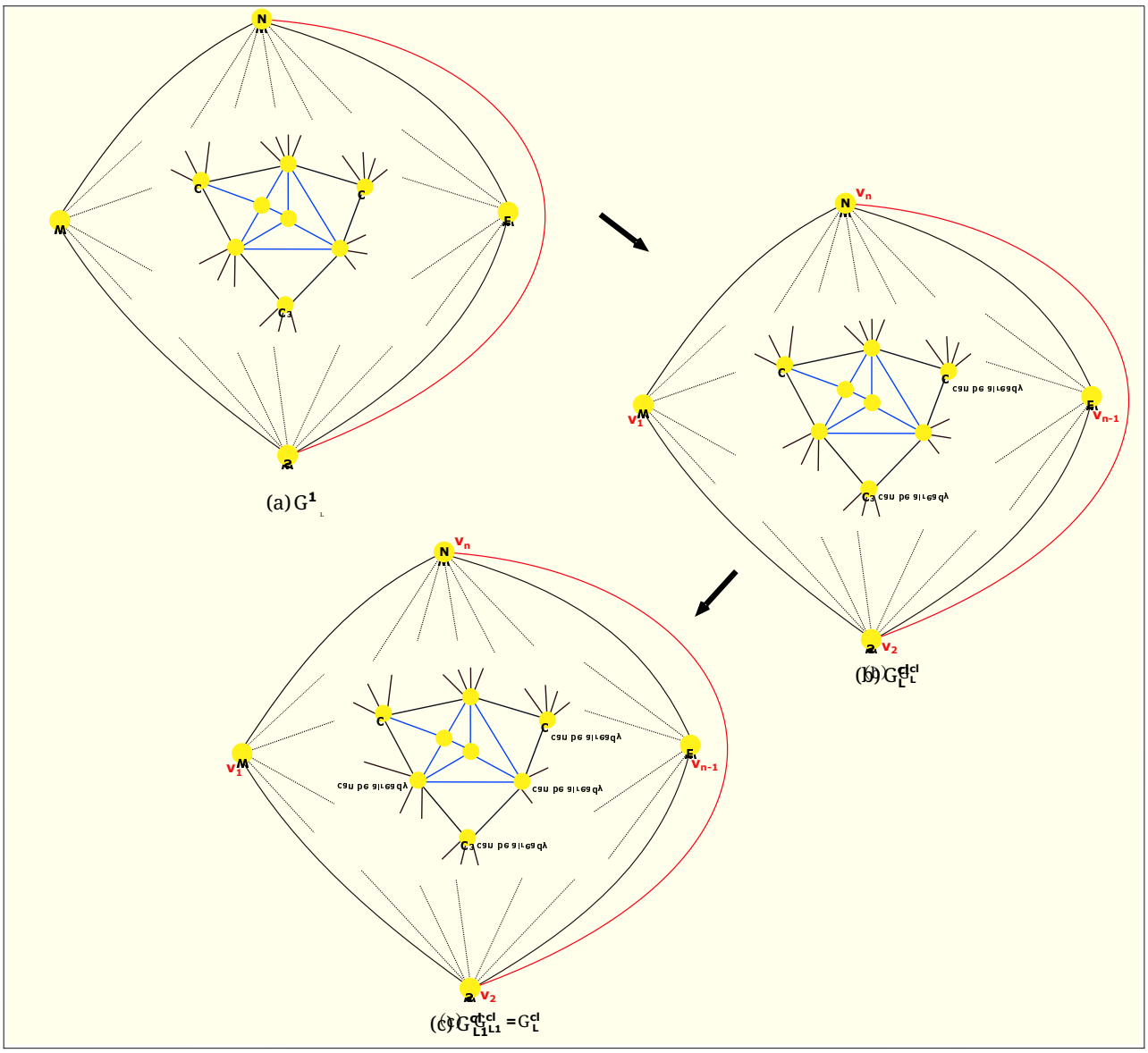


图39：（a）一个4-连通三角剖分图 G_L^1。（b）通过对 G_L^1 的所有顶点（$\{a, b, c, d, u, C_1\}$ 除外）进行规范序排序，并使用算法1的步骤（7–28）生成 G_L^{cl}。（c）通过调用算法1的函数 $Types\ of\ Priority\ label_L$ 生成 G_{L1}^{cl}。

定理 1. 对于给定的 G_L（V、E），其至少包含一个与 K_L 同构的内部子图，算法1生成的正交平面图 F_L 必然包含一个与子图 K_L 对应的L形模块。

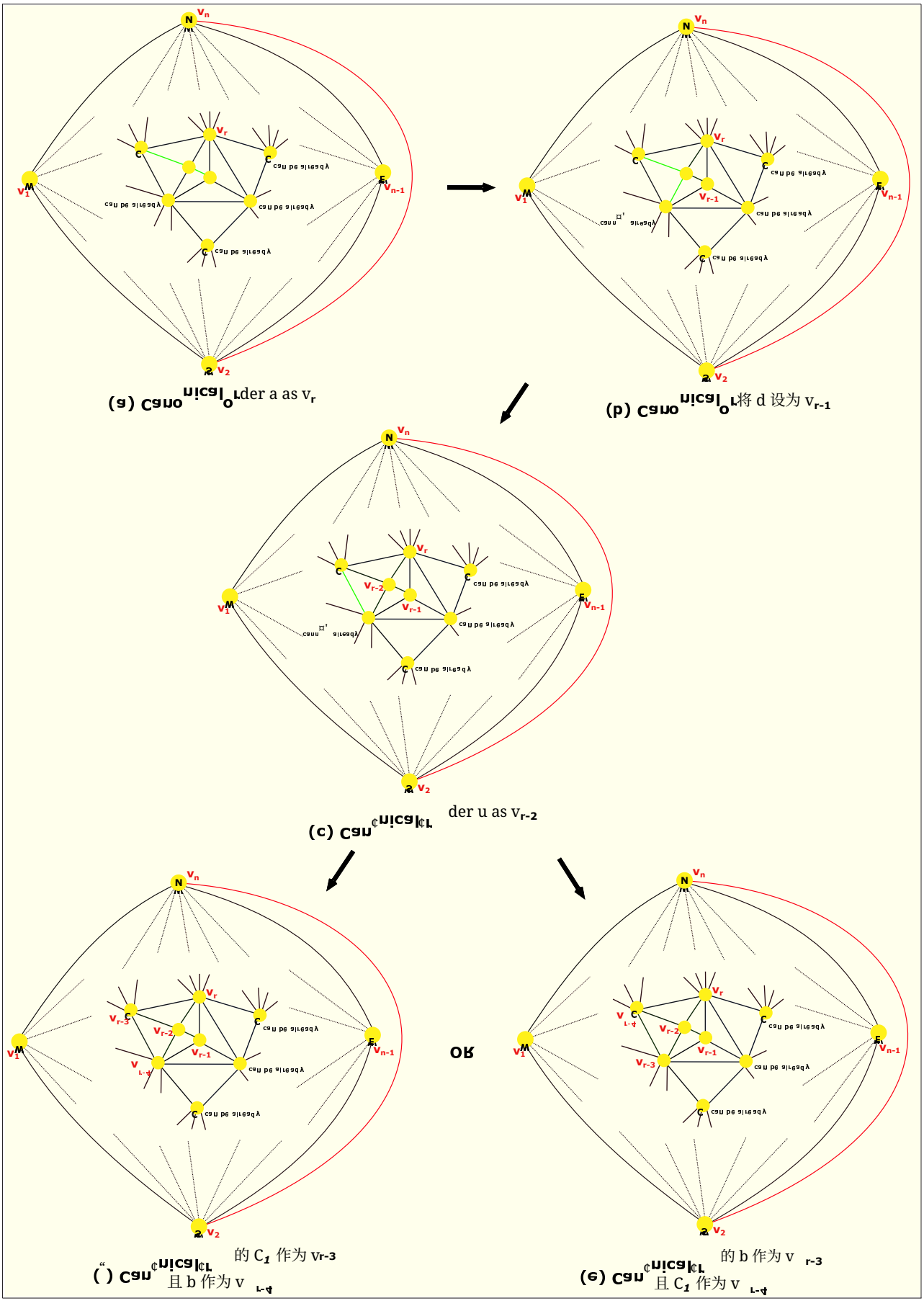


图 40：（a-e） G_L^1 中存在 D 类（规范序）。

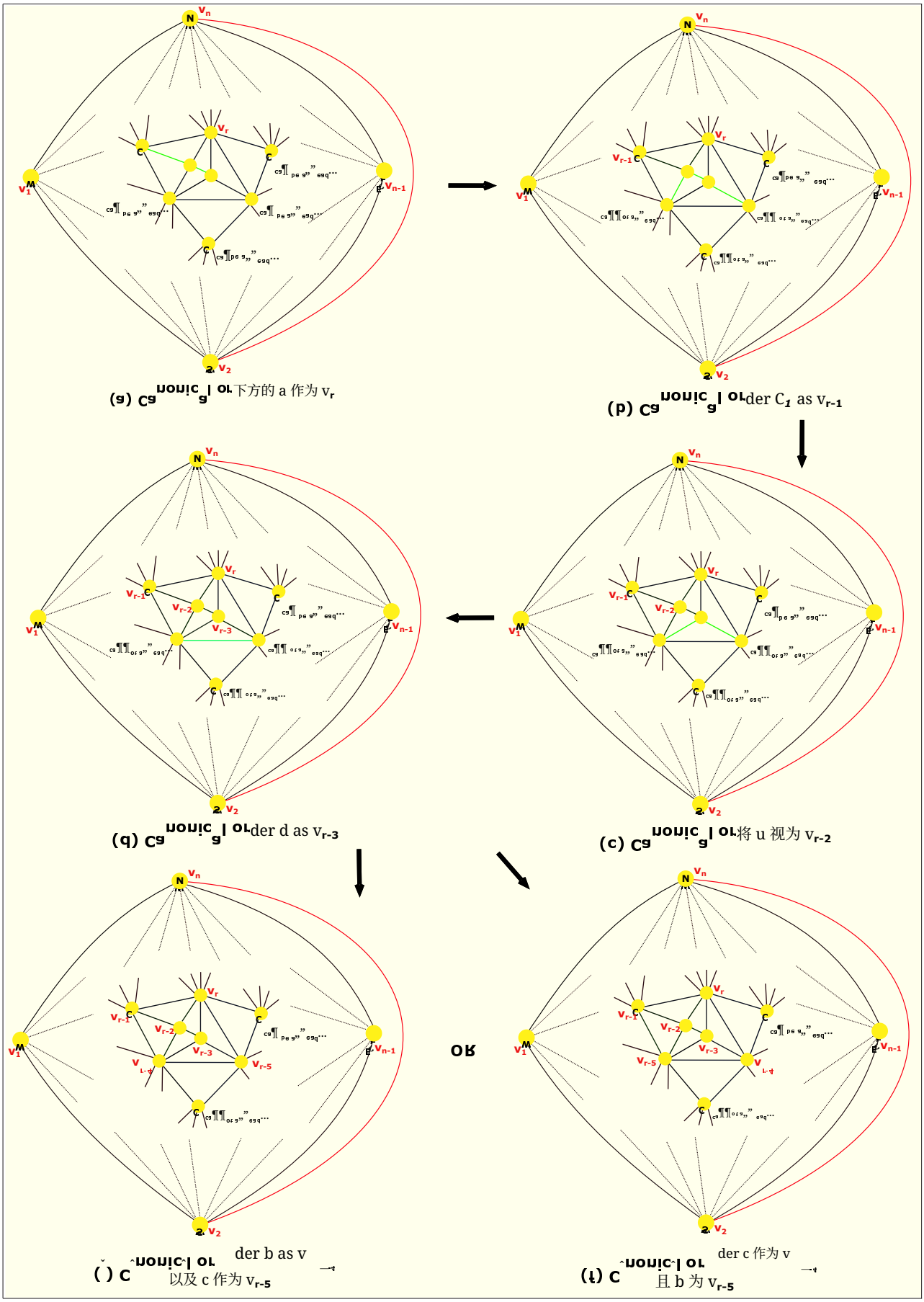


图41：（a-f）G_L^1中存在 Category-F（规范序）。

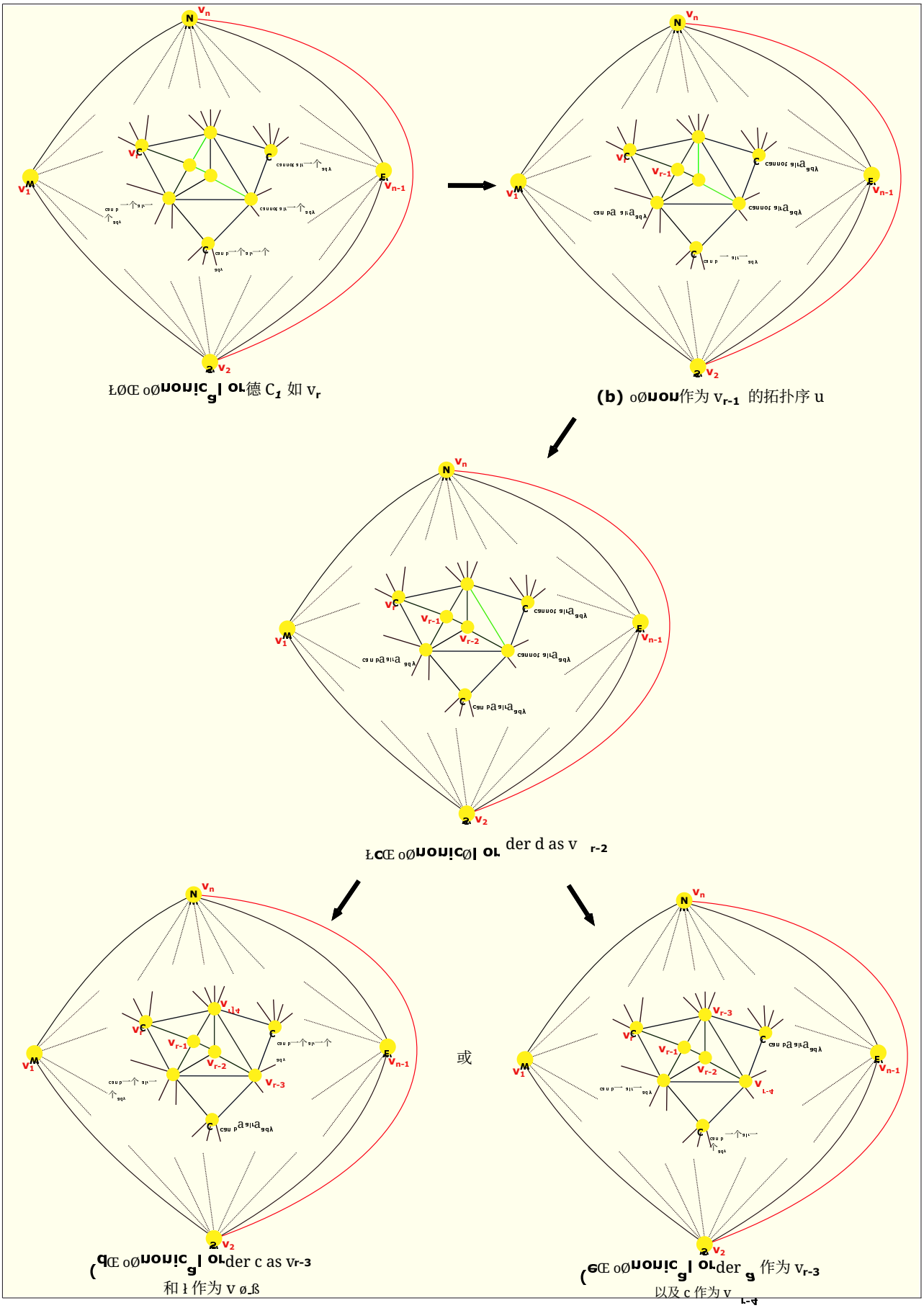


图 42：(a-e) 在 G_L^1 中存在 A 类（规范序）。

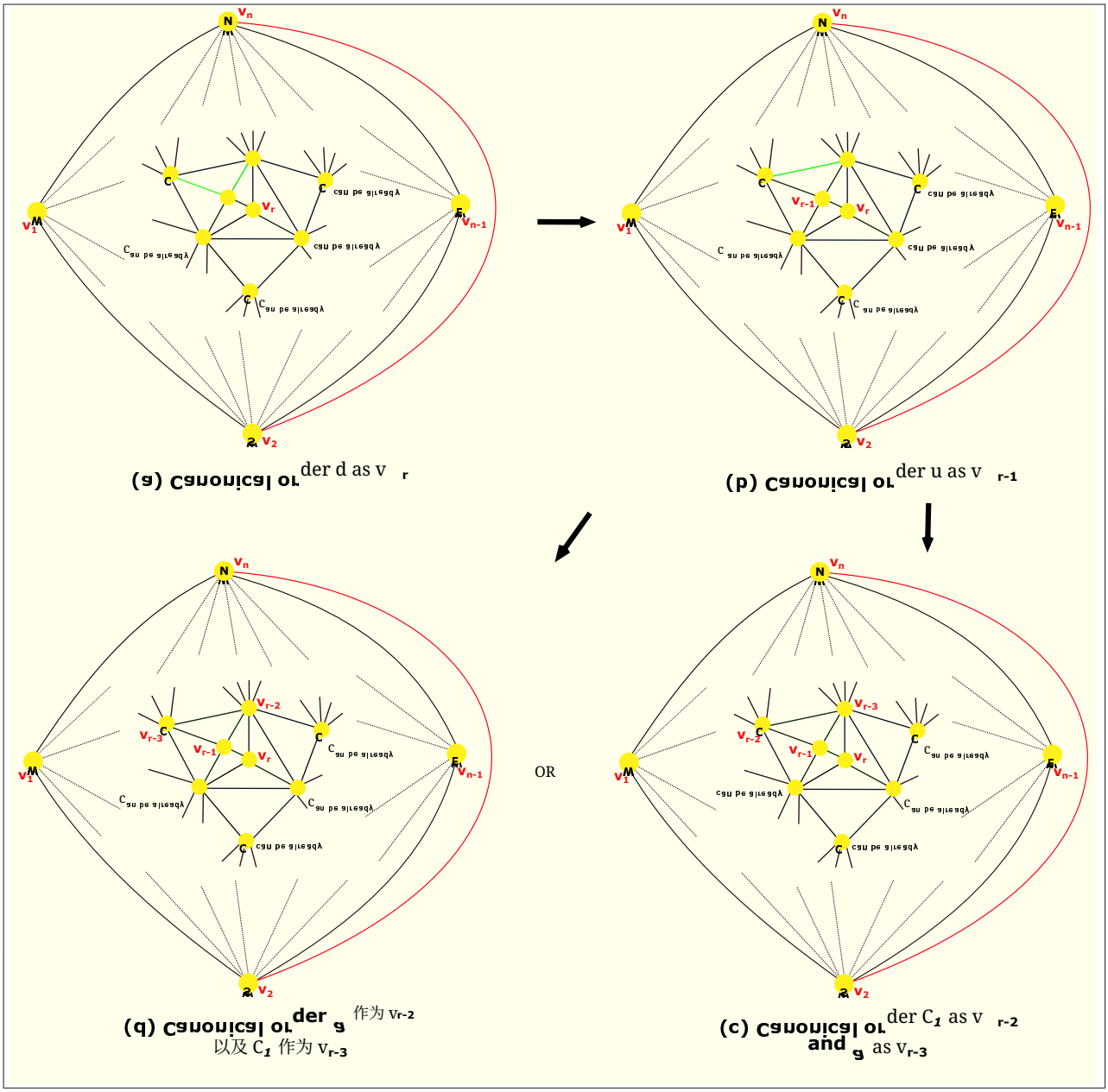


图 43: （a-d） G_L^1 中存在 B 类或 C 类（规范序）。

证明。考虑一个三角剖分图 G_L ，其中包含一个与 K_L同构的内部复杂三角形。正如 Kant [5]所证明的，对于每个四连通三角剖分图，都可以确定一个规范序。随后可以根据该规范序得到正则边标记（REL）。按照 Kant [5]所述的方法，该 REL 可作为构造矩形平面图的基础。

我们的目标是在与输入图 G_L对应的平面图中构造一个 L形模块。为此，我们需要一个包含额外模块（称为 u）的 RFP。要创建 L 形模块，需要按照指定的优先级顺序和 REL，将模块 u 与模块 a 或模块 b合并。为此，我们从一个包含内部 K_L 子图的图 G_L 开始。接着，通过引入一个新顶点u来修改 G_L ，从而细分 K_L的外部边。此修改会得到一个四连通平面三角剖分图，记作 G_L^1；这是建立规范序的关键前提。转换过程还包括消除所有剩余的复杂三角形、进行 4-补全，以及添加一条辅助边（N, S），这些步骤都是满足规范序要求所必需的。接下来，为G_L^1的顶点指定优先级规范序时，必须按照特定顺序优先排列顶点 $\{a, d, u, C_1\}$。然而，G_L^1 的其余顶点（ $\{a, d, u, C_1\}$ 顶点除外）不遵循任何特定的优先级顺序。 $\{a, d, u, C_1\}$ 顶点的优先级排序可分为六个类别（类别A至F）：请参阅图17、图18和图19；示例见第5.3小节。

我们旨在说明，通过执行所提出的算法1，始终可以在图 G_L 的平面图 F_L 内部构造一个 L形模块，该模块表示 G_L的内部部分 K_L 。为证明这一点，我们将逐步讲解算法1的每个步骤，说明其工作原理，并确认其能够成功构造所需的模块。

1. 步骤1至4（算法1）：给定一个至少包含一个 K_L 作为子图的输入图 G_L ，我们首先使用 [24] 中提出的算法检查除 K_L 之外是否存在任何复杂三角形（见图12）。如果 G_L 中存在此类复杂三角形，我们将应用 [24],中提出的复杂三角形消除算法，通过引入附加顶点和额外的边来消除它们（见图13c）。这一过程确保在 G_L中除 K_L之外的所有复杂三角形均被消除。由此，我们获得一个修改后的图 G_L ，其中除 K_L之外不含复杂三角形。

2. 步骤5至6（算法1）：为了修改剩余的 K_L 子图，我们首先通过插入一个新顶点 u，替换 K_L 的内部边（a， b）。为确保更新后的图仍然经过三角剖分，如图13d所示，我们引入了额外的边。我们将该修改后的图记为 G_L^1。接下来，我们应用 [5], 中所述的四连通算法，将其应用于G_L^1 （见图14b）。在最后阶段，我们在顶点 N 和 S 之间添加一条边，以形成 G_L^1 （4-连通图）。因此，可以始终从输入图 G_L^1 构造出一个4-连通图 G_L^1 （见图14c）。

3. 步骤7至53（算法1）：从先前得到的 4-连通三角剖分图 G_L^1 出发，我们按照定义4中说明的过程（参见术语第2节），执行步骤（7–28），为 G_L^1中的所有可能顶点指定规范序，但排除集合 $\{a, C_1, d, u\}$。当排序中下一个可能的顶点仅来自集合 $\{a, C_1, d, u\}$时，我们返回 G_L^2 并转至步骤（29–53）。此时，我们尝试按照类别（A–F）对集合 $\{a, C_1, d, u\}$ 中的顶点进行系统排序，并依次检查各个类别。如果在任何一个类别下都存在有效排序，我们就采用该排序，并继续生成规范有序图 G_L^2，用于构造正则边标记：参见图20至图26。

因此，要确认规范有序图 G_L^2的存在，只需证明在生成的图 G_L^{cl} 中，集合 $\{a, C_1, d, u\}$ 总是存在一种有效的、基于类别的排序。根据引理1，集合 $\{a, C_1, d, u\}$ 总是存在一种规范优先级排序，它对应于六种可能类别中的任意一种：A 至 F 类。

4. 步骤54（算法1）：利用先前生成的规范有序图G_L^2 ，现在继续采用算法2，为G_L^2构造正则边标记（REL，即 T_1 或 T_2），也就是G_L^3。若要在平面图 F_L中构造 L形模块，我们必须将额外模块 u 与 a 或 b 合并，使模块 a变为 L形，或模块 b 变为 L形，并将模块 a 构成 L形：顶点 a 与 C_1 和 u 相关的方向（即 T_1或 T_2）必须相反。同样，若要将模块 b 构成 L形，顶点 b 与 C_1 和 u 相关的方向（即 T_1 或 T_2）必须相反（见图17、图18：顶点 a 与 C_1 和 u 相关的方向（即 T_1或 T_2）相反；另见图19：顶点 b 与 C_1 和 u 相关的方向相反）。

（a.）算法2的步骤（1–10）：我们按照[5]中所述的方法论，为输入图 G_L^2生成正则边标记。根据连接顶点 C_1 与顶点 a 和 b的边所具有的边类型 T_1 或 T_2，再确定下一步适当的操作。

（b.）算法2的步骤（11–23）：已知顶点 u的邻点为 a、b、d、C_1，顶点d的邻点为 a、b、c、u，因此在根据规范序图 G_L^2构造正则边标记（REL）的过程中，关联于 u和 d的边具有固定的方向（T_1、T_2）（参见第2节中 REL 的定义）。现在，根据上述生成的 REL，可能出现以下两种情况：

情形A：如果模块 a 与模块 u 和 C_1都共享类型不同的墙（ T_1 或 T_2），或者模块 b满足相同条件，那么可以直接将模块 a 与 u 合并并返回 $m = 1$，或将模块 b 与 u 合并并返回 $m = 2$。这将形成一个 L形模块（见图28d、29）。

情况B：见图44(a-h)：如果不满足情形A中所述的条件，则输入图必定包含两个不同顶点，记为 x 和 y。这两个顶点定义为 $x \in (\text{nbd}(a) \cap \text{nbd}(C_1)) \setminus u$和 $y \in (\text{nbd}(b) \cap \text{nbd}(C_1)) \setminus u$，且条件为 $x \neq y$；否则，这将与子图 K_L 必须存在于 G_L 中的要求相矛盾（见图44(a–b)）。由此产生两个不同的子情形：

子情形A：如果模块 a 和 C_1 之间共享的墙（即 T_1 或 T_2）与模块 x 和 C_1之间共享的墙方向相反，那么适当的操作顺序是先翻转边（C_1，x），再翻转边（a，C_1）。完成这些修改后，我们返回 $m = 1$（参见图44(e–h)）。子情形B：如果 a 和 x 各自与 C_1 共享的墙对齐（即相同），那么只需翻转边（a，C_1）即可，之后我们返回 $m = 1$（参见图44(f–h)）。

由于根据定义的优先级排序（类别：A-F），顶点 a 相对于 C_1 和u 的方向必须始终相反，或者这一条件必须对顶点 b成立。因此，我们可以选择将模块 u 与 a 或 b 合并，通过返回相应的 m值来形成 L形状。因此，始终可以从规范有序图 G_L^2生成正则边标记 G_L^3。

5. 步骤55至63（算法1）：基于先前生成的正则边标记 G_L^3 of G_L^2，我们现在将按照 [5] 中所述的算法，使用 T_1 和 T_2 方向构建矩形平面图（ RFP ） F_L'。获得 RFP F_L' 后，我们将应用函数 $Merge\ Rooms$。此函数的输入包括：规范有序图 G_L^2、生成的 RFP F_L'、参数 m，以及额外节点的集合（$Enodes_L$：即在 G_L 中引入、用于消除复杂三角形的附加顶点，不包括 K_L）：图29。

（a.）步骤56至58（算法1）：现在，我们将使用集合 $Enodes_L$ 合并 F_L' 中的额外模块。具体而言，对于 $Enodes_L$ 中的每个顶点， F_L' 中相应的模块（即与 u_i 关联的模块）将与 F_L' 中的模块 a_i 或 b_i 合并。

（b.）步骤59至63（算法1）：对于修改后的 K_L 顶点，如果 $m = 1$，额外模块 u 将与 F_L'中的模块 a 合并；否则，它将与 F_L'中的模块 b 合并。最终，此过程生成正交平面图 F_L，其中包含一个与子图 K_L对应的 L形模块。

因此，对于给定的包含至少一个内部 K_L的平面三角剖分图 G_L（V，E），算法1始终会在正交平面布局 F_L中生成一个 L形模块（对应于内部 K_L）。

□

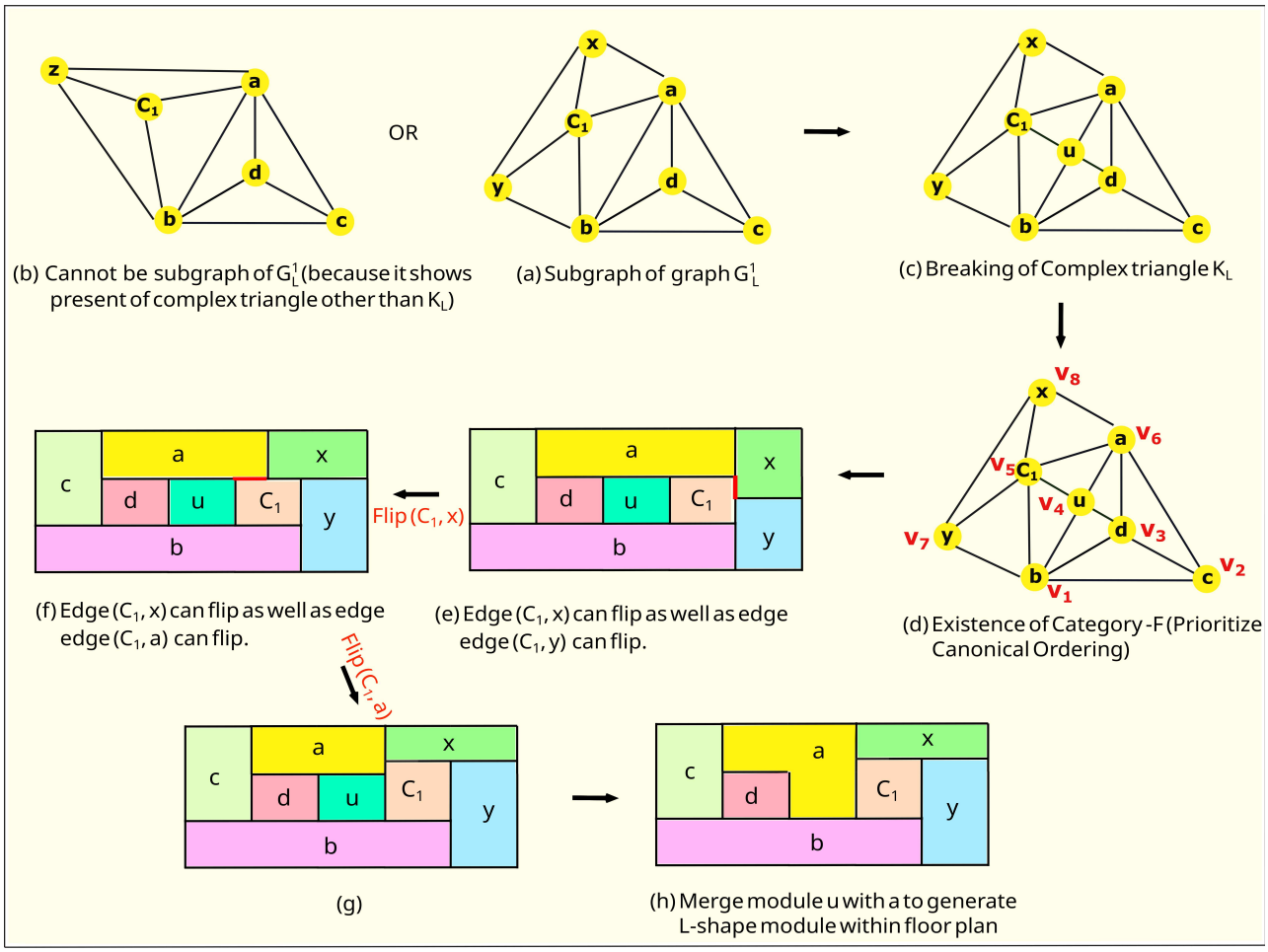


图44：(a-h) 通过翻转一条边，平面图中存在一个 L-形模块。

引理1. 对于由带有内部子图 K_L的输入图 G_L 生成的图 G_L^1，集合 $\{a, C_1, d, u\}$总是存在一种规范优先序，该优先序对应六种可能类别中的任意一种：A 至 F 类。

证明。对于生成的图 G_L^1，其中包含子图 K_L，我们需要分别检查从类别 A 到 F 的集合 $\{a, C_1, d, u\}$ 的规范优先序。可供排序的仅剩顶点来自集合 $\{a, C_1, d, u\}$（见图39(a-b)），而顶点 v_n,....., v_{r+1} 已在G_L^1中按规范形式排序。因此，G_L^1 中下一个可被赋予标签 v_r 的顶点必须来自集合$\{a, C_1, d, u\}$。

1. 参见图39c：在第 r步（算法1中的第11步）：顶点 u 最初无法从集合$\{a, C_1, d, u\}$ 中排序为 v_r，因为 u的度为 4，且与顶点 a、 d、 b和 C_1相邻。其中，至多一个顶点，即 b，可能已在对集合 $\{a, C_1, d, u\}$ 中的顶点排序之前完成排序（此处，“已排序”指在对 $\{a, C_1, d, u\}$排序之前已按规范序排序的顶点）。这意味着 $visited\ (u) < 2$，这表明无法将 u 排序为 v_r。因此，可排序为 v_r的顶点必须来自集合{a、 d、 C_1}，但不包括 u。

2. 情况1：假设顶点 a可以在 G_L^1 中按规范序排列为 v_r（从集合 $\{a, d, C_1\}$中选取），即$visited\ (a) \geq 2$ 且 $chord\ (a) = 0$。将 a 按规范序排列为 v_r （参见图40a）。

将 a 按 v_r排序后，剩余图$G_{L_{r-1}}^1$ （$G_{L_{r-1}}^1 = G_L^1$ - $\{v_n$,....., $v_{r+1}\}$）有一个连通分量 C_{r-1}^L （C_{r-1}^L指：$G_{L_{r-1}}^1$ 的外部面的边界是圈 C_{r-1}^L，其中包含边（v_1, v_2）），该连通分量包含一条路径

其中包含子路径 $C_1 - u - d$（见图40a）。这意味着现在我们可以将 d 或 C_1 排序为 v_{r-1}。假设我们将 d 排序为 v_{r-1}：见图40b（如果将 C_1 标记为 v_{r-1}，则存在F类规范序：见图41(a-f)）。之后，剩余图 $G^1_{L_{r-2}}$ 有一个连通分量 C^L_{r-2}，其中包含一条路径，该路径包含子路径 $C_1 - u - b$（见图40b）。这意味着现在可以将 u 规范排序为 v_{r-2}，因为 $chord(u) = 0$ 且 $visited(u) \geq 2$（因为 d 和 a 已经按规范序排序）。因此，将 u 按规范序排序为 v_{r-2}：见图40c。之后，对 C_1 或 b 进行排序很简单（见图40(d,e)）。因此，存在集合{a, C_1, d, u}相对于图 G^1_L的D类规范排序。

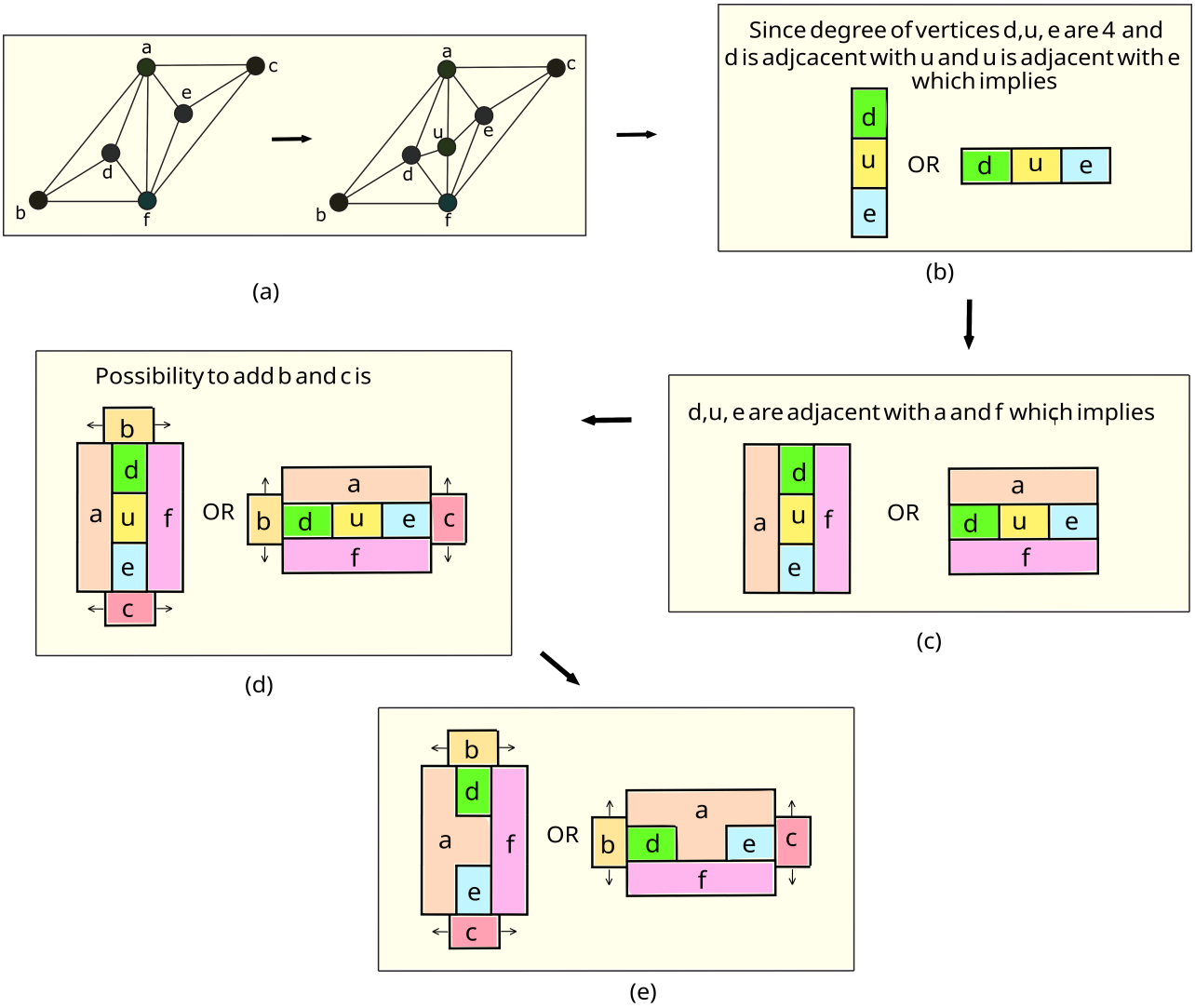


图45：(a-e) 与 K_T 相对应的平面图中存在 T形模块。

3. 情况2：假设顶点 a 无法在 G^1_L中按 v_r 的规范序排列。那么，在 G^1_L 中可按 v_r 的规范序排列的可能顶点为 C_1 或 d。

(a). 子情况2(i)：假设顶点 C_1 可以在G^1_L 中按 v_r 的规范序排列（从集合 $\{C_1, d\}$中选取），即 $visited(C_1) \geq 2$ 和 $chord(C_1) = 0$。将 C_1 按 v_r的规范序排列：见图42a。

将 C_1 排序为 v_r 后，剩余图 $G^1_{L_{r-1}}$ 有一个连通分量 C^L_{r-1}，其中包含一条必须包含子路径 $a - u - d - c$ 的路径（见图42a）。这表明我们可以轻松地将 u 标记为 v_{r-1}，因为$chord(u) = 0$，并且 $visited(u) \geq 2$（因为 C_1 和 b 已经按规范序排列）。因此，将标签 u 标记为 v_{r-1}：见

图42b。之后，剩余图$G^1_{L_{r-2}}$有一个连通分量 C^L_{r-2}，其中包含一条必须包含子路径 $c-d-a$ 的路径（见图42b）。这意味着我们现在可以将 d 标记为 v_{r-2}，因为 $chord\,(d)=0$，并且 $visited\,(d)\geq 2$（因为 u 和 b 已经按规范序排列）。因此，将 d 按规范序排列为 v_{r-2}：见图42c。之后，对 a 或 c 进行排序就很简单了（见图42(d-e)）。因此，存在集合 $\{a, C_1, d, u\}$ 相对于图 G^1_L 的一种 A 类规范序。

(b). 子情况 2(ii)：假设顶点 C_1 不能按规范序排为 v_r，这意味着唯一可能按序排为 v_r 的顶点是 d，即 $visited\,(d)\geq 2$ 和 $chord(d)=0$。将 d 按规范序排为 v_r：见图43a。将 d 排为 v_r后，剩余图 $G^1_{L_{r-1}}$ 有一个连通分量 C^L_{r-1}，其中包含一条必定包括子路径 $a-u-C_1$ 的路径（见图43a）。这表明我们可以轻松地将 u 标记为 v_{r-1}，因为$chord(u)=0$ 和 $visited(u)\geq 2$（因为 d 和 b 已经按规范序排列）。因此，将 u 按规范序排为 v_{r-1}：见图43b。此后，剩余图$G^1_{L_{r-2}}$有一个连通分量 C^L_{r-2}，其中包含一条必定包括子路径 $a-C_1$的路径（见图43b）。这意味着现在可以将 C_1标记为 v_{r-2}，然后将a 标记为 v_{r-3}（见图43c）；或者将 a 标记为 v_{r-2}，然后将 C_1 标记为 v_{r-3}（见图43d）。因此，相对于图 G^1_L，集合 $\{a, C_1, d, u\}$ 存在 B 类或 C 类规范序。

因此，对于由输入图 G_L 生成、且内部子图为 K_L的图 G^1_L，使用算法1的步骤7至53，总能得到集合 $\{a, C_1, d, u\}$的一个规范优先排序，该排序对应六种可能类别中的任意一种：A 至 F 类。

□

定理 2. 对于给定的 G_T（V， E），若其中至少包含一个与 K_T同构的内部子图，则算法3会生成一个正交平面图 F_T，其中必然包含一个与子图 K_T对应的 T形模块。

证明. 考虑一个包含内部复杂三角形 K_T的三角剖分图 G_T。我们的目标是在输入图 G_T的平面图 F_T 中创建一个 T 形模块。为此，需要构造一个带有额外模块（称为 u）的矩形平面图。要创建 T 形模块，需要将模块 u 与模块 a 或 c合并。为便于进行这一过程，我们首先考虑一个包含内部K_T 子图的图 G_T。随后，通过引入两个附加顶点 u 和 v，对 G_T 进行修改，从而细分 K_T的一条内部边。此修改得到一个4-连通三角剖分图，记为 G^1_T，这是建立规范序的关键前提。该变换过程还包括消除所有剩余的复杂三角形、执行4-补全，以及插入一条辅助边（N， S）。接下来，我们将为 G^1_L 的顶点指定规范序，然后进行正则边标记（REL）生成，并据此生成矩形平面图（RFP），最后合并RFP中的模块，以生成包含一个 T形模块的 F_T。我们的目标是证明，按照所提出的算法3，总能在图 G_T 的平面图 F_T 中构造一个 T形模块，用以表示 G_T的内部部分 K_T。为此，我们将逐步讲解算法3的每个步骤，说明其工作原理，并确认该算法能够成功构造所需模块。

1. 步骤1至4（算法3）：给定一个包含至少一个 K_T 作为子图的输入图 G_T，我们首先检查除 K_T 之外是否存在任何复杂三角形（使用 [24]中提出的算法）。如果 G_T 中存在此类复杂三角形，我们就应用复杂三角形消除算法（在 [24]中提出），通过引入附加顶点和边将其消除。此过程可确保从 G_T中移除所有复杂三角形（ K_T除外）：参见图34（a-d）。由此，我们得到一个除 K_T之外不含复杂三角形的修改后的图 G_T。

2. 步骤5至6（算法3）：为了修改剩余的 K_T 子图，我们首先通过插入一个新顶点 u，替换 K_T 的内部边（a， c）。为确保更新后的图仍保持三角剖分状态，我们引入额外的边，如图1所示。我们将此修改后的图记为 G^1_T。接下来，我们按照 [5], 中的描述，对 G^1_T应用四连通补全算法。最后，我们在顶点 N 和 S之间添加一条边，形成 G^1_T（4-连通图）：参见图34（e-g）。因此，始终可以从输入图 G^1_T构造出一个4-连通图 G^1_T。

3. 步骤7至17（算法3）：从之前获得的4-连通 PTG G^1_T开始，我们执行步骤（7–9），首先将 G^1_T中的每个顶点的 $chord\ value$ 赋值为0，并将 $status\ value$ 设为 $False$。随后，将顶点 W 标记为 v_1，将顶点 S 标记为 v_2，并将顶点 E 的访问值设为1。之后，使用步骤

（10–17），按照 [5] 中提出的方法，逐一为 G_T^1 的所有顶点分配规范序，并返回规范有序图 G_T^2：见图35、36。因此，我们始终可以为4-连通 PTG G_T^1构造规范有序图G_T^2。

4. 步骤18（算法3）：利用先前生成的规范有序图 G_T^2，我们现在继续执行 [5] 中讨论的算法，以构造 G_T^2 的正则边标记（REL，即 T_1 或 T_2），也就是 G_T^3：见图37。因此，总能从规范有序图 G_T^2 生成正则边标记 G_T^3。

5. 步骤19至24（算法3）：基于先前生成的 G_T^2 的正则边标记 G_T^3，接下来我们将按照 [5] 中概述的算法，利用 T_1 和 T_2 方向设计矩形平面布局（RFP） F_T'。获得 RFP F_T' 后，我们将应用函数 $Merge\ Rooms$。该函数的输入包括：规范有序图 G_T^2、生成的 RFP F_T'，以及额外节点集合（$Enodes_T$：这些是在 G_T 中引入的附加顶点，用于消除复杂三角形，但不包括 K_T）。为了在平面图 F_T 中构造一个 T 形模块，我们必须将额外模块 u 与 a 或 b 合并，使得模块 a 变为 T 形，或模块 c 变为 T 形（见图45：由于图 G_T^1 中顶点 d、u 和 e 的度为4，且 d 与 u 相邻，u 也与 e 相邻，因此这些模块可以在平面图中按两种不同构型排列。在平面图中进一步添加模块 a、f、b 和 c，意味着模块 a 将始终为 T 形。因此，我们总能通过将模块 u 与 a 合并，或将模块 u 与 c 合并，构造出 T 形模块。）

(a.) 步骤20至22（算法3）：现在，我们使用集合 $Enodes_T$，在F_T' 中合并额外模块。具体而言，对于 $Enodes_T$ 中的每个顶点，F_T'中对应的模块（即与 u_i关联的模块）都会与 F_T'内的模块 a_i 或 b_i 合并：见图38（a-b）。

(b.) 步骤23至24（算法3）：对于经过修改的 K_T 顶点，额外模块 u 将与 F_L' 中的模块 a 合并，或者与 F_L'中的模块 c 合并。最终，此过程会生成一个正交平面图 F_T，其中包含一个对应于子图 K_T的 T形模块：见图38（c）。

因此，对于由至少一个内部 K_T组成的 PTG G_T（V，E），算法3总会在正交平面图 F_T中生成一个 T形模块（对应于内部 K_T）。

□

7 算法中的时间复杂度分析

定理 3. 给定 $G_L(V, E)$，其中至少包含一个与 K_L 同构的内部子图，算法1 可在线性时间内生成正交平面图 F_L，并确保其中包含形状为 L 的模块（对应于子图 K_L），即 O(n)。

证明。对于至少包含一个与 K_L 同构的内部子图的 G_L，算法1 可生成正交平面图 F_L（对应于 G_L），该平面图必然包含一个与子图 K_L 对应的 L 形模块。我们将证明，该构造可在线性时间内完成，即 $O(n)$。我们将全面分析算法1的计算复杂度。

1. 步骤（1-4）：算法 1 [$\boldsymbol{O(n)}$]：给定一个包含至少一个内部 K_L 子图的输入图 G_L，我们可以在 $\boldsymbol{O(n)}$ 时间内，于 G_L 中识别任意一个这样的内部 K_L。具体做法是检测由三个顶点构成的三角形，其中包含一个度为 3 的内部顶点，方法见 [25]。找到这样的复杂三角形（K_4）后，我们将其指定为 K_L，这一过程需要线性时间，即 $O(n)$。K_L 的顶点按逆时针顺序排列如下：首先是 a，接着是 b，然后是 c，最后是 d，这也只需要 $\boldsymbol{O(1)}$ 时间。

如果图 G_L 除选定的 K_L外还包含复杂三角形，我们会采用 [24],提出的方法，该方法可在 $\boldsymbol{O(n)}$ 时间内运行，通过引入新顶点和边来消除这些复杂三角形。这确保 G_L 除 K_L外不含任何复杂三角形。因此，总计算复杂度为：步骤1至4（算法1）为 $[\boldsymbol{O(n)} + \boldsymbol{O(1)} + \boldsymbol{O(n)}$：$\boldsymbol{O(n)}]$,，这表明该预处理可在线性时间内高效执行。

2. 步骤（5-6）：算法 $\mathbf{1}$ [$\boldsymbol{O(n)}$]：为处理 G_L内剩余的 K_L 子图，我们首先移除子图 K_L的外部边（a，b），插入一个顶点 u，并插入四条新边，以确保该图经过三角剖分。此修改可在常数时间内完成，即 $\boldsymbol{O(1)}$，因为添加和删除都是常数时间操作。此步骤后得到的图称为 G_L^1。随后，我们应用四完备算法（Kant 和 He [5]提出的线性时间算法），该算法在 G_L^1中按顺序引入四个新顶点 E、N、S和 W，以及与之关联的新边（用于三角剖分）。此步骤在线性时间内完成，即 $\boldsymbol{O(n)}$。完成此过程后，向 G_L^1中添加一条新边（N，S）。由于插入边是一项 $\boldsymbol{O(1)}$ 操作，此步骤可在常数时间内完成。因此，算法1中步骤5至6的总体时间复杂度为 $[\mathbf{2}\boldsymbol{O(1)} + \boldsymbol{O(n)}\colon\ \boldsymbol{O(n)}]$。这表明，相对于输入规模 n，这些步骤可在线性时间内高效完成。

3. 第7步至第53步：算法 $\mathbf{1}$ [$\boldsymbol{O(n)}$]：从先前构建的 4-连通三角剖分图 G_L^1 出发，我们执行算法1的第7步至第53步，以得到其规范序图 G_L^2 （G_L^1）。

(a.) 第7步至第28步：算法 $\mathbf{1}$ [$\boldsymbol{O(n)}$]：首先，按照 [5], 中所述，使用函数 $CanonLabel$ 对 G_L^1 中的每个顶点进行规范排序（集合 $\{a, b, c, C_1, d, u\}$ 中的顶点除外）。该过程的时间复杂度为线性时间，即 $\boldsymbol{O(n)}$。这一规范排序过程会持续进行，直到仅剩下集合 $\{a, b, c, C_1, d, u\}$ 中的顶点符合排序条件。此时，按照预先设定的优先级顺序为这些顶点赋予标签，该顺序分为六个类别（A 至 F 类）。

(b.) 第29步至第53步：算法 $\mathbf{1}$ [$\boldsymbol{O(n)}$]：接下来，我们将按照定义的六种优先级排序，逐一尝试对集合 $\{a, b, c, C_1, d, u\}$ 中的顶点进行系统标记。对于每个类别，都会调用 $CanonLabel$ 函数，其时间复杂度为 $\boldsymbol{O(n)}$；同时还会检查一个优先级验证条件，记为 $PLabel(L)$，该检查可在常数时间内完成，即 $\boldsymbol{O(1)}$（因为检查该条件只需常数时间）。如果找到任何一种排序（A 至 F 类之一），则将所得有序图 G_L^2 送入正则边标记（REL）构造过程。因此，验证任何特定类别的排序需要 $\boldsymbol{O(n)}$ 时间。由于需要检查/评估六个类别，此阶段总共最多需要 $\mathbf{6}\boldsymbol{O(n)}$ 时间。

因此，第7步至第53步的总计算复杂度为 $[\boldsymbol{O(n)} + \mathbf{6}\boldsymbol{O(n)} + \boldsymbol{O(1)} = \boldsymbol{O(n)}]$,，这表明优先规范排序过程可以在线性时间内高效执行。

4. 步骤54：算法 $\mathbf{1}$ [$\boldsymbol{O(n)}$]：从规范有序图 G_L^2 开始，使用算法2构造 G_L^2 的正则边标记（REL，即 T_1 和 T_2）。

(a.) 步骤（1–10）：算法 $\mathbf{2}$ [$\boldsymbol{O(n)}$]：首先，按照 [9] 中详述的方法，为输入图 G_L^2 构造 REL（即 T_1 和 T_2）。该构造过程的时间复杂度为线性，即$\boldsymbol{O(n)}$，如 [9]中所述。因此，算法2的步骤1–10需要线性时间。

(b.) 步骤（11–23）：算法 $\mathbf{2}$ [$\boldsymbol{O(1)}$]：下一步是根据上述生成的 REL（T_1 和 T_2）选择并执行适当的操作。具体而言，算法将顶点 u 与顶点 a 或顶点 b 合并（返回值为 $r = 1$ 或 2），或者先执行边翻转，再将 u 与a 合并（返回值为 $r = 1$）。这些操作均可在常数时间内完成（因为检查条件需要常数时间，而改变边的方向，即 T_1 或 T_2，也需要常数时间），即 $\boldsymbol{O(1)}$。

因此，　　算法1第45步的计算复杂度为 $[\boldsymbol{O(n)} + \boldsymbol{O(1)} = \boldsymbol{O(n)}]$,，对应第　　ar.

5. 步骤（55-63）：算法 $\mathbf{1}$ [$\boldsymbol{O(n)}$]：

(a.) 步骤 55：算法 $\mathbf{1}$ [$\boldsymbol{O(n)}$]：利用正则边标记，即 G_L^3 （由 T_1 和 T_2表示），采用 Bhasker 和 Sahni [9],中描述的方法构造矩形平面图 F_L' 。该方法的时间复杂度为线性时间，$\boldsymbol{O(n)}$。

(b.) 步骤 (56-58)：算法 **1** [$O(r)$]：之后，我们将通过调用函数 $Merge\ Room$，在 F'_L 中合并额外模块。对于集合 $Enodes_L$ 中的每个顶点，若 $Enodes_L$ 的基数 r 小于 G^1_L中的顶点总数 n，则按需合并 F'_L 中的相应模块。由于每次合并操作都能在常数时间内完成，因此此步骤的总时间复杂度为$O(r)$，小于 $O(n)$。

(c.) 步骤（59–63）：算法 **1** [$O(1)$]：最后，对于与 K_L 相关联的顶点，算法将模块 u 与 a 或 b 合并，此操作可在常数时间内完成，即 $O(1)$。

总而言之，步骤 55 至 63（算法1）的整体计算复杂度为 $O(r)+O(1)+O(n)$:$O(n)$，可简化为 $O(n)$，因为 $r < n$。这表明，这些步骤能够根据输入图的规模在线性时间内高效执行。

因此，算法1的时间复杂度为线性，因为其五个核心阶段各自均可在 $O(n)$ 时间内运行。合并后，总时间为（$O(n) + O(n) + ... + O(n)$）（共五次），可简化为整体时间复杂度 $O(n)$。因此，给定一个 $G_L(V, E)$，其中至少包含一个与 K_L 同构的内部子图，算法1能够在线性时间内生成一个正交平面图 F_L，该图必然包含一个与子图 K_L 对应的 L 形模块，即 $O(n)$，其中 n 是 G_L 中的顶点数。

□

定理4. 给定 $G_T(V, E)$，其中至少包含一个与 K_T 同构的内部子图，算法3在线性时间内（即 O(n)）生成正交平面图 F_T，并确保其中包含形状为 T 的模块（对应子图 K_T）。

证明. 对于包含至少一个与 K_L 同构的内部子图的 G_T，算法3生成正交平面图 F_T（对应于 G_T），其中必然包含一个形状为 T 的模块，对应子图 K_T。我们将证明该构造可在线性时间内完成，即 $O(n)$。我们将全面分析算法3的计算复杂度。

1. 步骤 (1-4)：算法 **3** [$O(n)$]：给定一个包含至少一个内部 K_T 子图的输入图 G_T，我们可以在 G_T 内识别出任意一个这样的内部 K_T，所需时间为 $O(n)$。具体做法是检测两个共享一条公共边的三角形，其中每个三角形均由三个顶点构成，且包含一个度为 3 的内部顶点；该方法见 [25]。找到这样的复杂三角形后，我们将其指定为 K_T；这一过程需要线性时间，即 $O(n)$。K_T 的顶点按逆时针顺序排列如下：首先是 a，然后是 b，接着是 c、d、e，最后是 f，这一操作也只需 $O(1)$ 时间。
如果图 G_T 除选定的 K_T 外还包含复杂三角形，我们将采用 [24], 提出的方法，该方法以 $O(n)$ 时间运行，通过引入新顶点和边来消除这些复杂三角形。这确保 G_T 除 K_T 外不含任何复杂三角形。因此，总计算复杂度为：步骤 1 至 4（算法3）为 [$2O(n) + O(1)$： $O(n)$],，这表明该预处理可以在线性时间内高效完成。

2. 步骤（5-6）：算法 **3** [$O(n)$]：为处理 G_T 中剩余的 K_T 子图，我们首先移除子图 K_T 的内部边（a，c），插入一个顶点 u，并插入四条新边，以确保该图经过三角剖分。由于添加和删除都是常数时间操作，这一修改可在常数时间内完成，即 $O(1)$。此步骤完成后的图称为 G^1_T。随后，我们应用四完备算法（由 G. Kant 和 X. He 提出的线性时间算法 [5]），该算法在 G^1_T 中引入四个标记为 E、N、S 和 W 的新顶点，以及与之相关的新边（用于三角剖分）。此步骤在线性时间内完成，即 $O(n)$。完成该过程后，在 G^1_T 中添加一条新边（N，S）。由于插入一条边属于 $O(1)$ 操作，此步骤可在常数时间内完成。因此，步骤 5 至 9（算法3）的总体时间复杂度为 [$2O(1) + O(n)$： $O(n)$]。这表明，相对于输入规模 n，这些步骤可以在线性时间内高效完成。

3. 步骤（7–17）：算法 **3** [$O(n)$]：从之前获得的 4-连通 PTG G^1_T 开始，我们执行步骤（7–9），将每个顶点的 $chord\ value$ 初始赋值为 0，并将 $status\ value$ 赋值为 $False$ G^1_T

这需要常数时间，即 $\boldsymbol{O(1)}$。之后，将顶点 W 标记为 v_1，将顶点 S 标记为 v_2，并将顶点 E 的访问值设为 1，这同样需要 $\boldsymbol{O(1)}$。之后，使用步骤（10–17），我们逐一为 G_T^1 的所有顶点赋予规范序，采用 [5] 中提出的方法，该方法需要线性时间，即 $\boldsymbol{O(n)}$；同时确保该过程符合定义4（参见术语第2节），并返回规范有序图 G_T^2。
因此，步骤 7 至 17 的总计算复杂度为 $[\boldsymbol{2O(1)+O(n)}: \boldsymbol{O(n)}],$，这表明规范排序过程可以在线性时间内高效执行。

4. 步骤 18：算法 **3** $[\boldsymbol{O(n)}]$: 利用先前生成的规范有序图 G_T^2，我们现在按照 [5] 中讨论的算法，为 G_T^2（即 G_T^3）构造正则边标记（REL，即 T_1 或 T_2）。如 [5] 所述，该构造过程在线性时间内完成，即 $\boldsymbol{O(n)}$。因此，步骤 18 的总计算复杂度为 $[\boldsymbol{O(n)}],$，属于线性复杂度。

5. 步骤（19–24）：算法 **3** $[\boldsymbol{O(n)}]$:

（a.）步骤 19：算法 **3** $[\boldsymbol{O(n)}]$: 利用正则边标记，即 G_T^3（由 T_1 和 T_2 表示），采用 Bhasker 和 Sahni [9], 中描述的方法构造矩形平面图 F_T'，其时间复杂度为线性，即 $\boldsymbol{O(n)}$。

(b.) 步骤（20-22）：算法 **3** $[\boldsymbol{O(r)}]$: 之后，我们将调用函数 $Merge\ Room$，合并 F_T' 中的额外模块。对于集合 $Enodes_T$ 中满足 $Enodes_T$ 的基数 r 小于 G_T^1中顶点总数 n的每个顶点，都会按需合并 F_T'中对应的模块。由于每次合并操作都可以在常数时间内完成，因此此步骤的累计时间复杂度为 $\boldsymbol{O(r)}$，小于 $\boldsymbol{O(n)}$。

(c.) 步骤（23-24）：算法 **3** $[\boldsymbol{O(1)}]$: 最后，对于与 K_T相关联的顶点，算法会将模块 u 与 a 或 c合并；此操作可在常数时间内完成，即 $\boldsymbol{O(1)}$。

总之，步骤19至24（算法3）的整体计算复杂度为 $\boldsymbol{O(n)+O(1)+O(r)}:\boldsymbol{O(n)}$，可简化为 $[\boldsymbol{O(n)}],$，因为 $r < n$。这表明，相对于输入图的规模，这些步骤可以在线性时间内高效执行。

因此，算法3的时间复杂度为线性，因为其五个核心阶段各自的运行时间均为$O(n)$。合并后，总计为（$\boldsymbol{O(n)} + \boldsymbol{O(n)} + \boldsymbol{.....} + \boldsymbol{O(n)}$）（共五次），可简化为总体时间复杂度 $O(n)$。因此，对于给定的 $G_T(V,E)$，若其中至少包含一个与 K_T同构的内部子图，算法3便能在线性时间内（即 O(n)）生成一个正交平面图 F_T，该图必然包含一个与子图 K_T 对应的T形模块，其中 n 是 G_T中的顶点数。

□

8 结论与未来工作

本文提出了一种结构化方法，详见算法1和算法3，分别用于在最终平面图 F_L 和 F_T中嵌入 L形和T形模块。每种构造都从特定的输入图开始：对于 L形情形，输入为包含至少一个内部复杂三角形（K_L）的 PTG G_L；对于 T形情形，输入为包含至少一个内部子图（K_T）的 PTGG_T。整个过程分为四个关键阶段：首先，通过引入辅助顶点和边，将输入 PTG 转换为 4-连通三角剖分图。接着，对修改后的图应用规范序。然后，利用所得的规范序构造矩形平面图。最后，将代表新增顶点的辅助模块合并，形成正交平面图，其中包含与原图 G_L 或 G_T相对应的 L形或 T形模块。我们提出的用于生成 L形和 T形模块的方法具有一个重要特点：模块本身的结构非平凡，也就是说，最终平面图中所需的模块（ L 形或 T形）无法通过收缩或拉伸其模块的墙而简化为更简单的形式。要确保这种结构唯一性，规范序方法的应用至关重要。

在后续研究中，我们计划拓展现有框架，将更多非矩形模块类型纳入平面布局过程，具体包括十字形、阶梯形和 Z 形。这需要确定允许将这些复杂形状纳入平面图及其对应图中的结构判据，然后设计相应算法以实现其生成。

9 声明

利益冲突：作者声明，与本次投稿发表的作品直接或间接相关的财务或非财务利益均不存在利益冲突。准备本手稿期间未获得任何资金、资助或其他支持。

数据可用性：有关数据可用性的咨询应联系作者。

参考文献

1. Xin He. 一种用于求解平面三角图矩形对偶的高效并行算法。 *Algorithmica*, 13(6):553– 572, 1995年。2. Y. Sun 和 M. Sarrafzadeh. 基于图对偶化的平面布局规划：L形模块。 *Algorithmica*, 10:429–456, 1993年。3. Krishnendra Shekhawat、Rohit Lohani、Chirag Dasannacharya、Sumit Bisht 和 Sujay Rastogi. 自动生成带非矩形房间的平面图。 *Graphical Models*, 127:101175, 2023。4. Krzysztof Ko´zmi´nski 和 Edwin Kinnen. 平面图的矩形对偶。 *Networks*, 15(2):145–157, 1985。5. Goos Kant 和 Xin He. 4-连通平面图的正则边标记及其在图绘制问题中的应用。*Theoretical Computer Science*, 172(1-2):175–193, 1997。6. I Rinsma. 平面布局图的存在性定理。 *Bulletin of the Australian Mathematical Society*, 37(3):473–475, 1988年。7. Peter Hirsch Levin. 利用图确定建筑物的最优布局。 *The Architects' Journal*, 7:809–815, 1964。8. John Grason. 一种利用图论进行计算机空间规划的方法。In *Proceedings of the 8th Design automation workshop* 170 178, 1971, 页码 – 。

9. Jayaram Bhasker 和 Sartaj Sahni. 求解平面三角剖分图矩形对偶的线性算法。 *Algorithmica* 3 2 247 278, 1988, ( ): – 。

10. Xiao-Yu Wang、Yin Yang 和 Kang Zhang. 基于图变换的平面图定制与生成。*Automation in Construction*, 94:405–416, 2018年。11. Maciej Nisztuk 和 Pawe l B Myszkowski. 应用于自动平面图生成的混合进化算法。 *International Journal of Architectural Computing*, 17(3):260–283, 2019年。12. Feng Shi、Ranjith K Soman、Ji Han 和 Jennifer K Whyte. 使用蒙特卡洛树搜索解决矩形平面图中的邻接约束。 *Automation in Construction*, 115:103187, 2020年。13. Nitant Upasani、Krishnendra Shekhawat 和 Garv Sachdeva. 自动生成带尺寸标注的矩形平面布局图。 *Automation in Construction*, 113:103149, 2020年。14. Ruizhen Hu、Zeyu Huang、Yuhan Tang、Oliver Van Kaick、Hao Zhang 和 Hui Huang. Graph2plan：从布局图学习平面图生成。 *ACM Transactions on Graphics (TOG)*, 39(4):118–1, 2020年。15. Lufeng Wang、Jiepeng Liu、Yan Zeng、Guozhong Cheng、Huifeng Hu、Jiahao Hu 和 Xuesi Huang. 利用深度学习和图算法自动生成建筑布局。 *Automation in Construction*, 154:105036, 2023。16. Jiepeng Liu、Zijin Qiu、Lufeng Wang、Pengkun Liu、Guozhong Cheng 和 Yan Chen. 基于图约束生成对抗网络的模块化高层住宅建筑智能平面图设计。 *Automation in Construction* 159 105264, 2024, : 。17.

Shuji Tsukiyama、Keiichi Koike 和 Isao Shirakawa. 一种消除极大平面图中所有复杂三角形、用于 VLSI 平面布局的算法。In *Algorithmic Aspects Of VLSI Layout*, 页码 321–335。World Scientific, 1993年。18. Xin He. 关于平面图的平面布局。 *SIAM Journal on Computing*, 28(6):2150–2167, 1999。19. Maciej Kurowski. 简单高效的平面布局规划。 *Information processing letters*, 86(3):113–119, 2003年。20. Chien-Chih Liao、Hsueh-I Lu 和 Hsu-Chun Yen. 基于有序生成树的紧凑平面布局规划。 *Journal of Algorithms* 48 2 441 451, 2003, ( ): – 。

21. Muhammad Jawaherul Alam、Therese Biedl、Stefan Felsner、Andreas Gerasch、Michael Kaufmann 和 Stephen G Kobourov. 无孔直线型比例接触图表示的线性时间算法。In *Algorithms and Computation: 22nd International Symposium ISAAC 2011 Yokohama Japan December 5-8 2011. Proceedings 22* , , , , , , 页码 281–291。Springer, 2011。22. Christian A Duncan、Emden R Gansner、YF Hu、Michael Kaufmann 和 Stephen G Kobourov. 平面图的最优多边形表示。 *Algorithmica*, 63:672–691, 2012年。23. K. Shekhawat 和 Jose P. Duarte. 直线型平面布局图。 *Communications in Computer and Information Science*, 724 395 411, 2017 : – 。

24. S Roy、S Bandyopadhyay 和 U Maulik。关于一般邻接图中无权复杂三角形消除（cte）问题的 NP 完全性证明。

IEE Proceedings-Computers and Digital Techniques 148 6 238 244, , ( ): – . 2001

25. Donald B Johnson。寻找有向图中的所有初等回路。

SIAM Journal on Computing 4 1 77 84, , ( ): – . 1975