Connect Points - 解题报告

龚林源

1 Description

判断一个图是不是最大平面图。(不考虑重边和自环)

平面图 可以画在平面上并且使得不同的边可以互不交叠的图

最大平面图 加上任意一条边就不可能构成平面图的平面图

2 Constraints

$1 \leq T \leq 10,\ 1 \leq N \leq 70000,\ 1 \leq M \leq 300000$

3 Tags

- 最大平面图

4 Solution

4.1 定义和引理

- $G = (V, E)$——令 G 为一个平面图，点集为 V，边集为 E
- 平面图的面——平面图上的封闭区域叫做内部面，外部的区域叫做外部面
- $INC_v,\ v \in V$ ——在平面图 G 中与 v 邻接的节点按逆时针顺序排列组成的序列
- K_4 ——四个点组成的完全图，是最大平面图
- 外平面图——所有点都在同一个面上的平面图

引理 1. 最大平面图是三连通的。(删除任意两个点，图仍然连通) 所以有至少 4 个点的最大平面图不存在度数小于等于 2 的点。

引理 2. 一个图是最大平面图，当且仅当它的每个面 都是三角形。

引理 3. 一个平面图是最大平面图，当且仅当 $|E| = 3|V| - 6$

引理 4. 最大平面图有惟一的球面嵌入。在选定一个外部面之后，最大平面图的平面嵌入也是惟一的。

引理 5. 外平面图有至少两个度数不超过 2 的点。

以上五条引理的证明都可以在参考资料 [2] 中找到。

定义 1. 平面图 $G = (V, E)$ 中的一个点 v 是“小的”，当且仅当它的度数 $degree(v) < 18$，否则它是“大的”。一个点 $v \in V$ 是“可约的”当且仅当它满足以下几个条件之一：

- $degree(v) \leq 3$
- $degree(v) = 4$ 且 v 至少有两个小的点与它有边
- $degree(v) = 5$ 且 v 至少有四个小的点与它有边

引理 6. 每个点数 $|V| \geq 4$ 的平面图 $G = (V, E)$ 至少有四个点是可约的。

证明. 当图增加若干条边时，可约的点显然不会增加，因此只要证明这个引理在最大平面图上成立即可。

根据引理 2，一个最大平面图 $G = (V, E)$ 满足 $|E| = 3|V| - 6$。

令 n_i 表示度数为 i 的点的个数，令 r_4 和 r_5 分别表示度数为 4 和度数为 5 的可约点数目。

根据引理 1，有 $n_0 = n_1 = n_2 = 0$。

$$\sum_{i \geq 3} i n_i = 2|E| = 6|V| - 12 = -12 + \sum_{i \geq 3} 6 n_i \tag{1}$$

$$\Longrightarrow \sum_{i \geq 3} (i - 6) n_i = -12 \tag{2}$$

$$\Longrightarrow 2n_4 + n_5 = 12 - n_3 + \sum_{i \geq 7} (i - 6) n_i \tag{3}$$

统计度数为 4 或 5 的不可约点的和大的点之间的边，可以得到：

$$3(n_4 - r_4) + 2(n_5 - r_5) \leq \sum_{i \geq 18} i n_i \tag{4}$$

根据 (4) 式，可以得到：

$$\frac{2}{3} \sum_{i \geq 18} i n_i \geq (2n_4 + n_5) - 2r_4 - \frac{4}{3} r_5 + \frac{1}{3} n_5 \tag{5}$$

根据 (3) 式，(5) 式不等号的右边部分等于：

$$12 - 3n_3 - 2r_4 - \frac{4}{3} r_5 + \frac{1}{3} n_5 + \sum_{i \geq 7} (i - 6) n_i \tag{6}$$

根据 (5)(6) 可得：

$$\sum_{i \geq 18} (2i) n_i - \sum_{i \geq 7} (3i - 18) n_i \geq 36 - 9n_3 - 6r_4 - 4r_5 + n_5 \tag{7}$$

又因为：

$$\sum_{i \geq 18} (2i) n_i - \sum_{i \geq 7} (3i - 18) n_i = \sum_{i \geq 18} (18 - i) n_i - \sum_{7 \leq i < 18} (3i - 18) n_i \leq 0 \tag{8}$$

根据 (7)(8) 可得：

$$9n_3 + 6r_4 + 4r_5 \geq 36 + n_5 \tag{9}$$

$$\Longrightarrow n_3 + r_4 + r_5 \geq n_3 + \frac{2}{3}r_4 + \frac{4}{9}r_5 \geq 4 + \frac{1}{9}n_5 \geq 4 \tag{10}$$

$n_3 + r_4 + r_5 \geq 4$，证毕。

引理 7. 令 v 为最大平面图 $G = (V, E), |V| \geq 4$ 上的任意一点，令 $N(v)$ 代表 v 的邻接节点集合。至少存在两个 $u \in N(v)$，满足 $|N(u) \cap N(v)| = 2$

证明. 令 $v_1, v_2, ..., v_k$ 为 G 的某一固定的平面嵌入中 $N(v)$ 按逆时针排列的序列。因为 G 是最大平面图，根据引理 2，每个面都是三角形。所以对于 $1 \leq i \leq k$，v_i 和 $v_{(i \bmod k)+1}$ 是邻接的。所以保留 $N(v)$ 中的点和这些点之间的连边组成的子图是一个外平面图。根据引理 5，度数不超过 2 的点至少有 2 个，这些点 u 满足 $|N(u) \cap N(v)| = 2$，证毕。

4.2 用到的数据结构

首先我们需要一个数据结构来存图，使得可以在常数时间内加一条边，同时遍历一个点的所有邻居可以在每个邻居 $O(1)$ 时间内完成；可以用邻接表 来实现。

引理 8. 对于图 $G = (V, E)$，存在一个数据结构 $vertexSet\ N$ 可以在常数时间内完成以下几种操作 (v 是任意给定点)：

a) 插入 v

b) 删去 v

c) 判断 $v \in N$ 是否为真

d) 判断 $N = \emptyset$ 是否为真

e) 弹出任意一个满足 $u \in N$ 的点 u

f) 遍历 N ($\forall u \in N$，都是常数时间)

g) 清空 N ($\forall u \in N$，都是常数时间)

证明. 可以用一个栈加一个 $hash$ 表来实现 $vertexSet\ N$。用栈存储所有的元素，用 $hash$ 表存储每个元素在栈中的位置。a, d, e, f, g 都是栈的基本操作，同时在常数时间内维护 $hash$ 表中的有关信息。c 操作可以直接查询 $hash$ 表来实现。b 操作只要在 $hash$ 表中查询 v 的位置，然后把 v 与栈顶元素交换，再把栈顶元素弹出即可。

然后我们还需要一个数据结构来存 INC_v，可以在每个元素常数时间内完成清空、前驱、后继、在某个元素后加点、删除某个元素、判断某个元素是否存在的操作。前三种操作可以使用双向链表来实现。后三种操作可以使用上面提到的 hash 表类似的技巧来实现。

定义 2. 对一个图 $G = (V, E)$ 定义映射 $COMPACT : V \to \{E' | E' \subseteq E\}$，使得 $COMPACT_v \subseteq INC_v$ 对于每个点 v 满足以下性质：

- $\forall e = (u, v) \in E : e \in COMPACT_u \cup COMPACT_v$

- $\forall e = (u, v) \in E : COMPACT_u \cap COMPACT_v = \emptyset$

$COMPACT$ 是 k-限定的（$k > 0$），当且仅当 $\forall v \in V$，$COMPACT_v$ 至多有 k 条边。

引理 9. 平面图有 5-限定的 $COMPACT$。

证明. 令 $G = (V, E)$ 为一平面图。根据引理 3，有 $|E| \leq 3|V| - 6$。所以 $\sum_{v \in V} degree(v) = 2|E| \leq 2(3|V| - 6) < 6|V|$。所以，$\exists v \in V,\ degree \leq 5$。现在令图 $H = G$，然后重复以下操作直到 $V(H) = \emptyset$。

- 找到一个 $v \in V(H) \wedge degree(v) \leq 5$
- 令 $COMPACT_v = INC_v = e \in E(H) | v \in e$
- 从 H 中移除点 v

这个过程为 G 定义了一个 5-限定的 $COMPACT$ 映射。

推论. 给定平面图 $G = (V, E)$ 和 G 的一个 5-限定的 $COMPACT$，给定 u, v，可以在常数时间内判断 $(u, v) \in E$ 是否为真。

证明. 令 $L = COMPACT_u \cup COMPACT_v$。如果 $(u, v) \in E$ 那么一定有 $(u, v) \in L$。同时，$|L| \leq 10 = O(1)$。

4.3 约去操作

定义 3. 这样定义最大平面图 $G = (V, E)$ 中的一个可约点 v（参见定义 1）的约去操作：

- 如果 $degree(v) = 3$，把 v（和 v 的邻接边一起）从图 G 中删去
- 如果 $degree(v) \in \{4, 5\}$，令 $w \in N(v)$ 是一个满足 $|N(v) \cap N(w)| = 2$ 的点（引理 7）。把 v 从图 G 中删去，然后 $\forall x \in (N(v) - (\{w\} \cup N(w)))$，加一条 (w, x) 边。

引理 10. 令 $G = (V, E)$ 是一个满足 $|V| > 4$ 的最大平面图，令 $v \in V$ 是任一可约点。对 v 进行约去操作后，图 G 仍是最大平面图。

证明. 因为 G 是最大平面图所以每一个与 v 相邻的面都是三角形（引理 2）。在删去 v 后这些面连成了一个边数等于 $degree(v)$ 的面。当 $degree(v) = 3$ 时这个面就是三角形；当 $degree(v) > 3$ 时加边使得这个面分成了 2 或 3 个三角形。根据引理 2，这个图仍是最大平面图。

引理 11. 一次约去操作所需时间是 $O(1)$ 的。

证明. 假设对 v 进行了一次约去操作。

- 若 $degree(v) = 3$，显然成立。
- 若 $degree(v) = 5$，我们可以在 $N(v)$ 中找到两个小的点（定义 1），因为根据可约点的定义至少有 4 个这样的点，又因为两个点和 v 有两个共同邻居（引理 7），根据抽屉原理至少有一个小的点与 v 有两个共同邻居，所以可以在至多 $5 + 4 * 18 = O(1)$ 的时间内找到 w。

- 若 $degree(v) = 4$，我们检查每个（至少两个）小的相邻节点，判断它是否与 v 有两个共同邻居。如果有，那就在常数时间内找到了 w。如果没有，那么只要从剩下两个大的邻居中任选一个即可。根据引理 2，那两个小点与 v 的共同邻居数目大于等于 2，也就是 3。假设那两个大点之间有边，v 和 v 的四个邻居组成了 K_5 完全图，不满足平面图性质；所以假设不成立，那两个大点之间无边相连，因此那两个大点都恰与 v 的两个小的邻居相连。

定义 4. 令 $G' = (V', E')$ 是最大平面图 $G = (V, E)$ 把点 u 约去后的图（$V' = V - \{u\}$）。G' 的 5-限定 $COMPACT$ 对于每个 $v \in V'$ 分别是 $COMPACT_v$。

对 u 的反约去 操作是把 u 从 G 中约去的逆操作，定义为：

- 若 $k = 3$，x_0, y, z 为 u 在 G 中的邻居。在常数时间内在 G' 中判断是否有 $(x_0, y), (y, z), (z, x_0)$ 边（参见引理 9 的推论）。若 (x_0, y) 是 INC_{x_0} 中 (z, x_0) 的后继，令 $x_1 = y,\ x_2 = z$；否则令 $x_1 = z,\ x_2 = y$。

- 若 $k = 4$，(x_0, x_2) 为约去过程中加的边。令 (x_0, x_1) 为 (x_0, x_2) 在 INC_{x_0} 的后继，令 (x_0, x_3) 是 (x_0, x_2) 在 INC_{x_0} 的前驱。从 G'、$COMPACT_{x_0}$ 以及 $COMPACT_{x_2}$ 中删去 (x_0, x_2)。

- 若 $k = 5$，$(x_0, y),\ (x_0, z)$ 分别是约去过程中加的第一、第二条边。若 (x_0, y) 是 INC_{x_0} 中 (x_0, z) 的后继，令 $x_2 = y,\ x_3 = z$，否则令 $x_2 = z,\ x_3 = y$。令 (x_0, x_1) 为 INC_{x_0} 中 (x_0, x_2) 的后继、(x_0, x_4) 为 INC_{x_0} 中 (x_0, x_3) 的前驱。从 G' 中删去 $(x_0, x_2), (x_0, x_3)$。同时从相应节点的 $COMPACT$ 中也删去相应的边。

同理，其它所有的边都可以在常数时间内找到。

因此对于点 u 的反约去操作也是可以在常数时间内完成的。并且在执行完反约去操作之后，会得到原图 G 和它的 $COMPACT$。

4.4 最大平面图的嵌入

本文的最大平面图的嵌入算法分为三个步骤。在第一步中，$G = (V, E)$ 经过若干次约去操作变成了 K_4；在第二步中，得到 K_4 的嵌入；在第三步中，执行若干次对应的反约去操作，得到原图 G 的嵌入。

可约点的集合使用引理 8 中介绍的名为 $vertexSet\ REDUCIBLE$ 的数据结构来维护。预处理可以在 $O(|V| + |E|)$ 完成；下面将要证明 $REDUCIBLE$ 在每次约去操作之后的维护可以在 $O(1)$ 完成：

引理 12. 令 $G = (V, E)$ 是一个最大平面图，令 $REDUCIBLE$ 是 G 的可约点集合，令 v 为任一满足 $v \in REDUCIBLE$ 的点。$G' = (V', E')$ 是 G 约去 v 后得到的图，其中 $V' = V - v$。每次约去操作之后对 $REDUCIBLE$ 的更新至多需要 $O(1)$ 的时间。

证明. 更新一个点 x 操作可以用以下伪代码表示：

```
1: procedure UPDATE(REDUCIBLE, x)
2:     if x 是可约的 then
3:         REDUCIBLE. 插入 (x)
4:     else
5:         REDUCIBLE. 删除 (x)
6:     end if
```

```
7: end procedure
```

根据引理 8，$vertexSet$ 的单次插入操作和单次删除操作都是 $O(1)$ 的。根据可约点的定义（定义 1），判断一个点 x 是否是可约的也可以在 $O(1)$ 内完成。所以上面的 $update$ 过程至多需要 $O(1)$ 的时间。接下来我们还要证明每次约去操作后需要 $update$ 的次数也是常数级别的。

当约去 v 后，只有 $N(v)$ 中的点的度数会发生改变，这个改变会造成这些影响：

- 若 $x \in N(v)$，原来 $degree(x) > 5$，现在 $degree'(x) \leq 5$。那么 x 可能变得可约了
- 若 $x \in N(v)$，原来 $degree(x) \leq 5$，现在 $degree'(x) > 5$。那么 x 可能变得不可约了
- 若 $x \in N(v)$，原来 $degree(x) > 18$，现在 $degree'(x) \leq 18$。那么 x 的所有度数不超过 5 的邻居可能变得可约了
- 若 $x \in N(v)$，原来 $degree(x) \leq 18$，现在 $degree'(x) > 18$。那么 x 的所有度数不超过 5 的邻居可能变得不可约了

因此约去 v 后只需执行以下操作就可以正确维护 $REDUCIBLE$ 了：

```
1: procedure UPDATE_LOCAL(REDUCIBLE, v)
2:     for all x ∈ N(v) do
3:         if degree(x) ≤ 18 ∨ degree'(x) ≤ 18 then
4:             update(REDUCIBLE, x)
5:             for all y ∈ N(x) do
6:                 if degree'(y) ≤ 5 then
7:                     update(REDUCIBLE, y)
8:                 end if
9:             end for
10:        end if
11:    end for
12: end procedure
```

因为原来的 v 可约，根据定义 1，外层循环至多执行 5 次；因为 $degree'(x) \leq 18$，内层循环至多执行 18 次；我们又证明过单次 $update$ 操作是 $O(1)$ 的。因此，$update\_local$ 所需时间至多为 $5 \times 18 \times O(1) = O(1)$。证毕。

综上所述，整个算法可以用以下伪代码表示：

```
1: procedure MAXIMAL_PLANAR_EMBEDDING(G)
2:     stack S                                          ▷ 初始为空
3:     PHASE_I(G, S)
4:     PHASE_II: 重新排列 G = K_4 的 INC_v, ∀v ∈ V
5:     PHASE_III(G, S)
6: end procedure
```

先来看第一步：

```
1: procedure PHASE_I(G, S)
2:     vertexSet REDUCIBLE                              ▷ 初始为空
```

```
    for all v ∈ G do                                              ▷ 初始化
        update(REDUCIBLE, v)
    end for
    while |V| > 4 do                                              ▷ 主循环
        v ← REDUCIBLE. 弹出 ()
        k ← degree(v)
        for all u ∈ N(v) do
            删除 (v, u) 边
            S.push(v,u)
        end for
        删去 v
        S.push(v)
        if k ≥ 4 then
            找到满足 |N(v) ∩ N(w)| = 2 的 w
        end if
        if k = 4 then
            x ← H − (N(w) ∪ {w}) 中的唯一元素
            加入 (w, x) 边
            S.push(w,x)
        end if
        if k = 5 then
            x, y ← H − (N(w) ∪ {w}) 中恰好存在的两个元素
            加入 (w, x) 边
            S.push(w,x)
            加入 (w, y) 边
            S.push(w,y)
        end if
        update_local(REDUCIBLE, v)
        S.push(k)
    end while
end procedure
```

引理 13. $PHASE\_I(G, S)$ 对于完全平面图 $G = (V, E)$ 的复杂度是 $O(|V|)$ 的。

证明. $PHASE\_I$ 分为初始化和主循环两部分。

- 根据引理 12 的前半部分的证明，前面的初始化部分需要 $O(|V|)$ 的时间
- 主循环的次数为 $|V| - 4$
- 根据引理 11 和引理 12，一次主循环内的所有操作只需 $O(1)$ 的时间

综上所述，总复杂度为 $O(|V|)$。

整个算法的第二步是 $O(1)$ 的，十分显然。接下来我们看第三步：

```
procedure PHASE_III(G, S)
    edge h_1, h_2, h_3, h_4, h_5
    for all (x, y) ∈ G do                                         ▷ 初始化
        COMPACT_x. 插入 ((x, y))
    end for
```

```
    while ¬S.empty? do                                          ▷ 主循环
        k ← S.pop()
        if k = 3 then
            v ← S.pop()
            h_1 ← S.pop()
            h_2 ← S.pop()
            h_3 ← S.pop()
            x_0 ← G_opposite(v, h_1)        ▷ 把 x_0 赋为 h_1 边上 v 另一边的点
            y ← G_opposite(v, h_2)                              ▷ 同上
            z ← G_opposite(v, h_3)
        end if
        if k = 4 then
            (x_0, x_2) ← S.pop()
            v ← S.pop()
            h_1 ← S.pop()
            h_2 ← S.pop()
            h_3 ← S.pop()
            h_4 ← S.pop()
        end if
        if k = 5 then
            edgee_2 ← S.pop()
            edgee_1 ← S.pop()
            v ← S.pop()
            h_1 ← S.pop()
            h_2 ← S.pop()
            h_3 ← S.pop()
            h_4 ← S.pop()
            h_5 ← S.pop()
            y ← G_opposite(x_0, e_1)
            z ← G_opposite(x_0, e_2)
        end if
        做定义 4 中定义的反约去操作
    end while
end procedure
```

引理 14. $PHASE\_I(G,S)$ 对于完全平面图 $G=(V,E)$ 的复杂度是 $O(|V|)$ 的。

证明. $PHASE\_III$ 分为初始化和主循环两部分。

- 初始化所需时间为 $O(|E|)=O(3|V|-6)=O(|V|)$
- 根据 $PHASE\_I$，主循环有 $|V|-4$ 次
- 根据定义 4，反约去操作所需时间和约去操作的所需时间都是 $O(1)$

综上所述，总复杂度为 $O(|V|)$。

根据定理 13 和定理 14，整个算法的复杂度为 $O(|V|)$。因此我们得到了最大平面图嵌入的线性时间算法。

4.5 最大平面图检验

我们尝试通过上文所介绍的方法来判断一个一般无向图 $G=(V,E)$ 是否是最大平面图。在以下这些情况下，G 不是简单无向图，算法直接结束返回假：

1) 算法最开始：
   a) $|E| \neq 3|V|-6$
   b) G 有重边、自环
2) $PHASE\_I$ 的主循环中：
   a) 找不到可约点
   b) $\forall v, INC_v$ 中有重复元素
   c) 找不到满足 $|N(v) \cap N(w)|=2$ 的 w
3) $PHASE\_I$ 结束后：
   a) G 有重边
4) $PHASE\_III$ 主循环中 $k=3$ 的情况：
   a) $(x_0,y),(y,z),(z,x_0)$ 不在同一面上
5) $PHASE\_III$ 主循环中 $k \in \{3,4,5\}$ 的情况：
   a) $\exists e \in \{e_1,e_2,e_3,e_4,e_5\},\ G\_opposite(v,e) \notin \{x_0,x_1,x_2,x_3,x_4\}$

加上这些判断，原算法变成了一个最大平面图检验算法。如果 $G=(V,E)$ 是最大平面图，那么一定不会出现上面描述的异常状况。所以出现以上几种情况的图一定不是最大平面图。

引理 15. 若 $G=(V,E)$ 不会让该检验算法出现上面描述的几种情况，那么 G 是最大平面图。

证明. 在检验算法成功结束后，得到的是完全平面图的嵌入，因为：

- 在 $PHASE\_II$ 之后，G 变成了 K_4 的嵌入
- 在 $PHASE\_III$ 之后，G 是一个平面图的嵌入

所以在 $PHASE\_III$ 成功结束以后，可以确定 G 是一个最大平面图，因为已经得到了它的嵌入。

引理 16. 这个检验算法运行所需的时间也是线性的。

证明. 分为以下两种情况：

- 假设 $G=(V,E)$ 是最大平面图：
  - 根据引理 14，原算法所需时间是 $O(|V|)$ 的
  - $1a$ 和 $1b$ 所需的总时间复杂度不超过读入的时间复杂度，也就是 $O(|V|)$
  - 其它判断都只需要 $O(1)$ 的时间，执行总次数不超过 $O(|V|)$
- 假设 $G=(V,E)$ 不是最大平面图：
  - 这个算法会比是最大平面图的情况下结束得更早，时间复杂度更低，不超过 $O(|E|)$

4.6 总结

本文给出了一个 $O(|V|+|E|)$ 时间内判断一个无向图是不是最大平面图的、相对比较简便的算法，可以完美解决此题。

5 References

[1] Stamm-Wilbrandt, H., A simple linear time algorithm for embedding maximal planar graphs

[2] Even, S., Graph Algorithms, Computer Science Press, 1979