DOI：10.13718/j.cnki.xdzk.2016.10.007

图的点可区别边染色猜想的算法①

江世明，　李敬文，　江红豆

兰州交通大学 电子与信息工程学院，兰州 730070

摘要：针对图 $K_{2n}\backslash E(k_{1,m})$ 的点可区别边色数猜想，设计了一种新型的点可区别边染色算法. 根据点可区别边染色的约束条件构建目标函数，利用交换规则进行逐步寻优，直到目标函数的值满足要求时染色成功. 同时给出了算法的执行步骤、分析和测试结果. 实验结果表明，该算法验证了猜想是成立的.

关　键　词：完全图；点可区别边染色；点可区别边色数；交换规则

中图分类号：O157.5　　**文献标志码**：A　　**文章编号**：1673－9868(2016)10－0047－08

图染色问题是图论领域的一个重要研究方向，对它的研究具有重要的理论意义和实际意义. 图染色理论起源于著名的四色猜想，随后一系列新染色被相继提出. 文献[1]提出了图的点可区别边染色(强边染色)的概念和猜想，并由此揭开了图的点可区别边染色研究的序幕. 文献[2]证明了阶数不小于3($|V(G)|\geqslant 3$)的图，其点可区别边色数小于或等于图的阶数加1. 文献[3]得到了仅由路或者圈构成($\Delta(G)=2$)的图的点可区别边染色数. 文献[4]证明了任意无孤立边且最多只有1个孤立点的图，其点可区别边色数小于或等于顶点数加1. 随后，邻点可区别边染色[5]、邻点可区别全染色[6]、$D(\beta)$ 点可区别全染色[7]、点可区别全染色[8]、邻点可区别V-全染色[9]等一系列的概念和猜想相继被提出来. 文献[10]对图 $K_{2n}\backslash E(k_{1,m})$ 的点可区别边染色进行了研究，得到了该类图在 $2\leqslant n\leqslant 9$ 时的点可区别边色数，并猜想当 $n=2m,2m+1$ 时，其点可区别边色数仍为 $2n$.

本文设计了一种新型的点可区别边染色算法，该算法借鉴了文献[11－12]中提出的一般图的点可区别边染色算法，对解决色集合冲突问题提出了一种新的交换规则. 采用该算法对文献[10]中提出的图 $K_{2n}\backslash E(k_{1,m})$ 的点可区别边染色猜想进行证明，并给出了算法的描述、分析和测试结果.

1　相关定义及猜想

本文研究的图为无向简单连通图，$V(G)$ 和 $E(G)$ 分别表示图 G 的顶点集和边集，K_{2n} 表示顶点个数为 $2n$ 的完全图，$K_{1,m}$ 表示具有 $m+1$ 个顶点的爪图，图 $K_{2n}\backslash E(k_{1,m})$ 表示 K_{2n} 去掉子图 $K_{1,m}$ 的边集 $E(K_{1,m})$ 后的图. $C(v)$ 表示顶点 v 和其关联边所用颜色组成的集合，$\overline{C(v)}$ 表示顶点 v 的色补集合(即未使用颜色的集合).

由文献[10]的定义1我们可得到：$\chi_{v\mathrm{dec}}(G)$ 为 G 的点可区别边色数.

猜想 1[1]　对于阶数不小于3的连通图 $G(V,E)$，有

$$\mu(G)\leqslant\chi_{v\mathrm{dec}}(G)\leqslant\mu(G)+1$$

其中 $\mu(G)=\min\left\{\lambda:\binom{\lambda}{i}\geqslant n_i,\ \delta(G)\leqslant i\leqslant\Delta(G)\right\}$ 为图 G 的组合度，n_i 为 $V(G)$ 中度为 i 的点的个数，

① 收稿日期：2015－10－20
基金项目：国家自然科学基金项目(11461038).
作者简介：江世明(1991－)，男，江西宜春人，硕士研究生，主要从事智能计算与组合优化的研究.

$\delta(G)$ 和 $\Delta(G)$ 分别为图 G 的最小度与最大度.

文献[10]在研究图 $K_{2n}\backslash E(k_{1,m})$ 的点可区别边染色的过程中提出了如下猜想:

猜想 2[10] 对于 $2n$ 阶完全图 K_{2n},设 $K_{1,m}$ 是 K_{2n} 的子图,当 $n=2m,2m+1$ 时,则有 $\chi_{vdec}(K_{2n}\backslash E(k_{1,m}))=2n$.

下文用 H_{2n} 表示图 $K_{2n}\backslash E(k_{1,m})$,对于给定的正整数 m,设图 H_{2n} 的顶点数为 $p=2n$,则 p 与 m 的关系如下:

$$p=\begin{cases}4m & n=2m\\ 4m+2 & n=2m+1\end{cases} \tag{1}$$

依据公式(1),可得图 H_p 的顶点度分布情况如下:

$$n_{(p-1)}=p-1-m \qquad n_{(p-2)}=m \qquad n_{(p-1-m)}=1 \tag{2}$$

针对猜想2,本文设计了一种点可区别边染色算法,验证了当 m 与 p 满足公式(1)的关系时,偶阶完全图 K_p 去掉子图 $k_{1,m}$ 的边集 $E(k_{1,m})$ 后,其点可区别边色数等于 p. 由此说明猜想2成立. 另外,文中未描述的术语及符号参见文献[13].

2 算 法

对图 H_p 进行点可区别边染色,并要求满足2个约束条件:

(i) 相邻的边染不同的颜色;

(ii) 任意两个顶点所关联的边所组成的色集合不相同.

由上述约束条件,定义约束函数如下:

2.1 函数定义

(a) 边约束函数

对任意相邻的两边 $e_1,e_2\in E(H_p)$,令:

$$h_1(e_1,e_2)=\begin{cases}1 & f(e_1)=f(e_2)\\ 0 & \text{否则}\end{cases}$$

则有

$$F_{ee}=\sum_{e_1,e_2\in E(H_p)}h_1(e_1,e_2)$$

其中 F_{ee} 表示相邻的两条边染色冲突的个数,当且仅当 $F_{ee}=0$ 时满足约束条件(i).

(b) 色集合约束函数

对于任意的两顶点 $u,v\in V(H_p)$,令:

$$h_2(u,v)=\begin{cases}1 & C(u)=C(v)\\ 0 & \text{否则}\end{cases}$$

则有

$$F_{uv}=\sum_{u,v\in V(H_p)}h_2(u,v)$$

其中 F_{uv} 表示色集合冲突的顶点对的个数,当且仅当 $F_{uv}=0$ 时满足约束条件(ii).

(c) 目标函数

由上述2个约束函数,给出总目标函数:

$$F_{vdec}=F_{ee}+F_{uv}$$

当且仅当 $F_{vdec}=0$ 时,点可区别边染色成功.

2.2 算法描述

本文采用图的邻接矩阵来表示 H_p,算法中通过二维数组 color$[p+1][p+1]$ 来存储图的邻接矩阵. 下面对3个算法的执行步骤进行描述:

算法 1 随机生成图的邻接矩阵

输入:m;输出:图 H_p 的邻接矩阵. 算法伪代码如下:

假设图 H_p 的顶点序号为{1，…，p}，利用 random() 函数从中选取 $m+1$ 个数存入 star_v[]，把 star_v[] 作为 $k_{1,m}$ 的顶点集. 由于图 H_p 是高度对称的，无论爪图顶点集如何变化，通过图的旋转和顶点下标变换，都可以变换成同一个图. 所以本算法输出的 color[][] 可以代表给定 p 值的所有 H_p 图.

```
    Begin
  If(选择公式(1)中的情况 1：n = 2m) Then p = 4m
  Else Then p = 4m + 2
  End If
  For i = 1 to p
    For j = 1 to p
      If( j == i) Then color [i][j] = color [j][i] = 0(0 表示图中没有边：v_i v_j)
      Else Then color [i][j] = color [j][i] = 1(1 表示图中有边：v_i v_j)
      End If
    End For
  End For
  For t = 1 to m
    star_v[t] = random()
    If(t > 0) Then color[star_v[0]][star_v[t]] = color[star_v[t]][star_v[0]] = 1
    End If
  End For
End
```

算法 2　正常边染色交换函数

输入：图 H_p 的邻接矩阵；输出：正常边染色邻接矩阵. 算法伪代码如下：

根据图 H_p 的顶点度分布表计算 $\mu(H_p)$，令初始色数 $k=\mu(H_p)$. 利用 random() 函数对边进行预染色，依次扫描预染色后的 color[][] 得到色补集合 comset[][]、顶点排序集 manyc[]、F_{ee} 和 F_{uv} 的值. 其中 comset[i][] 存储的是 $\overline{C(v_i)}$ 的元素，manyc[] 存储的是各顶点按 $|\overline{C(v_i)}|$ 的值由大到小排序后的结果，$F_{ee}=\sum_{i=1}^{p}(d(v_i)+|\overline{C(v_i)}|-k)$，$F_{uv}$ 为 $C(v_i)$ 相同的顶点对数.

```
    Begin
  For i = 1 to p
    For j = 1 to i
      If (color[i][j] ≠ 0) Then color [i][j] = color [j][i] = random()
      End If
    End For
  End For
  While(F_ee ≠ 0) Do
    For i = 0 to (p − 1)
      u = manyc[i]
      For j = i + 1 to p
        v = manyc[j]
        If(comset[u][] ∩ comset[v][] = {a_1, …, a_t}) AND (color[u][v] ≠ 0)
          Then color[u][v] = color[v][u] = a_1，修改 comset[u][] 和 comset[v][]，重新计算 F_ee 和 F_uv
        End If
      End For
    End For
    If(F_ee ≠ 0) Then 重新获取顶点排序集 manyc[]
    End If
  End While
End
```

算法3 色集合冲突交换函数

输入：正常边染色邻接矩阵；输出：点可区别边染色邻接矩阵. 算法伪代码如下：

依次扫描 color[][] 得到 comset[][]，manyc[] 和色集合冲突顶点对(u, v). 其中(u, v)表示$C(u)=C(v)$.

```
Begin
While(F_uv ≠ 0) Do
For i = 1 to p
  x = (u+i)%p, y = (v+i)%p
  If(comset[u][] ⋂ comset[x][] = {a_1, …, a_t}) AND (color[u][x] ≠ 0)
    Then color[u][x] = color[x][u] = a_1, 修改 comset[u][] 和 comset[x][], 计算 F_uv
  Else If(comset[v][] ⋂ comset[y][] = {b_1, …, b_r}) AND (color[v][y] ≠ 0)
    Then color[v][y] = color[y][v] = a_1, 修改 comset[v][] 和 comset[y][], 计算 F_uv
  Else (comset[u][] ⋂ comset[v][] = {c_1, …, c_s}) AND (color[u][v] ≠ 0)
    Then color[u][v] = color[v][u] = c_1, 修改 comset[u][] 和 comset[v][], 计算 F_uv
  End If
End For
If(F_uv ≠ 0) Then 依次扫描 comset[][] 得到色补集合冲突色集 err_set[]
  For i = 0 to (p-1)
    m_1 = manyc[i]
    For j = i+1 to p
      m_2 = manyc[j]
      If(comset[m_1][] ⋂ comset[m_2][] = {d_1, …, d_t}) AND (color[m_1][m_2] ≠ 0) Then
        If(color[m_1][m_2] ∈ err_set[]) Then
          color[m_1][m_2] = color[m_2][m_1] = d_1, 修改 comset[m_1][] 和 comset[m_2][], 计算 F_uv
        Else Then 执行预交换，计算此时的 F'_uv
          IF(F'_uv ≤ F_uv) THEN
            color[m_1][m_2] = color[m_2][m_1] = d_1, 修改 comset[m_1][], comset[m_2][] 和 err_set[], 计算 F_uv
          End If
          IF(err_set[] 连续 3 次迭代保持不变) Then
            color[m_1][m_2] = color[m_2][m_1] = d_1, 修改 comset[m_1][], comset[m_2][] 和 err_set[], 计算 F_uv
          End If
        End If
      End If
    End For
  End For
End If
End While
End
```

3 算法分析

影响算法时间复杂度的因素主要有3个方面：

(a) 生成图 H_p 的邻接矩阵，其中 $T_1(p)=O(p^2)$；

(b) 正常边染色交换，其中从 manyc[] 依次取值，比较交换构成两层循环，令最大迭代次数为 max，所以 $T_2(p)=O(\max^* p^2)$；

(c) 色集合冲突交换，其中按色集合冲突顶点对进行交换的时间复杂度为 $O(p)$；依次从 manyc[] 取值，比较交换构成两层循环，其时间复杂度为 $O(p^2)$；令最大迭代次数为 max，所以 $T_3(p)=O(\max^* p^2)$.

综合上述分析，本算法的时间复杂度为 $O(\max^* p^2)$.

4　主要结果

4.1　算法测试

文献[10]中已证明了 $p \leqslant 18$ 的图，本文选取 20 个点的图进行算法测试，根据公式(1) 可得 $m=5$.

具体染色步骤如下：

步骤 1　利用 random() 函数得到爪图的顶点集 star_v[]$=\{1, 4, 5, 9, 17, 19\}$，此时图 H_{20} 的邻接矩阵 color[][] 如图 1 所示：

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}
v_1	0	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	0	1	0	1
v_2	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
v_3	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
v_4	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
v_5	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
v_6	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
v_7	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
v_8	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
v_9	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
v_{10}	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
v_{11}	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
v_{12}	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
v_{13}	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
v_{14}	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
v_{15}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
v_{16}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
v_{17}	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
v_{18}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v_{19}	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
v_{20}	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

图 1　H_{20} 的邻接矩阵

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}
v_1	0	5	13	0	0	20	15	2	0	9	11	14	1	16	19	10	0	3	0	4
v_2	5	0	15	19	8	10	12	13	6	7	17	1	4	11	16	20	3	2	18	14
v_3	13	15	0	4	6	11	20	19	14	17	10	18	3	1	8	16	12	9	2	5
v_4	0	19	4	0	17	7	11	18	2	9	5	20	13	14	12	16	1	3	10	6
v_5	0	8	6	17	0	14	13	20	1	7	15	19	5	4	9	2	12	11	10	3
v_6	20	10	11	7	14	0	6	9	13	15	12	3	16	18	8	19	1	2	4	17
v_7	15	12	20	11	13	6	0	9	4	8	10	16	1	7	19	2	17	3	14	5
v_8	2	13	19	18	20	9	9	0	14	7	1	5	4	17	10	6	12	16	15	3
v_9	0	6	14	2	1	13	4	14	0	12	11	18	10	5	20	17	8	7	3	15
v_{10}	9	7	17	9	7	15	8	7	12	0	11	10	6	14	5	2	1	3	18	19
v_{11}	11	17	10	5	15	12	10	1	11	11	0	13	3	2	20	16	8	18	14	4
v_{12}	14	1	18	20	19	3	16	5	18	10	13	0	15	8	4	6	11	7	2	12
v_{13}	1	4	3	13	5	16	1	4	10	6	3	15	0	19	14	8	11	18	9	7
v_{14}	16	11	1	14	4	18	7	17	5	14	2	8	19	0	12	6	13	3	9	20
v_{15}	19	16	8	12	9	8	19	10	20	5	20	4	14	12	0	13	7	18	6	11
v_{16}	10	20	16	16	2	19	2	6	17	2	16	6	8	6	13	0	15	18	5	1
v_{17}	0	3	12	1	12	1	17	12	8	1	8	11	11	13	7	15	0	20	19	10
v_{18}	3	2	9	3	11	2	3	16	7	3	18	7	18	3	18	18	20	0	8	4
v_{19}	0	18	2	10	10	4	14	15	3	18	14	2	9	9	6	5	19	8	0	7
v_{20}	4	14	5	6	3	17	5	3	15	19	4	12	7	20	11	1	10	4	7	0

图 2　H_{20} 边预染色结果

步骤2 根据公式(2)计算图 H_{20} 的组合度 $\mu(H_{20})=20$，令初始色数 $k=20$. 对图 H_{20} 进行边预染色，预染色后的color[][]和边冲突情况分别如图2、图3所示，由图3可得，此时 $F_{ee}=\sum_{i=1}^{20}(d(v_i)+|\overline{C(v_i)}|-k)=48$，$F_{uv}=0$.

v_i	$d(v_i)$	$\overline{C(v_i)}$	$d(v_i)+\|\overline{C(v_i)}\|-k$
v_1	14	{6, 7, 8, 12, 17, 18}	0
v_2	19	{9}	0
v_3	19	{7}	0
v_4	18	{8, 15}	0
v_5	18	{16, 18}	0
v_6	19	{5}	0
v_7	19	{18}	0
v_8	19	{8, 11}	1
v_9	18	{9, 16, 19}	1
v_{10}	19	{4, 13, 16, 20}	3

v_i	$d(v_i)$	$\overline{C(v_i)}$	$d(v_i)+\|\overline{C(v_i)}\|-k$
v_{11}	19	{6, 7, 9, 19}	3
v_{12}	19	{9, 17}	1
v_{13}	19	{2, 12, 17, 20}	3
v_{14}	19	{10, 15}	1
v_{15}	19	{1, 2, 3, 15, 17}	4
v_{16}	19	{3, 4, 7, 9, 11, 12, 14}	6
v_{17}	18	{2, 4, 5, 6, 9, 14, 16, 18}	6
v_{18}	19	{1, 5, 6, 10, 12, 13, 14, 15, 17, 19}	9
v_{19}	18	{1, 11, 12, 13, 16, 17, 20}	5
v_{20}	19	{2, 8, 9, 13, 16, 18}	5

图3 边冲突情况

步骤3 此时 manyc[]={v_{18}, v_{17}, v_{16}, v_{19}, v_1, v_{20}, v_{15}, v_{10}, v_{11}, v_{13}, v_9, v_4, v_5, v_8, v_{12}, v_{14}, v_2, v_3, v_6, v_7}，依次从 manyc[] 中取出顶点按规则交换. 其中用 $f(v_iv_j)$：color1⇒color2 来表示 $f(v_iv_j)$ 由 color1 色换成 color2 色，后面交换都用此符号来表示. 下面列出 v_{18} 的具体交换过程：$f(v_{18}v_{17})$：20⇒5，$f(v_{18}v_{16})$：18⇒12，$f(v_{18}v_{19})$：8⇒1，$f(v_{18}v_1)$：3⇒6，$f(v_{18}v_{20})$：4⇒8，$f(v_{18}v_{15})$：18⇒15，$f(v_{18}v_{10})$：3⇒4，$f(v_{18}v_{11})$：18⇒19，$f(v_{18}v_{13})$：18⇒17，$f(v_{18}v_5)$：11⇒18，$f(v_{18}v_8)$：16⇒11，$f(v_{18}v_{14})$：3⇒10. 按此交换规则，第一轮交换后 $F_{ee}=20$，$F_{uv}=0$. 重新获取 manyc[] 的值进行下一轮交换，经过16轮交换后可得 $F_{ee}=0$，$F_{uv}=7$，表示正常边染色成功. 正常边染色成功后的 color[][] 如图4所示.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}
v_1	0	9	14	0	0	13	5	15	0	2	8	1	18	19	11	16	0	20	0	17
v_2	9	0	15	11	16	10	12	2	8	19	18	17	4	5	6	20	3	13	1	14
v_3	14	15	0	19	6	11	20	12	5	17	10	18	3	1	4	13	7	8	9	2
v_4	0	11	19	0	17	7	2	6	12	9	4	14	13	16	3	18	8	15	5	1
v_5	0	16	6	17	0	14	13	20	4	11	5	10	9	8	12	15	2	1	3	19
v_6	13	10	11	7	14	0	6	16	17	15	20	2	5	18	8	12	1	9	19	4
v_7	5	12	20	2	13	6	0	9	18	4	15	16	17	10	19	7	14	3	11	8
v_8	15	2	12	6	20	16	9	0	13	7	1	5	19	17	10	11	18	4	8	3
v_9	0	8	5	12	4	17	18	13	0	1	3	11	10	9	15	19	16	2	20	7
v_{10}	2	19	17	9	11	15	4	7	1	0	12	8	16	20	18	3	13	5	14	6
v_{11}	8	18	10	4	5	20	15	1	3	12	0	13	14	2	9	17	6	19	7	11
v_{12}	1	17	18	14	10	2	16	5	11	8	13	0	15	4	20	9	19	7	6	12
v_{13}	18	4	3	13	9	5	17	19	10	16	14	15	0	7	2	8	11	6	12	20
v_{14}	19	5	1	16	8	18	10	17	9	20	2	4	7	0	14	6	12	11	13	15
v_{15}	11	6	4	3	12	8	19	10	15	18	9	20	2	14	0	1	5	17	16	13
v_{16}	16	20	13	18	15	12	7	11	19	3	17	9	8	6	1	0	4	14	2	5
v_{17}	0	3	7	8	2	1	14	18	16	13	6	19	11	12	5	4	0	10	17	9
v_{18}	20	13	8	15	1	9	3	4	2	5	19	7	6	11	17	14	10	0	18	16
v_{19}	0	1	9	5	3	19	11	8	20	14	7	6	12	13	16	2	17	18	0	10
v_{20}	17	14	2	1	19	4	8	3	7	6	11	12	20	15	13	5	9	16	10	0

图4 H_{20} 正常边染色结果

步骤4 此时色集合冲突顶点对为：(v_2, v_{15}),(v_3, v_{11}),(v_6, v_{12}),(v_6, v_{14}),(v_7, v_{13}),(v_{10}, v_{16}),(v_{12}, v_{14}). 按规则进行交换. 具体交换如下：$f(v_2v_1)$：9⇒7，$f(v_3v_{11})$：10⇒16，$f(v_3v_4)$：19⇒10，$f(v_6v_1)$：13⇒3，$f(v_7v_{13})$：17⇒1，$f(v_{10}v_1)$：2⇒10，$f(v_7v_{13})$：1⇒17，$f(v_{11}v_{16})$：17⇒10，$f(v_{12}v_{14})$：4⇒3，$f(v_{12}v_{19})$：6⇒4，$f(v_7v_{13})$：17⇒1，$f(v_7v_{11})$：15⇒17，$f(v_7v_{19})$：11⇒15，$f(v_{13}v_{16})$：8⇒17. 交换

后$\overline{C(v_1)}$，…，$\overline{C(v_{20})}$为：{2，4，6，9，12，13}，{9}，{19}，{19，20}，{7，18}，{13}，{11}，{14}，{6，14}，{2}，{15}，{6}，{8}，{4}，{7}，{8}，{15，20}，{12}，{6，11}，{18}. 此时 $F_{ee}=0$，$F_{uv}=1$.

步骤 5　此时 err_set[]＝{8}，manyc[]＝{v_1，v_4，v_5，v_9，v_{17}，v_{19}，v_2，v_3，v_6，v_7，v_8，v_{10}，v_{11}，v_{12}，v_{13}，v_{14}，v_{15}，v_{16}，v_{18}，v_{20}}，依次从 manyc[] 中取出顶点按规则交换. 具体交换如下：$f(v_1v_6)$：3⇒13，$f(v_1v_{10})$：10⇒2，$f(v_1v_{12})$：1⇒6，$f(v_1v_{18})$：20⇒12，$f(v_4v_{17})$：8⇒20，$f(v_4v_{13})$：13⇒8. 交换后$\overline{C(v_1)}$，…，$\overline{C(v_{20})}$为：{1，3，4，9，10，20}，{9}，{19}，{13，19}，{7，18}，{3}，{11}，{14}，{6，14}，{10}，{15}，{1}，{13}，{4}，{7}，{8}，{8，15}，{20}，{6，11}，{18}. 此时 $F_{ee}=0$，$F_{uv}=0$，表示各顶点的色集合都不相同，交换结束.

步骤 6　此时 $F_{vdec}=F_{ee}+F_{uv}=0$，表示点可区别边染色成功，且 k 与猜想值一致，此时 color[][] 如图 5 所示.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}
v_1	0	7	14	0	0	13	5	15	0	2	8	6	18	19	11	16	0	12	0	17
v_2	7	0	15	11	16	10	12	2	8	19	18	17	4	5	6	20	3	13	1	14
v_3	14	15	0	10	6	11	20	12	5	17	16	18	3	1	4	13	7	8	9	2
v_4	0	11	10	0	17	7	2	6	12	9	4	14	8	16	3	18	20	15	5	1
v_5	0	16	6	17	0	14	13	20	4	11	5	10	9	8	12	15	2	1	3	19
v_6	13	10	11	7	14	0	6	16	17	15	20	2	5	18	8	12	1	9	19	4
v_7	5	12	20	2	13	6	0	9	18	4	17	16	1	10	19	7	14	3	15	8
v_8	15	2	12	6	20	16	9	0	13	7	1	5	19	17	10	11	18	4	8	3
v_9	0	8	5	12	4	17	18	13	0	1	3	11	10	9	15	19	16	2	20	7
v_{10}	2	19	17	9	11	15	4	7	1	0	12	8	16	20	18	3	13	5	14	6
v_{11}	8	18	16	4	5	20	17	1	3	12	0	13	14	2	9	10	6	19	7	11
v_{12}	6	17	18	14	10	2	16	5	11	8	13	0	15	3	20	9	19	7	4	12
v_{13}	18	4	3	8	9	5	1	19	10	16	14	15	0	7	2	17	11	6	12	20
v_{14}	19	5	1	16	8	18	10	17	9	20	2	3	7	0	14	6	12	11	13	15
v_{15}	11	6	4	3	12	8	19	10	15	18	9	20	2	14	0	1	5	17	16	13
v_{16}	16	20	13	18	15	12	7	11	19	3	10	9	17	6	1	0	4	14	2	5
v_{17}	0	3	7	20	2	1	14	18	16	13	6	19	11	12	5	4	0	10	17	9
v_{18}	12	13	8	15	1	9	3	4	2	5	19	7	6	11	17	14	10	0	18	16
v_{19}	0	1	9	5	3	19	15	8	20	14	7	4	12	13	16	2	17	18	0	10
v_{20}	17	14	2	1	19	4	8	3	7	6	11	12	20	15	13	5	9	16	10	0

图 5　H_{20} 最终染色结果

4.2　测试结果分析

本文在 Windows 64 位系统、2G 内存、VS2005 的环境下，对 102 个顶点的所有偶阶图进行了测试，测试结果如表 1 所示(其中 m 与 p 满足公式(1)，k 为点可区别边染色所用的最少颜色数)：

表 1　颜色数 k 统计

m	2	2	3	3	…	25	25	…
p	8	10	12	14	…	100	102	…
k	8	10	12	14	…	100	102	…

由表 1 可知，在所有测试结果中：$k=p$，并由此可得 $\chi_{vdec}(H_p)=k=p$，即猜想 2 是成立. 通过本文的算法，可以对 $p>102$ 的图继续进行验证. 综合上述可知，当 $n=2m$，$2m+1$ 时，$\chi_{vdec}(H_p)=p$. 因此，猜想 2 是成立的.

5　结　　语

在对图 H_p 进行点可区别边染色时，极易出现色集合冲突问题. 本文设计的算法采用了一种新的色集合冲突交换规则来解决图的点可区别边染色过程中出现的色集合冲突问题. 大量的实验结果表明，该算法是正确的，并为验证猜想提供了可靠的数据支持. 另外，对该算法略加修改，可以用于解决其它完全图去掉子图边集后的点可区别边染色问题.

参考文献:

[1] BURRIS A C, SCHELP R H. Vertex-Distinguishing Proper Edge-Colorings [J]. Journal of Graph Theory, 1997, 26(2): 73－82.

[2] BAZGAN C, HARKAT-BENHAMDINE A, LI H, et al. On the Vertex-Ddistinguishing Proper Edge-Colorings of Graphs [J]. Journal of Combinatorial Theory(Ser B), 1999, 75: 288－301.

[3] BALISTER P N, BOLLOBÁS B, SCHELP R H, et al. Vertex-Distinguishing Colorings of Graphs with $\Delta(G)=2$ [J]. Discrete Mathematics, 2002, 252: 17－29.

[4] BALISTER P N, RIORDAN O M, SCHELP R H. Vertex-Distinguishing Edge Colorings of Graphs [J]. Journal of Graph Theory, 2003, 42: 95－109.

[5] ZHANG Z F, lIU L Z, WANG J F. Adjacent Strong Edge Coloring of Graphs [J]. Applied Mathematics Letters, 2002, 15(5): 623－626.

[6] ZHANG Z F, CHEN X E, LI J W, et al. On Adjacent-Vertex-Distinguishing Total Coloring of Graphs [J]. Science in China(Series A), 2005, 48(3): 289－299.

[7] ZHANG Z F, LI J W, CHEN X E, et al. $D(\beta)$-Vertex-Distinguishing Total Coloring of Graphs [J]. Science in China (Series A), 2006, 49(10): 1430－1440.

[8] ZHANG Z F, QIU P X, XU B G, et al. Vertex-Distinguishing Total Coloring of Graphs [J]. Ars Combinatoria, 2008, 87(2): 33－45.

[9] 李沐春, 王双莉, 张伟东, 等. 若干路的冠图的邻点可区别V-全染色 [J]. 西南大学学报(自然科学版), 2014, 36(6): 97－99.

[10] 李敬文, 王鸿杰, 文 飞, 等. 图 $K_{2n}\backslash E(k_{1,m})(n\geqslant 2)$的点可区别边染色 [J]. 西南大学学报(自然科学版), 2012, 34(8): 86－90.

[11] 董 威, 贾西贝, 李小慧, 等. 随机图的邻点可区别 I-全染色算法 [J]. 西南师范大学学报(自然科学版), 2015, 40(4): 8－15.

[12] 李敬文, 张云寒, 陈志鹏, 等. 图的点可区别边染色算法研究 [J]. 计算机应用研究, 2014, 31(3): 760－764.

[13] BONDY J A, MURTY S R. Graph Theory with Applications [M]. New work: The Macmillan Press Ltd, 1976.

Algorithm for Conjecture of Vertex Distinguishing Edge Coloring of Graphs

JIANG Shi-ming, LI Jing-wen, JIANG Hong-dou

School of Electronic & Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China

Abstract: A proper edge coloring of a graph such that no two vertices have the same color set is called vertex distinguishing edge coloring, and the minimal number of coloring is called vertex distinguishing edge chromatic number. A new algorithm of vertex distinguishing edge coloring has been designed for conjecture of $K_{2n}\backslash E(k_{1,m})$. Accurately, the algorithm has established objective function by the constraint rules of vertex distinguishing edge coloring, used exchange rules to optimize the results gradually and accomplished coloring till the value of objective function met the requirements. And also described the algorithm steps, algorithm analysis and test results. The experimental results show that the conjecture is right.

Key words: complete graph; vertex distinguishing edge coloring; vertex distinguishing edge chromatic number; exchange rules

责任编辑 廖 坤