ELSEVIER

Information Sciences 114 (1999) 105–126

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

Three-quarter approximation for the number of unused colors in graph coloring

Wen-Guey Tzeng *, Gow-Hsing King

Department of Computer and Information Science, National Chiao Tung University, Hsinchu 30050, Taiwan, ROC

Received 27 January 1997; accepted 27 April 1998

Communicated by Kao Ming Yang

Abstract

The *graph coloring* problem is to color vertices of a graph so that no adjacent vertices are of the same color. The problem is difficult not only in finding the optimal solution, but also in approximation. Since it is hard to approximate the minimum number of colors, we consider to approximate *the maximum number of unused colors*. This approximation is based on saving colors with respect to the most naive coloring method, which colors each vertex with a different color. In this paper we propose a polynomial-time graph coloring algorithm with approximation ratio 3/4 for the maximum number of unused colors, which improves the previous result 2/3.

Keywords: Graph coloring; Approximation

1. Introduction

Let G be an undirected simple graph. The *graph coloring* problem is to color vertices of G so that no adjacent vertices are of the same color. The *chromatic number* $\chi(G)$ of G is the minimum number of colors among such colorings. The graph coloring problem has been studied extensively in the past and has many applications in solving other problems [6]. However, the problem is difficult not

* Corresponding author. Fax: 886 35 721 490; e-mail: tzeng@cis. nctu.edu.tw

PII: S0020-0255(98)10058-0

only in finding the optimal solution, but also in approximation. The best polynomial-time approximation algorithm for $\chi(G)$ is of ratio $O(n(\log\log n)^2/(\log n)^3)$, where n is the number of vertices of G. If $\chi(G)$ is a constant, such as 3, the approximation ratio can be slightly improved [2,10]. Lund and Yanakakis [9] show that there is a constant $\epsilon > 0$ such that no polynomial-time graph coloring algorithm can have an approximation ratio better than n^ϵ unless $P = NP$. Note that $O(n(\log\log n)^2/(\log n)^3)$ is greater than n^ϵ for any $\epsilon > 0$.

Since it is hard to approximate $\chi(G)$, we consider to approximate $n - \chi(G)$, called *the maximum number of unused colors* [3,7]. This approximation is based on saving colors with respect to the most naive coloring method, which colors each vertex with a different color. This approximation is to maximize the approximation ratio $\gamma = (n - \beta(G))/(n - \chi(G))$, where $\beta(G)$ is the number of colors found by a polynomial-time graph coloring algorithm. If γ is 1, $\beta(G)$ is equal to $\chi(G)$. We note that a constant γ does not imply that $\chi(G)$ can be approximated within some constant ratio. Demange et al. [3] propose an algorithm of $\gamma = 1/2$. Later, Hassin and Lahav [7] propose another algorithm of $\gamma = 2/3$.

To see the importance of approximation from this direction, we examine the value $\chi(G)$. The current result on approximating $\chi(G)$ works only for very small $\chi(G)$. If $\chi(G)$ is greater than $O((\log n)^3/(\log\log n)^2)$, the approximation works no better than a naive coloring algorithm in the worst case. However, for large $\chi(G)$ an algorithm for approximating $n - \chi(G)$ will work better. For example, if $\chi(G)$ is $n/10$ and the approximation ratio γ is 3/4, the algorithm can color G with less than $(13/40)n$ colors.

In this paper we propose a polynomial-time graph coloring algorithm of $\gamma = 3/4$. To explain our algorithm better, we use *clique cover* to describe graph coloring. The clique cover problem is to partition the vertices of G into vertex sets $V_1, V_2, \ldots, V_k$ such that each $V_i, 1 \leqslant i \leqslant k$, induces a clique. Let $\phi(G)$ be the minimum number of cliques to which the vertices of G can be partitioned. To find the minimum coloring for G is equivalent to find the minimum clique cover for $\bar{G}$, the complimentary graph of G, that is, $\phi(\bar{G}) = \chi(G)$. The clique cover problem is NP-complete even when k is fixed to 3. It remains NP-complete when graphs are restricted to have maximum cliques of size at most 3 (4-clique-free graphs) [5].

The key part of our approximation algorithm is to approximate $n - \phi(G)$ for 4-clique-free graphs within ratio 3/4. The algorithm uses a concept similar to *augmenting path* in the matching problem, however, more complicated. We use a charge scheme to distribute weights of single-vertex cliques found by our algorithm to 3-cliques found by our algorithm as well as to 1-cliques and 2-cliques in the optimal partition. A consequence of this result is a clique cover algorithm of approximation ratio 1.5 for the minimum number of cliques for 4-clique-free graphs. We note that the set of tripartite graphs is a subset of 4-clique-free graphs.

2. The algorithm AUC

Our graph coloring algorithm AUC in Table 1, which denotes "approximating unused cliques", approximates $n - \phi(G)$ with ratio $\geqslant 3/4$.

AUC first finds a *maximal* 4-clique set S from G. It then removes S from G to induce a 4-clique-free graph G' and calls procedure A4CFG, which denotes "approximating 4-clique-free graphs", to find a clique partition P' for G'. The maximum clique in P' is at most 3 and G' has $n' = n - 4|S|$ vertices. If $(n' - |P'|)/(n' - \phi(G')) \geqslant 3/4$, the 4-clique set S together with P' form a clique partition for G and approximate $n - \phi(G)$ with ratio 3/4.

Theorem 2.1. *Assume that procedure A4CFG approximates $n - \phi(G)$ for 4-clique-free graphs with ratio 3/4. The algorithm AUC finds a clique partition P for G with $(n - |P|)/(n - \phi(G)) \geqslant 3/4$.*

Proof. $n - \phi(G)$ is at most $4|S| + n' - \phi(G') = n - \phi(G')$. The size of clique partition $P = S \cup P'$ is $|S| + |P'|$. Therefore,

$$n - |P| = n - |S| - |P'| = 3|S| + (n' - |P'|) \geqslant \frac{3}{4}(4|S| + n' - \phi(G'))$$
$$= \frac{3}{4}(n - \phi(G')) \geqslant \frac{3}{4}(n - \phi(G)). \qquad \square$$

3. The procedure A4CFG

Procedure A4CFG finds a clique partition P for a 4-clique-free graph G of n vertices such that $(n - |P|)/(n - \phi(G)) \geqslant 3/4$. We assume that $G = (V, E)$ is 4-clique-free hereafter if not stated otherwise. A set of 3-cliques without common vertices is a *tri-matching* of G, while a set of 2-cliques without common vertices is a *bi-matching* of G. We shall use *matching* and *clique-partition* interchange-

Table 1
The algorithm AUC

Input: A graph $G=(V,E)$.
Output: A clique partition P for G.
1. $S, P, P' \leftarrow \emptyset$;
2. Find a maximal 4-clique set S for G;
3. $G' \leftarrow G \backslash S$;
4. $P' \leftarrow$ A4CFG(G');
5. $P \leftarrow S \cup P'$
6. end.

ably, by which readers should not be confused. Procedure A4CFG uses the concept of augmenting path, which we call *C-augmenting path*, in contrast to the M-augmenting path in solving the maximum (bi-) matching problem [8]. A C-augmenting path is used to reduce the number of cliques in a clique partition by one.

Let $P = T \cup M \cup F$ be a clique partition for G and T, M, and F be the sets of 3-, 2-, and 1-cliques in P, respectively. The vertices of 1-cliques are also called the *free* vertices. For example, for the graph in Fig. 1, $P = \{[a], [i], [p], [q], [r], [b,d], [j,k], [c,e,f], [g,l,m], [h,n,o]\}$ is a clique partition with $T = \{[c,e,f], [g,l,m], [h,n,o]\}$, $M = \{[b,d], [j,k]\}$ and $F = \{[a], [i], [p], [q], [r]\}$.

M-alternating and M-augmenting paths in the bi-matching problem are denoted by our notations in the following. Note that we distinguish M-alternating path from M-augmenting path in this paper.

Definition 3.1. An M-alternating path (with respect to P) is a path

$$[f] - m_1 - m_2 - \cdots - m_{2k-1} - m_{2k}$$

or

$$m_1 - m_2 - \cdots - m_{2k-1} - m_{2k}$$

with (f, m_1) and (m_{2i}, m_{2i+1}) in $E \setminus M$ for $k \geqslant 0$, $1 \leqslant i \leqslant k-1$, $[f]$ in F, and $[m_{2j-1}, m_{2j}]$ in M for $1 \leqslant j \leqslant k$, where $E \setminus M$ denotes the set of edges that are not in cliques of M.

An *M-augmenting path* is a path

$$[f_1] - m_1 - m_2 - \cdots - m_{2k-1} - m_{2k} - [f_2]$$

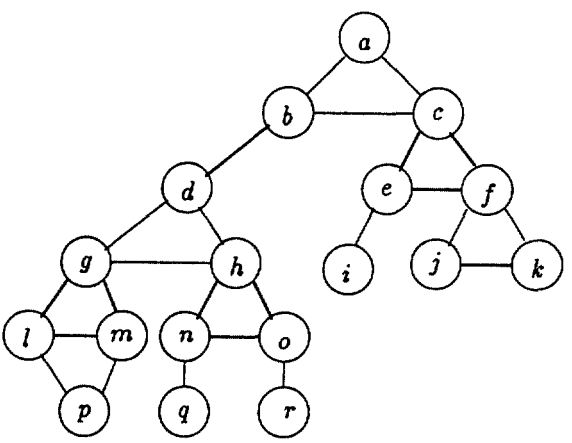

Fig. 1. A 4-clique-free graph and its clique partition.

with (f_1, m_1), (m_{2k}, f_2) and (m_{2i}, m_{2i+1}) in $E \setminus M$ for $k \geqslant 0$, $1 \leqslant i \leqslant k-1$, $[f_1]$ and $[f_2]$ in F, and $[m_{2j-1}, m_{2j}]$ in M for $1 \leqslant j \leqslant k$.

The C-augmenting path, which has two types C_1 and C_2, is crucial in our algorithm and defined as follows.

Definition 3.2. A C_1-augmenting path (with respect to P) is a path

$$[f_1] - m_1 - m_2 - \cdots - m_{2k-1} - m_{2k} - [t_1, t_2, t_3] - [f_2], [f_3]$$

with (f_1, m_1), (m_{2k}, t_1), (t_2, f_2), (t_3, f_3) and (m_{2i}, m_{2i+1}) in $E \setminus M$ for $1 \leqslant i \leqslant k-1$, $[f_1]$, $[f_2]$ and $[f_3]$ in F, $[t_1, t_2, t_3]$ in T and $[m_{2j-1}, m_{2j}]$ in M for $1 \leqslant j \leqslant k$.

A C_2-augmenting path (with respect to P) is a path

$$[f_1] - m_1 - m_2 - \cdots - m_{2k-1} - m_{2k} - [t_1, t_2, t_3] = [f_2]$$

with (f_1, m_1), (m_{2k}, t_1), (t_2, f_2), (t_3, f_2) and (m_{2i}, m_{2i+1}) in $E \setminus M$ for $1 \leqslant i \leqslant k-1$, $[f_1]$ and $[f_2]$ in F, $[t_1, t_2, t_3]$ in T and $[m_{2j-1}, m_{2j}]$ in M for $1 \leqslant j \leqslant k$.

For example, in Fig. 1 $[a] - b - d - [h, n, o] - [q], [r]$ is a C_1-augmenting path and $[a] - b - d - [g, l, m] = [p]$ is a C_2-augmenting path with respect to the clique partition P.

The vertices in a C-augmenting path can be re-partitioned to reduce the number of cliques of P by one. For a C_1-augmenting path $[f_1] - m_1 - m_2 - \cdots - m_{2k-1} - m_{2k} - [t_1, t_2, t_3] - [f_2], [f_3]$, removing cliques $[f_1]$, $[m_{2i-1}, m_{2i}]$, $1 \leqslant i \leqslant k$, $[t_1, t_2, t_3]$, $[f_2]$ and $[f_3]$ from P and then adding cliques $[f_1, m_1]$, $[m_{2i}, m_{2i+1}]$, $1 \leqslant i \leqslant k-1$, $[m_{2k}, t_1]$, $[t_2, f_2]$ and $[t_3, f_3]$ into P will reduce the number of cliques in P by one. Similarly, for a C_2-augmenting path $[f_1] - m_1 - m_2 - \cdots - m_{2k-1} - m_{2k} - [t_1, t_2, t_3] = [f_2]$, removing cliques $[f_1]$, $[m_{2i-1}, m_{2i}]$, $1 \leqslant i \leqslant k$, $[t_1, t_2, t_3]$ and $[f_2]$ from P and then adding cliques $[f_1, m_1]$, $[m_{2i}, m_{2i+1}]$, $1 \leqslant i \leqslant k-1$, $[m_{2k}, t_1]$ and $[t_2, t_3, f_2]$ into P also decreases the size of P by one.

For example, the clique partition in Fig. 1 can be adjusted to

$$\{[i], [p], [a, b], [d, h], [n, q], [o, r], [j, k], [c, e, f], [g, l, m]\}$$

using the C_1-augmenting path $[a] - b - d - [h, n, o] - [q], [r]$.

Remark 3.1. There are other types of augmenting paths that can reduce the size of P. However, they are either hard to find in polynomial time [4] or not important for the correctness of our algorithm.

Procedure A4CFG is shown in Table 2. The input to the procedure is a 4-clique-free graph G. A4CFG first finds a *maximal* 3-clique set T (tri-matching) for G. It then finds a *maximum* 2-clique set M (bi-matching) for $G \setminus T$. Finally, one by one it finds a C-augmenting path and adjusts the current clique partition till no C-augmenting path can be found.

Table 2
The procedure A4CFG

Input: A 4-clique-free graph $G=(V, E)$.
Output: A clique partition P for G.
1. $T, M, F \leftarrow \emptyset$;
2. Find a maximal 3-clique set T for G;
3. Find a maximum 2-clique set M for $G\backslash T$;
4. Let F be the set of vertices in $(G\backslash T)\backslash M$;
5. $P \leftarrow T \cup M \cup F$;
6. **while** $\exists$ C-augmenting path D with respect to P **do**
7. Adjust P according to D;
8. **return** partition P;
9. end.

To find a C-augmenting path can be done in polynomial time by a straightforward (brute-force) method. Adjusting a C-augmenting path reduces the number of free vertices of P by at least two. Therefore, the total time for finding C-augmenting paths and adjusting partitions is polynomial. The time complexity of procedure A4CFG is thus polynomial since to find a maximal 3-clique partition set and a maximum 2-clique partition set is also solvable in polynomial time. Note that to find a maximum 2-clique-partition set is well known as the maximum matching problem.

For each C-augmenting path the bi-matching in the adjusted clique partition P is *maximum* for $G \setminus T$. That is, adjusting the clique partition by C-augmenting paths does not generate M-augmenting paths. This property is important for our analysis in the next section.

Lemma 3.1. *Let $P = T \cup M \cup F$ be a clique partition for a 4-clique-free graph G and D be a C-augmenting path for G with respect to P. If M is a maximum bi-matching for $G \setminus T$, M' of the adjusted partition $P' = T' \cup M' \cup F'$ by D is a maximum bi-matching for $G \setminus T'$.*

Proof. We consider the case for D being a C_1-augmenting path only. The case for D being a C_2-augmenting path can be discussed similarly. Let D be $[f_1] - m_1 - m_2 - \cdots - m_{2k-1} - m_{2k} - [t_1, t_2, t_3] - [f_2], [f_3]$. We assume otherwise that H of $[g_1] - n_1 - n_2 - \cdots - n_{2l-1} - n_{2l} - [g_2]$ is an M-augmenting path with respect to M' after adjusting P by D with $f_1 \neq g_1$ and $f_1 \neq g_2$. Let $m_p = n_{p'}$ be the first vertex that beginning from f_1, D encounters H. That is, vertices m_j, $1 \leqslant j \leqslant p-1$, are not in H. Let $n_{q'} = m_q$ be the first vertex that beginning from g_1, H encounters D. That is, vertices n_j, $1 \leqslant j \leqslant q'-1$, are not in D. Let $m_r = n_{r'}$ be the first vertex that beginning from m_{2k}, D encounters H. That is, vertices m_j, $r+1 \leqslant j \leqslant 2k$ are not in H. Let $n_{s'} = m_s$ be the first vertex that beginning from g_2, H encounters D. That is vertices n_j, $s'+1 \leqslant j \leqslant 2l$, are not in D. There are

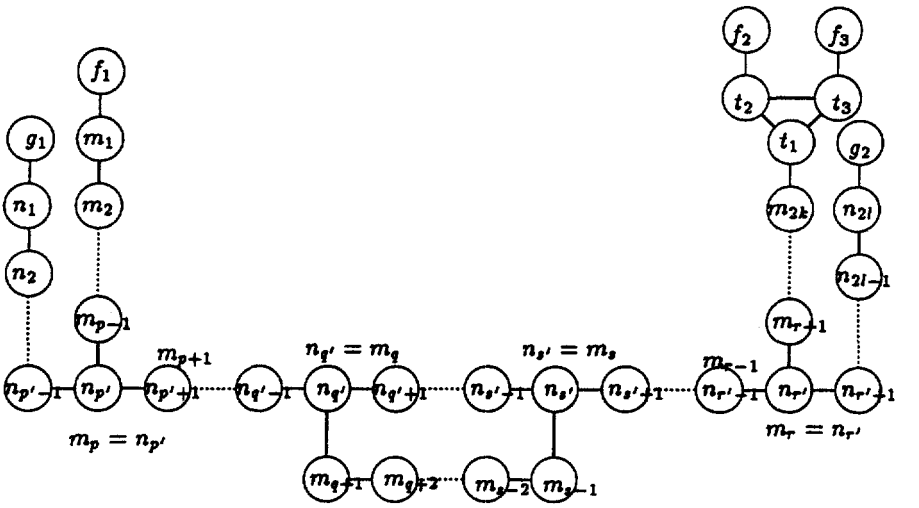

Fig. 2. Intersection of D and H: case 1.

four cases for the position relation of vertices m_p and n_q and that of vertices m_r and n_s, which is shown in Figs. 2–5.

It can be seen that no matter which case there is an M-augmenting path among free vertices f_1, g_1, and g_2, with respect to M, which is a contradiction. For example, in the fourth case (see Fig. 5) with $q < p$ and $r < s$ path $[g_1] - n_1 - \cdots - n_{q'} - m_{q+1} - m_{q+2} - \cdots - m_s - n_{s'+1} - \cdots - n_{2l} - [g_2]$ is an augmenting path with respect to M. Therefore, the lemma holds. □

4. The charge scheme

In this section we prove that procedure A4CFG approximates $n - \phi(G)$ for 4-clique-free graphs with ratio $\geqslant 3/4$.

Let $P = T \cup M \cup F$ be the clique partition found by A4CFG, where $T = \{t_1, t_2, \ldots, t_p\}$, $M = \{m_1, m_2, \ldots, m_q\}$ and $F = \{f_1, f_2, \ldots, f_s\}$. Let $\tilde{P} = \tilde{T} \cup \tilde{M} \cup \tilde{F}$ be the minimum clique partition for G, where $\tilde{T} = \{\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_\alpha\}$, $\tilde{M} = \{\tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_\beta\}$ and $\tilde{F} = \{\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_\gamma\}$ be the sets of 3-, 2- and 1-cliques of $\tilde{P}$, respectively.

We use a charge scheme which assigns weights w to cliques of P and $\tilde{P}$. The initial weights of cliques are assigned as follows:

$$w(t) = -1 \quad \text{for } t \in T,$$

$$w(m) = 0 \quad \text{for } m \in M,$$

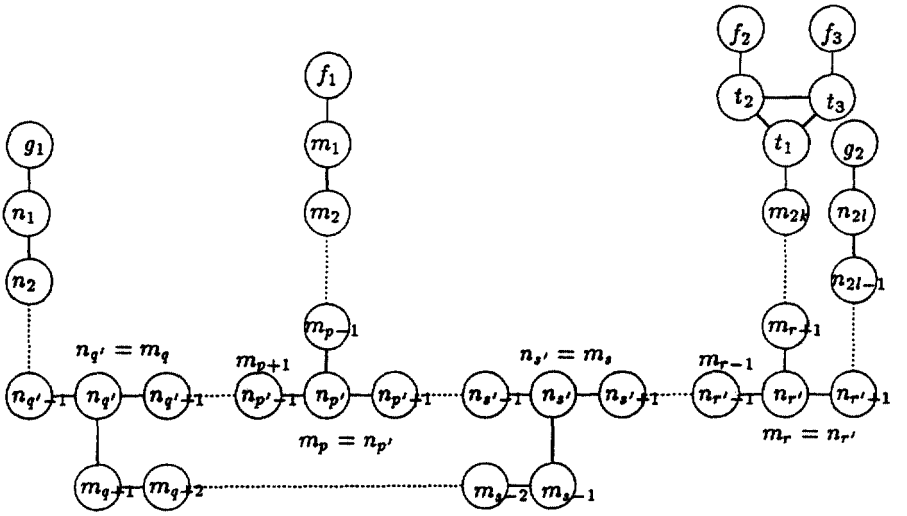

Fig. 3. Intersection of D and H: case 2.

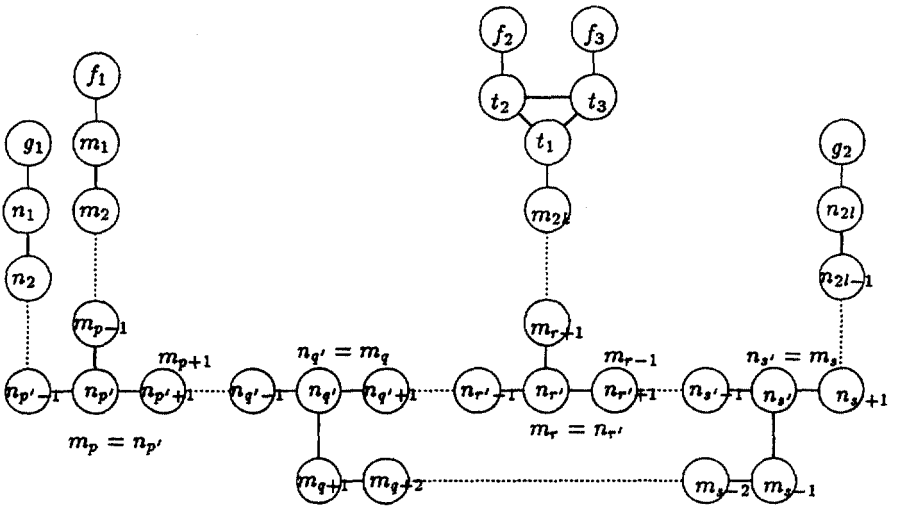

Fig. 4. Intersection of D and H: case 3.

$w(f) = +1$ for $f \in F$,

$w(\tilde{t}) = 0$ for $\tilde{t} \in \tilde{T}$,

$w(\tilde{m}) = -0.5$ for $\tilde{m} \in \tilde{M}$,

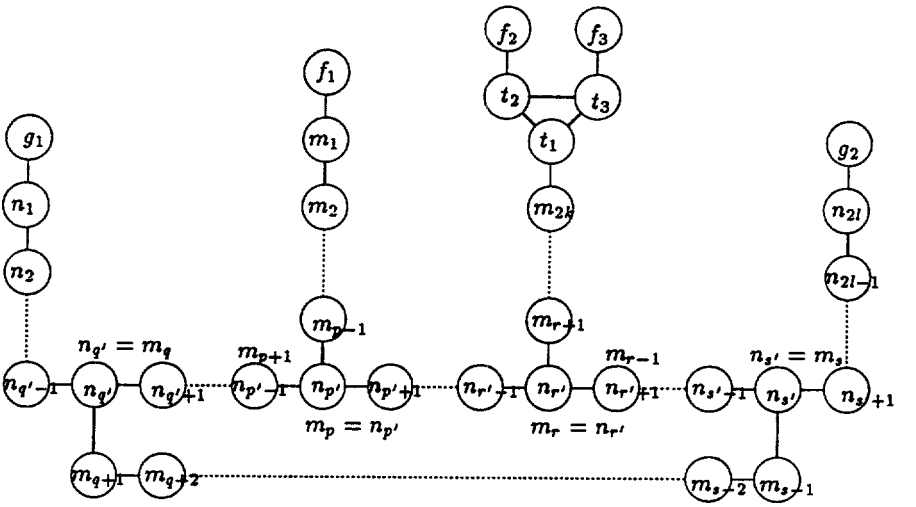

Fig. 5. Intersection of D and H: case 4.

$$w(\tilde{f}) = -1 \quad \text{for } \tilde{f} \in \tilde{F}.$$

Let $w(S) = \sum_{s \in S} w(s)$ for any subset S of cliques in $P \cup \tilde{P}$. By this weight assignment we show a necessary and sufficient condition for A4CFG to have ratio 3/4.

Lemma 4.1. *The approximation ratio of A4CFG is* $\geqslant 3/4$ *if and only if* $w(F) + w(T) + w(\tilde{M}) + w(\tilde{F}) \leqslant 0$.

Proof. To have the approximation ratio, it needs

$$\frac{n - (p + q + s)}{n - (\alpha + \beta + \gamma)} \geqslant \frac{3}{4}.$$

Since $3\alpha + 2\beta + \gamma = 3p + 2q + s = n$, the above equation can be simplified to $s - p - \beta/2 - \gamma \leqslant 0$. Therefore, $w(F) + w(T) + w(\tilde{M}) + w(\tilde{F}) \leqslant 0$. □

The remaining work is to show that the total weight of cliques in $P \cup \tilde{P}$ is $\leqslant 0$. To achieve this, we propagate the positive weights of the 1-cliques f in F to the cliques in $\tilde{F}, \tilde{M}$, and T. The propagation scheme is shown in Table 3. We shall show that the weight of every clique in $P \cup \tilde{P}$ is $\leqslant 0$ after this propagation.

Table 3
The charge scheme

```
1.  ∀t ∈ T, w(t) ← −1.0;                ∀m ∈ M, w(m) ← 0;
2.  ∀f ∈ F, w(f) ← +1.0;                ∀t̃ ∈ T̃, w(t̃) ← 0;
3.  ∀m̃ ∈ M̃, w(m̃) ← −0.5;               ∀f̃ ∈ F̃, w(f̃) ← −1.0;
4.  ∀v ∈ V, visited(v) ← false;
5.  ∀f = [a] ∈ F, w(a) = +1.0 and w(f) = 0;
6.  while (there is a vertex a' with visited(a') = false and w(a') > 0)
7.  {   case [a'] = f̃ ∈ F̃:
8.          w(f̃) ← w(f̃) + w(a');
9.      case [a', b] = m̃ ∈ M̃ and visited(b) = false:
10.         w(m̃) ← w(m̃) + min{+0.5, w(a')};
11.         w(b) ← max{0, w(a') − 0.5};
12.         visited(b) = true;
13.     case [a', b] ∈ M and visited(b) = false:
14.         w(b) ← w(a');
15.         visited(b) = true;
16.     case [a', b, c] ∈ T̃ and visited(b) = false and visited(c) = false:
17.         w(b) =← w(a')/2;
18.         w(c) =← w(a')/2;
19.         visited(b) ← true;
20.         visited(c) ← true;
21.     case [a', b, c] = t_i ∈ T:
22.         w(t_i) ← w(t_i) + w(a');
23.         visited(a') ← true;
24.         w(a') ← 0;
25. }
```

The positive weights of 1-cliques $f \in F$ are propagated as follows. In propagation the related induced graphs are also established for later analysis. For each 1-clique $f = [a]$ in F, we first transfer its weight to vertex a such that $w(f) = 0$ and $w(a) = +1$. There are five cases for further propagation for a vertex a' with a positive weight. That is, the propagation stops when no vertices of positive weights exist. We first claim that all visited vertices of G will not be visited again to resolve the case conflicts in the following. For the first case, if $[a']$ is a clique $\tilde{f} \in \tilde{F}$, the weight of vertex a' is charged to $\tilde{f}$ such that $w(\tilde{f}) = w(\tilde{f}) + w(a')$ and $w(a') = 0$. For the second case, if vertex a' is in a 2-clique $[a', b] = \tilde{m} \in \tilde{M}$, the weight of vertex a' is charged to $\tilde{m}$ up to $+0.5$ and the remaining weight of a' is charged to vertex b such that $w(\tilde{m}) = w(\tilde{m}) + \min\{+0.5, w(a')\}$, $w(b) = \max\{0, w(a') - 0.5\}$ and $w(a') = 0$. For the third case, if vertex a' is in a 2-clique $[a', b] \in M$, the weight is transferred to b such that $w(b) = w(a')$ and $w(a') = 0$. For the fourth case, if vertex a' is in a 3-clique $[a', b, c] \in \tilde{T}$, the weight of vertex a' is equally charged to vertices b and c such that $w(b) = w(c) = w(a')/2$ and $w(a') = 0$. For the last case, if vertex a' is in a 3-clique $t = [a', b, c] \in T$, the weight of vertex a' is

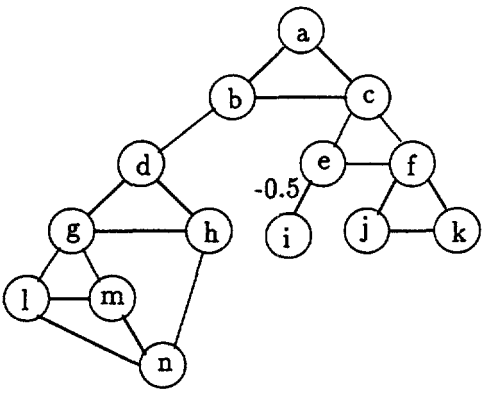

Fig. 6. A 4-clique-free graph and its minimum clique partition.

charged to t such that $w(t) = w(t) + w(a')$ and $w(a') = 0$. Note that in the above weight propagation vertices b and c should both be first visited. We shall show that no vertices can be visited twice or more. The process of propagating the weight of an $f \in F$ to other cliques is called a weight propagation process.

An example of this charge propagation is shown in Figs. 6–9. The graph in Fig. 6 is a 4-clique-free graph and its minimum clique partition such that the 2-clique $[e,i]$ is with a negative weight -0.5. The clique partition found by A4CFG is shown in Fig. 7 with negative weights on 3-clique $[g,l,m]$ and $[c,e,f]$ and positive weights on 1-cliques $[a]$ and $[i]$. We consider the charge graph $G_{[a]}$ induced by the 1-clique $[a]$ first. The weight of $[a]$ is transfered to vertex a first. The weight is distributed to vertices b and c with 0.5 each. Since vertex c is a vertex in the 3-clique $[c,e,f] \in T$, its weight is charged to $[c,e,f]$ and stops. For vertex b, its weight is transfered to vertex d since $[b,d]$ is a 2-clique in M. The weight of vertex d is equally distributed to vertices g and h with 0.25 each. The weight of vertex g is then charged to the 3-clique $[g,l,m]$ and stops. The weight of vertex h is transfered to vertex n and then further distributed to vertices l and m with 0.125 each. The weights of vertices l and m are then both charged to the 3-clique $[g,l,m]$ and stops. For the 1-clique $[i]$, since $[e,i]$ is a 2-clique in $\tilde{M}$, 0.5 of its weight is charged to $[e,i]$ and the other 0.5 is charged to vertex e. The weight of vertex e is then charged to the 3-clique $[c,e,f]$ and stops. The charge graphs $G_{[a]}$ and $G_{[i]}$ are shown in Fig. 8. The weight distribution before charged to cliques in P is shown in Fig. 9. The final weight of 3-clique $[g,l,m]$ is $-1.0 + w(g) + w(l) + w(m) = -0.5$. The final weight of 3-clique $[c,e,f] = -1.0 + w(c) + w(e) + w(f) = 0$. The final weight of 2-clique $[e,i]$ is $-0.5 + 0.5 = 0$.

By the charge scheme it can be seen that the total weight $w(P \cup \tilde{P})$ is unchanged after any weight propagation process.

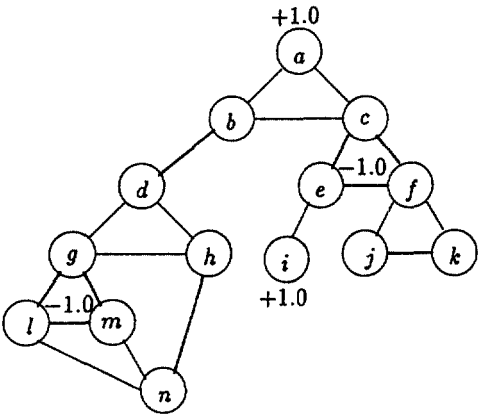

Fig. 7. The clique partition found by A4CFG.

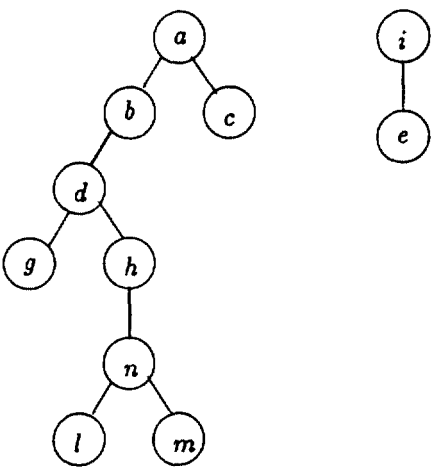

Fig. 8. Two charge graphs $G_{[a]}$ and $G_{[f]}$.

Lemma 4.2. *After each weight propagation process the total weight* $w(P \cup \tilde{P})$ *is unchanged.*

Proof. This is straightforward from the defined charge scheme in the weight propagation process. □

For each $f = [a] \in F$, all the passed vertices and edges form a charge graph G_f. Note that for the clique $[a', b, c]$ in the above, only edges (a', b) and (a', c) are passed.

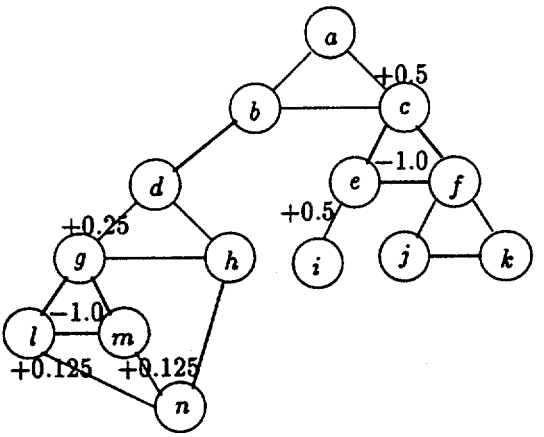

Fig. 9. Weight distribution before charged to cliques in P.

Definition 4.1. For each $f = [a] \in F$, the charge graph $G_f = (V_f, E_f)$ is defined such that V_f is the set all vertices that have been charged a positive weight and E_f is the set of all passed edges in the weight propagation process of f.

By the charge scheme the edges of a path in G_f is alternating between $\tilde{M}$ and M except that the path can end at a vertex in $m \in M$ or $t \in T$.

Lemma 4.3. *For any G_f of $f = [a]$, any path starting from vertex a is an M-alternating path except that the end vertex could be in a 2-clique $m \in M$ or a 3-clique $t \in T$.*

Proof. This is proved by induction on the length of the path. Let $p = a - x_1 - x_2 - \cdots - x_k$ for $k \geqslant 0$ be a path of G_f. It holds for the induction basis of $p = a$ since a is a free vertex. Assume that the above holds for $k = i$. For the induction step of $k = i + 1$, x_i cannot be a vertex in some 3-clique in T since such a vertex cannot transfer its weight to other vertices in the charge scheme and thus no further vertices are connected to it in the charge graph. If vertices x_{i-1} and x_i form a 2-clique in M, vertices x_i and x_{i+1} are in the same clique in $\tilde{M} \cup \tilde{T}$, that is, x_{i+1} is a vertex in a clique in $M \cup T$. Otherwise, vertex x_{i+1} is free and path p is M-augmenting, which contradicts Lemma 3.1. If vertices x_{i-1} and x_i do not form a 2-clique in M, vertex x_i is a vertex in M. Therefore, $[x_i, x_{i+1}]$ is a 2-clique in M and p is an M-alternating path. □

It is easy to see that the cliques that are both in M and $\tilde{M}$ or both in T and $\tilde{T}$ are not in any charge tree.

Lemma 4.4. *The vertices of the cliques in $M \cap \tilde{M}$ and $T \cap \tilde{T}$ are not in any charge graph G_f.*

Proof. Vertices in $M \cap \tilde{M}$ and $T \cap \tilde{T}$ cannot be charged positive weights since no weight can be transferred in by vertices in 2-cliques of M. □

If two charge graphs share a common vertex, there will exist an augmenting path. By Lemma 3.1 this cannot happen.

Lemma 4.5. *All charge graphs are vertex disjoint.*

Proof. Otherwise, assume that vertex x is in two charge graphs G_f and $G_{f'}$ with $f = [a]$ and $f' = [a']$ such that M-alternating paths $[a] - \cdots - y - x$ and $[a'] - \cdots - y' - x$ are vertex-disjoint. If $[y,x] \in M$ and $[y',x] \notin M$, path $[a] - \cdots - y - x - y' - \cdots - [a']$ is M-augmenting. If $[y,x] \notin M$ and $[y',x] \notin M$, clique $[x,y,y']$ must be in $\tilde{T}$. Thus, path $[a] - \cdots - y - y' - \cdots - [a']$ is M-augmenting. Both cases contradict Lemma 3.1. Therefore, no charge graphs share common vertices. □

We can actually show that a charge graph is a tree. Otherwise, there is a cycle in the charge graph such that the charge scheme will not end in charging positive weights of vertices to cliques.

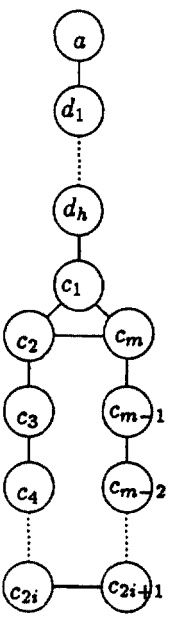

Fig. 10. A charge cycle $c_1 - c_2 - \cdots - c_m$.

Lemma 4.6. *For any $f \in F$, the charge graph G_f is a tree.*

Proof. We assume otherwise that G_f with $f = [a]$ is not a tree such that $c_1 - c_2 - \cdots - c_m - c_1$ in G_f is a cycle. By the charge scheme $[c_1, c_2, c_m]$ is a clique in $\tilde{T}$ and m is an odd number. Without loss of generality, we assume that c_1 is the closest vertex to the free vertex a among vertices $c_1, c_2, \ldots, c_m$, shown in Fig. 10. Therefore, by Lemma 4.3 the path from a to c_1 is M-alternating and 2-cliques $[c_{2i}, c_{2i+1}]$ for $1 \leqslant i \leqslant (m-1)/2$ are in final M. The execution of procedure A4CFG can be divided into two phases. The first phase finds a maximal 3-clique set T and then a maximum 2-clique set M. The second phase adjusts P by C-augmenting paths.

Since $[c_1, c_2, c_m]$ is in $\tilde{T}$, at least one of c_1, c_2 and c_m is a vertex in $t \in T$ found in the first phase. For the case that c_m is such vertex in $t = [c_m, y, z]$, since $[c_{m-1}, c_m]$ is a 2-clique in final M there is a C-augmenting path D passing c_{m-1} and c_m such that $[c_{m-1}, c_m]$ is not a 2-clique in M found in the first phase. If D is of C_2-type (the C_1-type case is similar), it is of form $[p] - q_1 - q_2 - \cdots - q_{2k} - c_{2j-1} - c_{2j} - \cdots - c_{m-1} - [c_m, y, z] = [x]$ for some free vertices p and x. For the case that c_2 is a vertex in some $t \in T$, it can be discussed similarly. For the case that c_1 is a vertex in some $t = [c_1, y, z] \in T$ found in the first phase. Let the M-alternating path from a to c_1 be $[a] - d_1 - d_2 - \cdots - d_h - c_1$, where h is odd. Analogously, there is a C-augmenting path D passing d_h and c_1 such that D is of form $[p] - q_1 - q_2 - \cdots - q_{2k} - d_{2j} - d_{2j+1} - \cdots - d_h - [c_1, y, z] = [x]$, where p and x are free vertices. Therefore, if G_f contains a cycle, there exists at least one C-augmenting path after the first phase.

We now classify the C_2-augmenting paths found in the second phase into the following 4-types.

1. Type-1: $[h_1] - m_1 - m_2 - \cdots - m_k - c_i - c_{i+1} - \cdots - c_{i+j} - r_1 - r_2 - \cdots - r_l - [t_1, t_2, t_3] = [h_2]$, which is shown in Fig. 11.
2. Type-2: $[h_1] - m_1 - m_2 - \cdots - m_k - c_i - c_{i-1} - \cdots - c_{i-j} - r_1 - r_2 - \cdots - r_l - [t_1, t_2, t_3] = [h_2]$, which is shown in Fig. 12.
3. Type-3: $[h_1] - m_1 - m_2 - \cdots - m_k - d_i - d_{i+1} - \cdots - d_{i+j} - r_1 - r_2 - \cdots - r_l - [t_1, t_2, t_3] = [h_2]$, which is shown in Fig. 13.
4. Type-4: $[h_1] - m_1 - m_2 - \cdots - m_k - d_i - d_{i-1} - \cdots - d_{i-j} - r_1 - r_2 - \cdots - r_l - [t_1, t_2, t_3] = [h_2]$, which is shown in Fig. 14.

We can define type-α C_1-augmenting paths similarly, where $1 \leqslant \alpha \leqslant 4$.

Suppose that there are β C-augmenting paths found in the second phase. Let $D_1, D_2, \ldots, D_\beta$ be the order that A4CFG finds them. At least one of them is Type-1, Type-2, or Type-3 as we discussed above. We have the following cases to discuss.

1. If D_β (the last one) is of Type-1, before adjusting by D_β the M-augmenting path $[h_1] - m_1 - m_2 - \cdots - m_k - c_i - c_{i-1} - \cdots - c_1 - d_h - d_{h-1} - \cdots - d_1 - [a]$ exists.

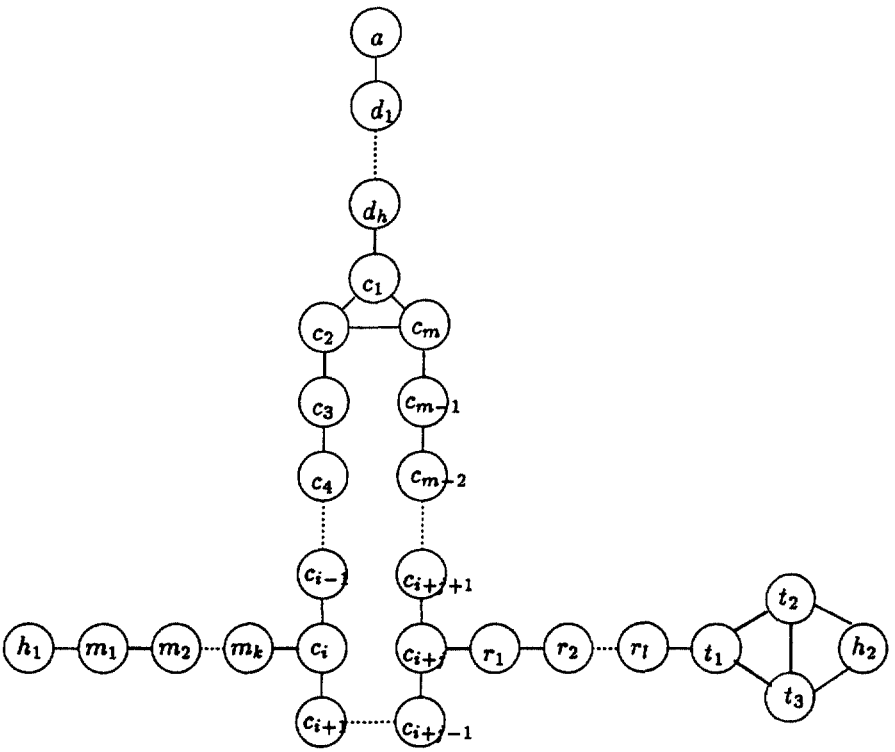

Fig. 11. Type-1 C_2 augmenting path.

2. If D_β is of Type-2, before adjusting by D_β the M-augmenting path $[h_1] - m_1 - m_2 - \cdots - m_k - c_i - c_{i+1} - \cdots - c_m - c_1 - d_h - d_{h-1} - \cdots - d_1 - [a]$ exists.

3. If D_β is of Type-3, before adjusting by D_β the M-augmenting path $[h_1] - m_1 - m_2 - \cdots - m_k - d_i - d_{i-1} - \cdots - d_1 - [a]$ exists.

All the above three cases contradict Lemma 3.1. If D_β is of Type-4, we use $[h_1]$ to play the role of $[a]$ and consider the C-augmenting paths $D_1, D_2, \ldots, D_{\beta-1}$ for the above three cases further. Then some of $D_1, \ldots, D_{\beta-1}$ is of Type-1, Type-2, or Type-3. We can show that an M-augmenting path exists also, which is a contradiction. Therefore, the assumption that G_f has a cycle is false. This completes the proof. □

Remark 4.1. In the above lemma, we consider simpler C-augmenting paths only. More complicated cases like those in Lemma 3.1 can be discussed also. However, those cases all results in M-augmenting paths, which is a contradiction similarly.

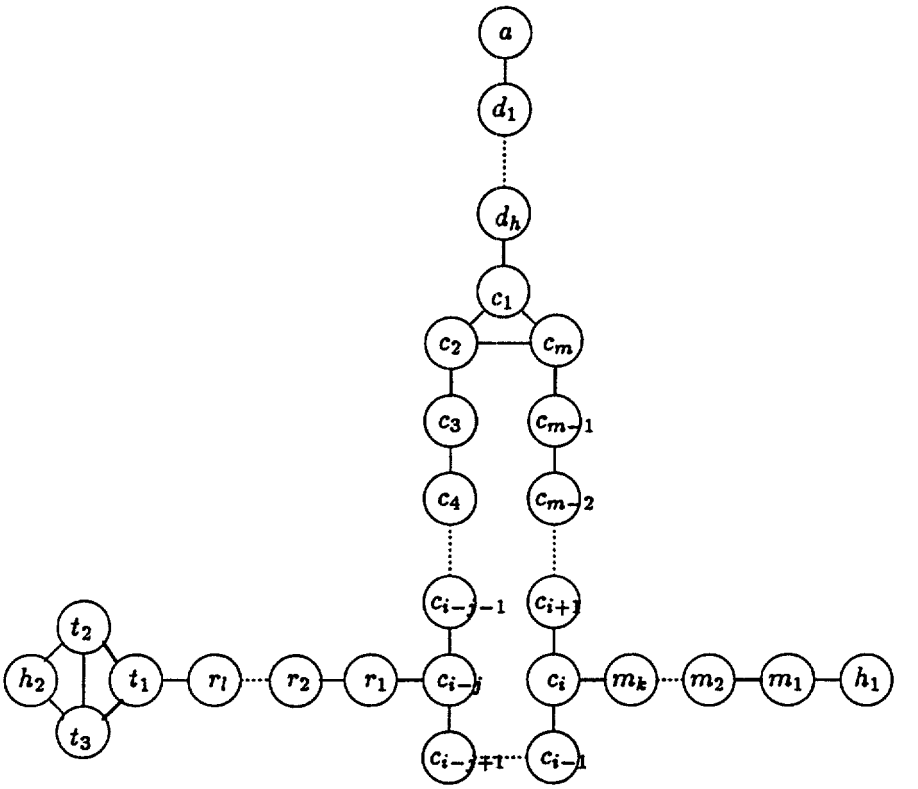

Fig. 12. Type-2 C_2 augmenting path.

We can now show that any 2-clique $\tilde{m}$ is charged at most once and thus has a final weight of at most 0.

Lemma 4.7. *For any 2-clique $\tilde{m} = [x, y] \in \tilde{M}$, vertices x and y cannot be in different charge trees.*

Proof. Otherwise, there is a 2-clique $\tilde{m} = [x, y]$ such that vertex x is in G_f and vertex y is $G_{f'}$ with $f = [a]$ and $f' = [a']$, where $a \neq a'$. By Lemma 4.3 path $[a] - \cdots - x - y - \cdots - [a']$ is M-augmenting, which is a contradiction by Lemma 3.1. □

Lemma 4.8. *After all weight propagation processes, the weight of any 2-clique $\tilde{m} \in \tilde{M}$ is $\leqslant 0$.*

Proof. Let $\tilde{m} = [a', b]$ be a 2-clique in $\tilde{M}$. If $\tilde{m}$ is not in any charge tree, its weight is -0.5 since its weight is not changed from the initial weight assign-

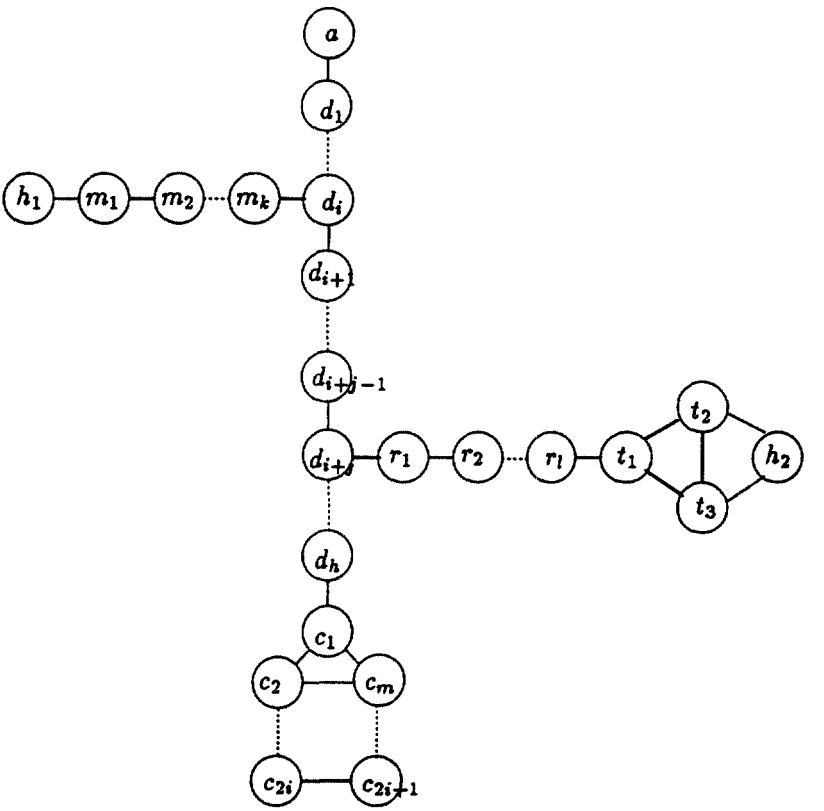

Fig. 13. Type-3 C_2 augmenting path.

ment. By Lemma 4.6 and Lemma 4.7, $\tilde{m}$ is in at most one charge tree G_f and $\tilde{m}$ can be charged from f at most once. By the second case of the charge scheme, $\tilde{m}$ is charged more weight one time only when vertex b is first visited. The charged weight is at most +0.5. Thus the final weight of $\tilde{m}$ is at most 0. □

It can be shown that the final weight of $t \in T$ is at most 0.

Lemma 4.9. *For any 3-clique $t = [x, y, z] \in T$, each vertex of x, y and z can be in at most one charge tree.*

Proof. This follows from Lemma 4.5 directly. □

Lemma 4.10. *After all weight propagation processes the weight of any 3-clique $t \in T$ is $\leqslant 0$.*

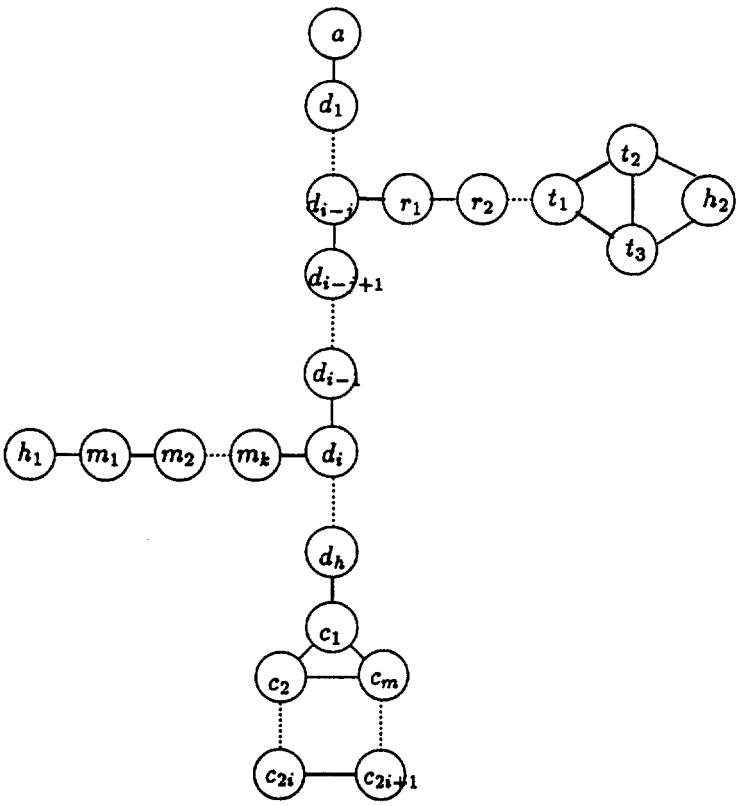

Fig. 14. Type-4 C_2 augmenting path.

Proof. Let $t = [a', b, c]$ be a 3-clique in T. By Lemma 4.9 each vertex in t is charged at most +0.5. On the other hand, if it is not charged +0.5, it is charged at most +0.25. The C_1- and C_2-augmenting paths are the only two cases such that the total charged weight of the three vertices in t is more than +1. However, by procedure A4CFG there are no C-augmenting paths with respect to the final P. Thus, the final weight of t is at most 0, which is the sum of the initial weight of t and the weights of the three vertices in t in the weight propagation processes. □

The final weight of any 1-clique in $\tilde{F}$ is at most 0.

Lemma 4.11. *After all weight propagation processes, the weight of any 1-clique* $\tilde{f} \in \tilde{F}$ *is* $\leqslant 0$.

Proof. By Lemma 4.6 any vertex a of $\tilde{f} = [a]$ can be in at most one charge tree. Therefore $\tilde{f}$ can be charged at most $+1$ more. Thus the final weight of $\tilde{f}$ is $\leqslant 0$. □

By the above lemmas the weights of cliques in $P \cup \tilde{P}$ are all $\leqslant 0$. Therefore the total weight of cliques in $P \cup P'$ is $\leqslant 0$.

Corollary 4.1. *After all weight propagation processes, the weight of every clique in $P \cup \tilde{P}$ is $\leqslant 0$.*

Proof. This follows from Lemmas 4.8, 4.10, and 4.11. □

The main result for procedure A4CFG is now proved.

Theorem 4.1. *Procedure A4CFG has the approximation ratio $\geqslant 3/4$.*

Proof. Let iw be the initial weights on cliques in $P \cup \tilde{P}$ and fw be the final weights on cliques in $P \cup \tilde{P}$ after all propagation processes. By Lemma 4.2 the total weight of all cliques after each propagation process is unchanged, that is,

$$iw(F) + iw(T) + iw(\tilde{F}) + iw(\tilde{M}) = fw(F) + fw(T) + fw(\tilde{F}) + fw(\tilde{M})$$

Therefore, by Corollary 4.1

$$fw(F) + fw(T) + fw(\tilde{F}) + fw(\tilde{M}) \leqslant 0. \qquad \square$$

Corollary 4.2. *The algorithm AUC finds a clique partition P for a graph G such that $(n - |P|)/(n - \phi(G)) \geqslant 3/4$.*

The ratio 3/4 is also the best that the procedure *A4CFG* can achieve.

Theorem 4.2. *There is a 4-clique-free graph G such that the procedure A4CFG has the approximation ratio 3/4.*

Proof. For the 4-clique-free graph in Fig. 15, A4CFG could find a clique partition $\{[a], [c,f], [b,d,e]\}$. However, the minimum clique partition is $\{[a,b,c], [d,e,f]\}$. The approximation ratio for this example is $(6-3)/(6-2) = 3/4$. □

Corollary 4.3. *The approximation ratio γ of algorithm AUC is 3/4.*

A consequence of this result is an algorithm with approximation ratio 1.5 for the minimum clique partition problem for 4-clique-free graphs, therefore for tripartite graphs.

Theorem 4.3. *There is a polynomial-time algorithm that finds a clique partition for 4-clique-free graphs such that the number of cliques in it is no more than 1.5 times that of the minimum clique partition.*

Proof. Let G be a 4-clique-free graph of n vertices and P is a clique partition found by A4CFG. Since $(n - |P|)/(n - \phi(G)) \geqslant 3/4$ and $\phi(G) \geqslant n/3$, $|P| \leqslant n/2 \leqslant 1.5\phi(G)$. □

Corollary 4.4. *There is a polynomial time algorithm that finds a clique partition for tripartite graphs such that the number of cliques in it is no more than 1.5 times that of the minimum clique partition.*

For graph coloring, if a graph has no independent set of size more than 3, its chromatic number can be approximated with ratio 1.5.

Theorem 4.4. *There is a polynomial-time algorithm that colors vertices of 4-independent-set-free graph such that the number of used colors is no more than 1.5 times that of its minimum coloring.*

Proof. This follows from Theorem 4.3 directly. □

5. Conclusion

We have presented a polynomial-time graph coloring algorithm of $\gamma = 3/4$, which improves the previous best result 2/3. It is natural to further improve this ratio, or to show that 3/4 is a bound.

In the direction to improve the ratio, we try to follow the technique used in this paper, that is, to find more complicated augmenting paths for adjustment. However, there are two obstacles. The first is that to find more complicated augmenting paths for a better ratio is not known to be polynomial-time solv-

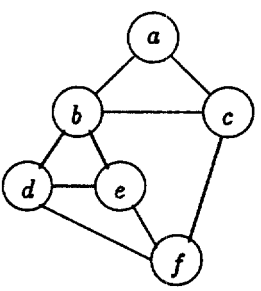

Fig. 15. An example with approximation ratio 3/4.

able. The second is for the analysis. We find no nice equation like the equation in Lemma 4.1, which is independent of n. This makes the analysis difficult.

Acknowledgements

This research was supported in part by National Science Council, Taiwan, ROC, under grant NSC-86-2213-E-009-024.

References

[1] S. Arora, S. Safra, Approximating clique is NP-Complete, in: Proc. of The 33rd IEEE Symp. on Foundations of Computer Science, 1992.
[2] A. Blum, Some tools for approximate 3-coloring, in: Proc. of The 31st IEEE Symp. on Foundations of Computer Science, 1990, pp. 554–562.
[3] M. Demange, P. Grisoni, V.T. Paschos, Approximation results for the minimum graph coloring problem, Information Processing Letters 50 (1994) 19–23.
[4] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[5] M. Garey, D. Johnson, L. Stockmeyer, Some simplified NP-Complete graph problems, Theoret. Comput. Sci. 1 (1976) 237–267.
[6] A. Gibbons, Algorithmic Graph Theory, Cambridge Univ. Press, Cambridge, 1985.
[7] R. Hassin, S. Lahav, Maximizing the number of unused colors in the vertex coloring problem, Information Processing Letters 52 (1994) 87–90.
[8] P. Peterson, M. Loui, The general matching algorithm of Micali and Vazirani, Algorithmica (1988) 511–533.
[9] C. Lund, M. Yanakakis, On the hardness of approximating minimization problems, in: Proc. 25th Annual ACM Symp. Theory of Computing, 1993, pp. 286–293.
[10] A. Widgerson, Improving the performance guarantee for approximate graph coloring, J. ACM 30 (1983) 729–735.
[11] C.K. Wong, L.T. Kou, L.J. Stockmeyer, Covering edges by cliques with regard to keyword conflicts and intersection graphs, Commun. ACM 21 (1978) 135–139.