Journal of Combinatorial Optimization, 9, 331–347, 2005

A Combinatorial Algorithm for Minimum Weighted Colorings of Claw-Free Perfect Graphs

XUELIANG LI*
Center for Combinatorics, Nankai University, Tianjin, China

WENAN ZANG†‡ wzang@maths.hku.hk
Department of Mathematics, University of Hong Kong, Hong Kong, China

Received May 30, 2004; Revised February 26, 2005; Accepted March 8, 2005

Abstract. We present an $O(n^5)$ combinatorial algorithm for the minimum weighted coloring problem on claw-free perfect graphs, which was posed by Hsu and Nemhauser in 1984. Our algorithm heavily relies on the structural descriptions of claw-free perfect graphs given by Chavátal and Sbihi and by Maffray and Reed.

Keywords: perfect graph, graph coloring, network flow, algorithm, complexity

1. Introduction

Given a graph $G = (V, E)$ with a nonnegative integral weight $w(v)$ on each vertex v, the *minimum weighted coloring problem* is to find stable sets $S_1, S_2, \ldots, S_t \subseteq V$ and nonnegative integers $x(S_1), x(S_2), \ldots, x(S_t)$ such that for each vertex $v \in V$ the sum $\sum_{v \in S_i} x(S_i) \geq w(v)$ and that $\sum_{i=1}^{t} x(S_i)$ is as small as possible, where a *stable set* is a set of pairwise nonadjacent vertices. As is well known, this problem is *NP*-hard (Garey and Johnson, 1979) even when $w(v) = 1$ for all $v \in V$, so there is no polynomial-time algorithm for solving it exactly unless $NP = P$. The present paper is devoted to the minimum weighted coloring problem on claw-free perfect graphs.

Let us introduce some notions before presenting the problem. A *claw* is the complete bipartite graph $K_{1,3}$; a graph is called *claw-free* if none of its induced subgraphs is a claw. Berge proposed to call a graph *perfect* if, for each of its induced subgraphs H, the chromatic number of H equals the largest number of pairwise adjacent vertices in H. He also made the so-called *strong perfect graph conjecture*: a graph is perfect iff it contains neither a chordless odd cycle of length at least five nor the complement of such a cycle. This conjecture has attracted tremendous research efforts and been confirmed in various cases, including claw-free graphs (Parthasarathy and Ravindra, 1976); see Berge and Chvátal (1984) and

*Supported by the National Science Foundation of China.
†Supported by the Research Grants Council of Hong Kong (Project No. HKU 7109/01P) and Seed Funding for Basic Research of HKU.
‡Corresponding author.

Ramírez-Alfonsín and Reed (2001) for comprehensive results on perfect graphs. Recently, managed to make a final breakthrough and successfully proved this important conjecture completely.

In addition to the conjecture, the minimum weighted coloring, maximum weighted clique, minimum weighted clique cover, and maximum weighted stable set problems on perfect graphs are also of great significance and have been subjects of extensive research. Yet, the proof by Chudnovsky et al. (to appear) does not yield efficient algorithms for these problems. A fundamental result in this direction is due to Grötschel et al. (1984), who devised polynomial-time algorithms for all these optimization problems; their algorithms are ingenious variations on the celebrated ellipsoid method for linear programming, and therefore very much unlike the typical combinatorial optimization procedures. Since these algorithms are not practically efficient, it will be nice to have polynomial-time algorithms with a more transparent combinatorial nature for all perfect graphs or even subclasses of perfect graphs. In a series of papers (Hsu, 1981; Hsu and Nemhauser, 1984; Hsu and Nemhauser, 1981; Hsu and Nemhauser, 1982), Hsu and Nemhauser obtained efficient combinatorial algorithms for these optimization problems on claw-free perfect graphs except for the minimum weighted coloring, which was posed as an open problem (Hsu and Nemhauser, 1984); the purpose of this paper is to present an $O(n^5)$ combinatorial algorithm for solving it exactly. Our algorithm heavily relies on the structural descriptions of claw-free perfect graphs given by Chvátal and Sbihi (1988) and Maffray and Reed (1999).

The following terms are involved in their descriptions. A *clique* in a graph is a set of pairwise adjacent vertices; a *clique-cutset* of a graph is a clique whose removal disconnects the graph. Chvátal and Sbihi (1988) proposed to call a graph elementary if its edges can be colored by two colors in such a way that every chordless path on three vertices has its two edges colored differently. They also called a graph *peculiar* if it can be obtained as follows: begin with a complete graph K whose set of vertices is split into pairwise disjoint nonempty sets $A_1, B_1, A_2, B_2, A_3, B_3$. Then, for each $i = 1, 2, 3$, remove at least one edge with one endpoint in A_i and the other endpoint in B_{i+1} (the subscript 4 is interpreted as 1). Finally, add pairwise disjoint nonempty cliques K_1, K_2, K_3 and, for each $i = 1, 2, 3$, make each vertex in K_i adjacent to all the vertices in $K - (A_i \cup B_i)$. Chvátal and Sbihi obtained the following characterization of claw-free perfect graphs.

Theorem 1.1 (Chvátal and Sbihi (1988))**.** *If a claw-free perfect graph has no clique-cutset, then it is either elementary or peculiar.*

The structure of peculiar graphs is completely determined by their definition, however it is not so for elementary graphs. A complete structural description of elementary graphs can be found in Maffray and Reed (1999), which is based on the following definitions. A graph is *cobipartite* if it is the complement of a bipartite graph. An edge is *flat* if it does not lie in a triangle. The neighborhood of a vertex v is denoted by $N(v)$.

Definition 1.1 (Augmenting a Flat Edge). Let xy be a flat edge of a graph $G = (V, E)$ and let $B = (X, Y; E_{XY})$ be a connected cobipartite graph disjoint from G. We can obtain a new graph G' from $G\{x, y\}$ and B by adding all possible edges between X and $N(x)\backslash y$

and between Y and $N(y)\backslash x$. We say that (in the construction of G') graph G is augmented along xy, edge xy is augmented, and B is the augment of xy.

Definition 1.2 (Augmenting a Matching of Flat Edges). Let $G = (V, E)$ be a graph, let $x_1y_1, x_2y_2, \ldots, x_ky_k$ be k pairwise non-incident flat edges of G, and let $(X_1, Y_1; E_1), (X_2, Y_2; E_2), \ldots, (X_k, Y_k; E_k)$ be k pairwise disjoint connected cobipartite graphs that are also disjoint from G. We can obtain a new graph G' by augmenting each edge x_{iyi} with the augment $(X_i, Y_i; E_i)$, respectively. (This graph is the same whatever the order in which the augments are done.) We call G' an augmentation of G.

Theorem 1.2 (Maffray and Reed (1999)). *A graph is elementary iff it is an augmentation of the line-graph of a bipartite multigraph.*

We shall appeal to the above structural description to find minimum weighted colorings of elementary graphs in our algorithm. It is worthwhile pointing out that (X_i, Y_i) in Definition 1.2 is a special homogeneous pair as defined in Chvátal and Sbihi (1987), where Chvátal and Sbihi proved that no minimal imperfect graph contains a homogeneous pair by using a beautiful reduction method. Since the reduced graph H in their proof contains a claw formed by u_1, v_1, v_2 and a Q_2-universal Q_1-null vertex (see page 131 of Chvátal and Sbihi (1987)), their method does not directly apply to the problem in our consideration, and we shall develop it to fulfill our needs.

We are now ready to outline our algorithm. Let G be the input claw-free perfect graph. If G contains no clique-cutset, then G is either an elementary graph or a peculiar graph by Theorem 1.1; else, let C be a clique-cutset of G and let A be the vertex-set of one component of $G\backslash C$. We view $G_1 = G[A \cup C]$, $G_2 = [V\backslash A]$ as a decomposition of G, where $G[U]$ stands for the subgraph of G induced by all the vertices in U. If any of these smaller graphs admits a clique-cutset, then the decomposition procedure can be continued. So we obtain a decomposition tree whose leaves are elementary graphs or peculiar graphs; such a tree can be constructed in polynomial time, see, for instances, Gavril (1985), Whitesides (1981), and Tarjan (1985). We then produce a minimum weighted coloring for each leaf, based on the definition of peculiar graphs and the structural description of elementary graphs featured in Theorem 1.2. Finally, we piece together all these colorings along clique-cutsets to get a minimum weighted coloring of the entire graph G.

2. Preliminaries

Let S be the collection of all stable sets of G. Clearly the minimum weighted coloring problem can be formulated as the following integer program:

$$\begin{aligned} \min \ & \sum_{S\in\mathcal{S}} x(S) \\ \text{s.t.} \ & \sum_{v\in s} x(S) \geq w(v) \quad \forall v \in V \\ & x(S) \geq 0, \text{ integral} \qquad \forall S \in \mathcal{S}. \end{aligned} \tag{1}$$

To tackle this problem, we may resort to its dual (Chvátal, 1983; Schrijver, 1986), the *maximum weighted clique problem*, which is to find a clique with maximum total weight:

$$\begin{aligned} \max \quad & \sum_{v \in v} w(v)y(v) \\ \text{s.t.} \quad & \sum_{v \in \mathcal{S}} y(v) \leq 1 \qquad \forall S \in \mathcal{S} \\ & y(v) \geq 0, \text{ integral} \qquad \forall v \in v \end{aligned} \tag{2}$$

Remark. By the duality theorem, if a feasible weighted coloring and a clique of G have the same total weight, then both of them are optimal. This simple observation will be used repeatedly later.

Let $\chi(G, w)$ and $\omega(G, w)$ stand for the optimal values of (1) and (2), respectively. The following statement was established by Lovász.

Theorem 2.1 (Lovász (1972))**.** *Let $G = (V, E)$ be a perfect graph. Then $\chi(G, w) = \omega(G, w)$ for any non-negative integral weight function w defined on the vertex set V.*

For convenience, we shall use the following equivalent form of the minimum weighted coloring problem hereafter.

Definition 2.1. The minimum weighted coloring problem is equivalent to finding stable sets $S_1, S_2, \ldots, S_t \subseteq V$ and nonnegative integers $x(S_1), x(S_2), \ldots, x(S_t)$ such that for each vertex $v \in V$ the sum $\sum_{v \in S_i} x(S_i) = w(v)$ and that $\sum_{i=1}^{t} x(S_i)$ is minimized.

To devise the algorithm for the minimum weighted coloring problem on claw-free perfect graphs, we shall make use of algorithms for some other problems. It is well known that the maximum weighted stable set problem on bipartite graphs can be solved using the network flow technique. Nevertheless, we cannot find the proper origin for this algorithm, nor can we find an existing algorithm for the minimum weighted clique cover problem on bipartite graphs, so we present the following algorithm simply for completeness.

(2.1) Algorithm for finding a minimum weighted coloring and a maximum weighted clique in a cobipartite graph $G = (X, Y; E)$ with a nonnegative integral weight $w(v)$ on each vertex v.

Step 1. Construct a flow network N with vertex-set $V = X \cup Y \cup \{s, t\}$, where s is the source and t is the sink. For each nonadjacent vertex-pair $\{i, j\}$ in G with $i \in X$ and $j \in Y$, create an arc (i, j) in N with capacity ∞; for each $i \in X$, there is an arc (s, i) with capacity $w(i)$; and for each $j \in Y$, there is an arc (j, t) with capacity $w(j)$. There is no other arc in N.

Step 2. Find an integral maximum $s - t$ flow f and a minimum $s - t$ cut $(U, \bar{U})$ in the network N.

Step 3. For each pair $\{i, j\}$ with $i \in X$, $j \in Y$ and $f(i, j) > 0$, define $S_{ij} = \{i, j\}$ and $x(S_{ij}) = f(i, j)$; for each $i \in X \cap U$ with $\sum_{j \in Y} f(i, j) < w(i)$, define $S_i = \{i\}$ and $x(S_i) = w(i) - \sum_{j \in Y} f(ij)$; and for each $j \in Y \cap \bar{U}$ with $\sum_{i \in x} f(i, j) < w(j)$, define $S_j = \{j\}$ and $x(S_j) = w(j) - \sum_{i \in x} f(i, j)$, stop: all these S_{ij}, S_i, S_j together with $x(S_{ij})$, $x(S_i)$, $x(S_j)$ form a minimum weighted coloring of G and $C = (X \cap U) \cup (Y \cap \bar{U})$ is a maximum weighted clique of G.

Lemma 2.1. *Algorithm* (2.1) *correctly finds a minimum weighted coloring of a cobipartite graph* $G = (X, Y; E)$ *with* $O(n^2)$ *stable sets and a maximum weighted clique of* G *in* $O(n^3)$ *time, where* $n = |X| + |Y|$.

Proof: Since the capacity of each arc in N is integer-valued, there is an integral maximum flow f as described in Step 2. Let $(U, \bar{U})$ be the minimum cut output in Step 2. Observe that there is no arc between $X \cap U$ and $Y \cap \bar{U}$ in N, for otherwise such an arc has capacity ∞ and therefore $\text{cap}(U, \bar{U})$ would be ∞, a contradiction. So $\text{cap}(U, \bar{U}) \sum_{i \in x \cap \bar{U}} w(i) + \sum_{j \in Y \cap U} w(j)$, and $C(X \cap U) \cup (Y \cap \bar{U})$ is a clique of G. According to the max-flow min-cut theorem (Ford, Jr. and Fulkerson, 1956) and the above observation, $val(f) = cap(U, \bar{U})$, $f(s, i) = w(i)$ for each $i \in X \cap \bar{U}$, $f(j, t) = w(j)$ for each $j \in Y \cap U$, and $f(i, j) = 0$ for each $i \in X \cap \bar{U}$ and $j \in Y \cap U$. Since flow f is conserved at each vertex in $X \cap Y$, for each i in $X \cap \bar{U}$ and each i in $X \cap U$ for which S_i is undefined in Step 3, we have $\sum_{j \in S_{ij}} x(S_{ij}) f(i, j) = \sum_{j \in Y} f(s, i) = w(i)$. For the remaining i in $X \cap U$, it follows from the definition of S_i that $x(S_i) + \sum_{j \in S_{ij}} x(S_{ij}) = w(i)$. Similarly, we have $\sum_{j \in s} x(S) = w(j)$ for each j in Y, where S ranges over all stable sets output by the algorithm. So S_{ij}, S_i, S_j, and $x(S_{ij})$, $x(S_i)$, $x(S_j)$ form a feasible weighted coloring of G (recall Definition 2.1).

It is easy to see that $\sum x(S_{ij}) + \sum_{i \in x} x(S_i) + \sum_{j \in Y} x(S_j) = \sum_{i \in x \cap U} w(i) + \sum_{j \in Y \cap \bar{U}} w(j) = w(C)$, so from the remark above Theorem 2.1 we deduce that S_{ij}, S_i, S_j (with $O(n^2)$ stable sets in total) and $x(S_{ij})$, $x(S_i)$, $x(S_j)$ form a minimum weighted coloring of G and C is a maximum weighted clique of G.

The running time of the algorithm is dominated by that of Step 2, which is $O(n^3)$ (see, for instances, Karzanov (1974) and Tarjan (1983)), completing the proof. □

Each of the following Algorithms (2.2)–(2.5) can be implemented more sophisticatedly to achieve better computational complexity, yet we present them in the current form for simplicity rather than superiority, since the complexity of our main algorithm is dominated by something else.

Given a multigraph $G = (V, E)$ with a nonnegative integral weight $w(e)$ on each edge e, the *minimum weighted edge-coloring problem* is to find matchings $M_1, M_2, \ldots, M_t \subseteq E$ and nonnegative integers $x(M_1), x(M_2), \ldots, x(M_t)$ such that for each edge $e \in E$ the sum $\sum_{e \in M_i} x(M_i) = w(e)$ and that $\sum_{i=1}^{t} x(M_i)$ is as small as possible.

The basic idea of the following algorithm is taken from Gonzalez and Sahni 1976. Since their algorithm is described for bipartite graphs (rather than bipartite multigraphs) and since the number of matchings involved is not stated explicitly over there (which is also crucial to the design of our algorithm), we reproduce their algorithm in the current form for ease of reference.

(2.2) Algorithm for finding a minimum weighted edge-coloring of a bipartite multigraph $G = (X, Y; E)$ with a nonnegative integral weight $w(e)$ on each edge e.

Step 1. Set $\delta(v) = \sum_{v \in e \in E} w(e)$ for each vertex $v \in X \cup Y$ and $\Delta = \max_{v \in X \cup Y} \delta(v)$. If $\delta(v) = \Delta$ for each vertex $v \in X \cup Y$, set $H = G$; else, let $G' = (X', Y'; E')$ be a copy of G and let H be the graph obtained from the disjoint union of G and G' by adding edges vv' for all vertices v with $\delta(v) < \Delta$ and set the weight $w(vv') = \Delta - \delta(v)$. Denote $H = (A, B; F)$. (Note that in the latter case) $(A, B) = (X \cup Y', Y \cup X')$ and that $\Delta = \sum_{v \in e \in F} w(e)$ (for each vertex v of H.) Set $t = 1$.

Step 2. Find a perfect matching M_i in $H = (A, B; F)$ by the Hopcroft-Karp algorithm (Hopcroft and Karp, 1973). Set $x(M_t) = \min_{e \in M_t} w(e)$. For each edge $e \in M_t$, set $w(e) = w(e) - x(M_t)$. Set $F = F \backslash \{e : w(e) = 0\}$.

Step 3. If $F \neq \emptyset$, set $t = t + 1$, return to Step 2; else, for $i = 1, 2 \ldots, t$, set $M_i' = M_i \cap E$ and set $x(M_i') = x(M_i)$ stop: $M_1', M_2', \ldots, M_t'$ together with $x(M_1), x(M_2'), \ldots, x(M_t')$ form a minimum weighted edge-coloring of G.

Lemma 2.2. *Algorithm* (2.2) *correctly finds a minimum weighted edge-coloring of G with O(m) matchings in* $O(m^2\sqrt{n})$ *time, where* $n = |X| + |Y|$ *and* $m\ |E|$.

Proof: For each $U \subseteq A$, let $N(U)$ denote the set of all neighbors of U in B at the beginning of the algorithm. Since $\sum_{v \in e \in F} w(e) = \Delta$ holds for each vertex v in H, we have $\Delta|U| = \sum_{v \in U} \sum_{v \in e \in F} w(e) \leq \sum_{v \in e \in N(U)} \sum_{v \in e \in F} w(e) = \Delta|N(U)|$, which implies that $|N(U)| \geq |U|$. In particular, we have $|A| \leq |B|$ and thus equality must hold by symmetry. It follows from Hall's theorem that H has a perfect matching M_1.

At each of the subsequent iterations, from the algorithm it can be seen that $\sum_{u \in e \in F} w(e) = \sum_{v \in e \in F} w(e)$ for any two vertices u and v in H. So the same reasoning implies that $H = (A, B; F)$ has a perfect matching at each iteration and thus the validity of Step 2 follows.

At the end of the algorithm, $F = \emptyset$, so for each edge e of H we have $w(e) = \sum_{e \in M_i} x(M_i)$. Hence for $e \in E$, there holds $\sum_{e \in M_i'} x(M_i') = \sum_{e \in M_i} x(M_i) = w(e)$, and therefore $M_1', M_2', \ldots, M_t'$ together with $x(M_1'), x(M_2'), \ldots, x(M_t')$ form a feasible edge-coloring of G. Furthermore, the coloring is of minimum weight as $\sum_{i=1}^{t} x(M_i') = \sum_{i=1}^{t} x(M_i) = \Delta = \delta(v)$ for some vertex v of G.

Notice that at each iteration we delete at least one edge from F, so t is bounded above by the initial $|F| = O(m)$. Since the Hopcroft-Karp algorithm (Hopcroft and Karp, 1973) takes $O(|E(H)|\sqrt{|V(H)|}) = O(m\sqrt{n})$ time to find a perfect matching in H, the running time of the algorithm is $O(tm\sqrt{n}) = O(m^2\sqrt{n})$, as asserted. □

Given a multigraph H, its *line graph* $L(H)$ is the graph whose vertices are the edges of H and whose edges are the pairs of incident edges of H. By convention, we call H a *root* of its line-graph. Several authors (Degiorgi and Simon, 1995; Lehot, 1974; Roussopoulos, 1973) have given liner-time algorithms to test if a graph G is the line graph of a simple graph and, if it is, to produce the root graph H. Moreover, the root is unique (with the only exception of $G = K_3$ with either $H = K_3$ and $H = K_{1,3}$). So such algorithms can be used to test line graphs of bipartite graphs. In Maffray and Reed (1999), Maffray and Reed fully

characterized the line-graphs of bipartite-multigraphs in terms of forbidden structures; they also suggested an algorithm for constructing the roots of these graphs. Since they did not give the complexity analysis of this algorithm, we fill in some details here.

(2.3) Algorithm for constructing a root H of the line-graph $G = (V, E)$ of a bipartite multigraph.

Step 1. For each $v \in V$, find all the maximal cliques in the induced subgraph $G|N(v) \cup \{v\}]$. Let C_v denote the set of all these cliques and let $f(v) = |C_v|$. For each clique $C \in C_v$, set $g(C) = |\{v \in C : f(v) = 1\}|$. Set $\mathcal{C} = \cup_{v \in V} \mathcal{C}_v$.

Step 2. For each $A \in \mathcal{C}$, create a vertex x_A in H. For any two members A, B of $\mathcal{C}$, put $|A \cap B|$ parallel edges between x_A and x_B. For each $A \in C$ with $g(A) > 0$, add one additional vertex y_A in H and $g(A)$ parallel edges between y_A and x_A, stop: the resulting graph is H.

Lemma 2.3. *Algorithm* (2.3) *correctly finds a root H of G in $O(n^3)$ time, where $n = |V|$.*

Proof: The correctness of the algorithm was established by Maffray and Reed 1999. Let us now analyze the time complexity of the algorithm. Since G is the line-graph of a bipartite multigraph, $G[N(v) \cup v]$ in Step 1 is either a maximal clique or the union of two maximal cliques A and B such that $A \backslash B$ and $B \backslash A$ are nonadjacent. So by enumerating all the vertex pairs in $G[N(v) \cup \{v\}]$, we can determine in $O(n^2)$ time which case occurs, and in the latter case, output A, B and $|A \cap B|$. Thus Step 1 takes $O(n^3)$ time. As $f(v) \leq 2$ for each vertex v, we have $|\mathcal{C}| \leq 2n$, and therefore the running time of the whole algorithm is dominated by that of Step 1. □

(2.4) Algorithm for coloring the line-graph $G = (V, E)$ of a bipartite multigraph such that each vertex v of G is associated with a nonnegative integral weight $w(v)$.

Step 1. Apply Algorithm (2.3) to find a bipartite multigraph H such that $G = L(H)$.

Step 2. For each edge e of H, define $w(e) = w(v)$, where v is the vertex of G corresponding to e. Apply Algorithm (2.2) to find a minimum weighted edge-coloring ϕ of H with respect to w. Suppose θ consists of a collection of matchings $M_1, M_2, \ldots, M_t$ and nonnegative integers $x(M_1), x(M_2), \ldots, x(M_t)$. Define S_i to be the set of all vertices in G corresponding to edges in M_i and $x(S_i) = x(M_i)$, for $i = 1, 2, \ldots, t$, stop: $S_1, S_2, \ldots, S_t$ together with $x(S_1), x(S_2), \ldots, x(S_t)$ form a minimum weighted coloring of G

Lemma 2.4. *Algorithm* (2.4) *correctly finds a minimum weighted coloring of G with $O(n)$ stable sets in $O(n^3)$ time, where $n = |V|$.*

Proof: The correctness of the algorithm is obvious. By Lemma 2.3, Algorithm (2.4) takes $O(n^3)$ time to find the bipartite multigraph H in Step 1. By Lemma 2.2, Algorithm (2.2) takes $O(|E(H)|^2\sqrt{|V(H)|}) = O(n^{2.5})$ time to produce a minimum weighted edge-coloring of H with $O(|E(H)|) = O(n)$ matchings in Step 2 as $|E(H)| = |V(G)| = n$

and $|V(H)| \leq |E(H)| + 1$. So the whole algorithm takes $O(n^3)$ time to find a minimum weighted coloring of G with $O(n)$ stable sets. □

(2.5) Algorithm for recognizing a peculiar graph $G = (V, E)$.

Step 1. Find three pairwise nonadjacent vertices v_1, v_2, v_3 in G. If such three vertices are unavailable, stop: G is not peculiar; else, go to Step 2.

Step 2. (In this step the subscripts are taken modulo 3.) For $i = 1, 2, 3$, set $K_i = \{v_i\} \cup N(v_i)\backslash(N(v_{i+1}) \cup N(v_{i+2}))$, $C_i = N(v_{i+1}) \cap N(v_{i+2})$. Partition C_i into any two subsets A_i and B_i such that $C_{i+2} \subseteq N(u)$ for each $u \in A_i$ and that $C_{i+1} \subseteq N(v)$ for each $v \in B_i$, stop: G is peculiar if K_i, A_i, B_i, for $i = 1, 2, 3$, satisfy all the properties described in the definition of a peculiar graph, and not peculiar otherwise.

Lemma 2.5. *Algorithm* (2.5) *correctly recognizes a peculiar graph G in $O(n^3)$ time, where $n = |V|$.*

Proof: From the definition of a peculiar graph, it is can be seen that if it has three pairwise nonadjacent vertices v_1, v_2, v_3, then we must have $v_i \in K_i$ (rename the subscripts if necessary) for $i = 1, 2, 3$. Thus K_i and C_i (which equals $A_i \cup B_i$), for $i = 1, 2, 3$, are uniquely determined and precisely the same as defined in the algorithm. It is easy to see that if G is a peculiar graph, then the partition of C_i into A_i and B_i as described in Step 2 will do, and hence the correctness of the algorithm follows.

Since it takes $O(n^3)$ time to get v_1, v_2, v_3 in Step 1 by enumeration method (actually there exists some much faster algorithm for finding a triangle in a graph, but $O(n^3)$ is good enough for the algorithm designed in the next section), the running time of the algorithm is $O(n^3)$. □

3. The main algorithm

Recall that the basic idea of our main algorithm is to decompose a claw-free perfect graph into peculiar graphs and elementary graphs using clique-cutsets, then we turn to color each of these primitive graphs, and finally piece together all these colorings to get a desired coloring of the input claw-free perfect graph.

(3.1) Algorithm for a minimum weighted coloring of a peculiar graph $G = (V, E)$ such that each $v \in V$ is associated with a nonnegative integer $w(v)$.

Step 1. Produce A_i, B_i, K_i, for $i = 1, 2, 3$, in G using Algorithm (2.5).

Step 2. (In this step the subscripts are taken modulo 3.) For $i = 1, 2, 3$, find a minimum weighted coloring ϕ_1 for each cobipartite graph (A_i, B_{i+1}) using Algorithm (2.1). Let $\mathcal{T}$ denote the collection of all the stable sets generated in ϕ_1, ϕ_2, ϕ_3. (Recall that there is a nonnegative integer $x(S)$ corresponding to each $S \in \mathcal{T}$.) Set $j = 1$.

Step 3. If $j \geq 4$, go to Step 5. Else, if there exist $u \in K_j$ and $v \in A_j \cup B_j$ such that $\{v\} \in \mathcal{T}$ set $S = \{u, v\}$, $x(S) = \min\{w(u), x(\{v\})\}$, $w(u) - x(S)$, and $x(\{v\}) = x(\{v\}) - x(S)$.

Set $K_j = K_j \backslash u$, if $w(u) = 0$, and set $\mathcal{T} = \mathcal{T} \cup \{S\}$ if $x(\{v\}) \neq 0$ and $(\mathcal{T} \backslash \{\{v\}\}) \cup \{S\}$ otherwise. Repeat Step 3 until $K_j = \emptyset$ or there is no $v \in A_j \cup B_j$ such that $\{v\} \in \mathcal{T}$.

Step 4. Set $j = j + 1$, go back to Step 3.

Step 5. If $K_1 \cup K_2 \cup K_3 = \emptyset$, stop: $\mathcal{T}$ together with $x(S)$ or each $S \in \mathcal{T}$ form a minimum weighted coloring of G. Else, let S be a stable set formed by picking one vertex u_i from each nonempty K_i, $1 \leq i \leq 3$. Set $x(S) = \min_{u_i \in S} w(u_i)$ and $\mathcal{T} = \mathcal{T} \cup \{S\}$. For each $u_i \in S$, set $w(u_i) - x(S)$ and set $K_i = K_i \backslash u_i$ if $w(u_i) = 0$. Repeat Step 5.

Lemma 3.1. *Algorithm* (3.1) *correctly finds a minimum weighted coloring of a peculiar graph* $G = (V, E)$ *with* $O(n^2)$ *stable sets in* $O(n^3)$ *time, where* $n = |V|$.

Proof: Let us consider Step 2 first. For $i = 1, 2, 3$, let C_i be a maximum weighted clique in the cobipartite graph (A_i, B_{i+1}), let $\mathcal{T}_i$ be the collection of all the stable sets produced in ϕ_i, and let $x(S)$ be the nonnegative integer corresponding to each $S \in \mathcal{T}_i$. By Definition 2.1 and Theorem 2.1, we have $\sum_{v \in S \in \mathcal{T}_i} x(S) = w(v)$ for each $v \in A_i \cup B_{i+1}$ and $\sum_{s \in \mathcal{T}_i} x(S) = w(C_i)$. Note that $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ at the current step. Let S_i be the collection of all stable sets S in $\mathcal{T}$ such that $S = \{v\}$ for some $v \in A_i \cup B_i$. Then $S_i \subseteq \mathcal{T}_i \cup \mathcal{T}_{i+2}$. Let us now make some simple observations.

(a) $\sum_{S \in \mathcal{S}_i} x(S) = w(C_i) + w(c_{i+2}) - w(B_{i+1}) - w(A_{i+2})$.

Indeed, since (A_i, B_{i+1}) is a cobipartite graph, $\sum_{S \in \mathcal{S}_i \cap \mathcal{T}_i} x(S) = \sum_{S \in \mathcal{T}_i} x(S) - \sum_{S \in \mathcal{T}_i \backslash \mathcal{S}_i} x(S) = w(C_i) - w(B_{i+1})$. Similarly, we have $\sum_{S \in \mathcal{S}_i \cap \mathcal{T}_{i+2}} x(S) = w(C_{i+2}) - w(A_{i+2})$. From $\sum_{S \in \mathcal{S}_i} x(S) = \sum_{S \in \mathcal{S}_i \cap \mathcal{T}_i} x(S) + \sum_{S \in \mathcal{S}_i \cap \mathcal{T}_{i+2}} x(S)$, statement (a) follows.

For $i = 1, 2, 3$, let $\bar{K}_i$ denote the K_i at the end of Step 4 and let $\bar{w}(u)$ denote the then $w(u)$ for each $u \in \bar{K}_i$. From Step 3 it can be seen that

(b) $\bar{w}(\bar{K}_i) = \max\{w(K_i) - \sum_{S \in \mathcal{S}} x(S), 0\}$.

Clearly $\sum_{S \in \mathcal{T}} x(S)$ is unchanged throughout Step 3, so at the end of the algorithm $\sum_{S \in \mathcal{T}} x(S)$ is the sum of $\sum x(S)$ (S is produced in Step 2; this sum equals $w(C_1) + w(C_2) + w(C_3)$) and $\sum x(S)$) (S is produced in Step 5; this sum equals $\max\{\bar{w}(\bar{K}_1), \bar{w}(\bar{K}_2), \bar{w}(\bar{K}_3)\}$). Hence for the output of the algorithm, we have

(c) $\sum_{S \in \mathcal{T}} x(S) = w(C_1) + w(C_2) + w(C_3) + \max\{\bar{w}(\bar{K}_1), \bar{w}(\bar{K}_3)\}$.

To show the algorithm produces a minimum weighted coloring, we turn to prove that G has a clique whose weight is equal to the right hand side of (*c*). Let C be a maximum weighted clique of G. From the structure of a peculiar graph, it is easy to see that

(d) If $C \cap K_i \neq \emptyset$ for some i, then C can be written as $C = K_i \cup A_{i+2} \cup B_{i+1} \cup D_{i+1}$, where D_{i+1} is a maximum weighted clique in the cobipartite graph (A_{i+1}, B_{i+2}), and that

(e) If $C \cap K_i = \emptyset$ for each i, then C can be written as $C = D_i \cup D_2 \cup D_3$, where D_j is a maximum weighted clique in the cobipartite graph (A_j, B_{j+1}) for $j = 1, 2, 3$.

By virtue of (d) and (e), we get

$$\begin{aligned} w(C) &= \max\{w(D_1) + w(D_2) + w(D_3), \max_{1 \leq i \leq 3}\{w(K_i) \\ &\quad + w(A_{i+2}) + w(B_{i+1}) + w(D_{i+1})\}\} \\ &= \max\{w(C_1) + w(C_2) + w(C_3), \max_{1 \leq i \leq 3}\{w(K_i) \\ &\quad + w(A_{i+2}) + w(B_{i+1}) + w(C_{i+1})\}\} \end{aligned}$$

$$
\begin{aligned}
&= \max\left\{ w(C_1) + w(C_2) + w(C_3),\ \max_{1\le i\le 3}\left\{ w(K_i) \right.\right. \\
&\quad \left.\left. + w(C_i) + w(C_{i+1}) + w(C_{i+2}) - \sum_{S\in S_i} x(S) \right\}\right\} \\
&= \max\left\{ w(C_1) + w(C_2) + w(C_3),\ w(C_1) + w(C_2) + w(C_3) \right. \\
&\quad \left. + \left\{ \max_{1\le i\le 3}\{ w(K_i) - \sum_{S\in S_i} x(S) \right\}\right\} \\
&= \text{RHS of (c)},
\end{aligned}
$$

where the second equality follows from the fact that C_i and D_i are both maximum weighted cliques of the cobipartite graph (A_i, B_{i+1}), the third one follows from (a), and the last one follows from (b). By the remark above Theorem 2.1, $\mathcal{T}$ together with $x(S)$ for each $S \in \mathcal{T}$ form a minimum weighted coloring of G and hence the correctness of the algorithm is established.

According to Lemma 2.5, Ai, B_i, K_i, for $i = 1, 2, 3$, in Step 1 can be constructed in $O(n^3)$ time using Algorithm (2.5). By Lemma 2.1, ϕ_1, ϕ_2, ϕ_3 in Step 2 can be obtained in $O(n^3)$ time using Algorithm (2.1) and the current size of $\mathcal{T}$ is $O(n^2)$. Clearly Step 3 takes $O(n)$ time to generate some new stable sets, but it never changes $|\mathcal{T}|$. And Step 5 takes $O(n)$ time to generate $O(n)$ stable sets. So the algorithm correctly finds a minimum weighted coloring of G with $O(n^2)$ stable sets in $O(n^3)$ time. □

Let H be the line graph of a bipartite multigraph, let $x_1y_1, x_2y_2, \ldots, x_ky_k$ be k pairwise nonincident flat edges of H, and let $H_1 = (X_1, Y_1; E_1)H_2 = (X_2, Y_2; E_2), \ldots, H_k = (X_k, Y_k; E_k)$ be k pairwise disjoint connected cobipartite graphs that are also disjoint from H. For each i with $1 \le i \le k$, suppose X_{i1}, X_{i2} is a partition of X_i, and Y_{i1}, Y_{i2} is a partition of Y_i (possibly some X_{ij} or Y_{ij} is empty). Let $H_i' = (X_i, Y_i; E_i')$ be the cobipartite graph such that $X_i, Y_i, X_{i1} \cup Y_{i1}$, and $X_{i2} \cup Y_{i2}$ are all cliques and that there is no other edge in H_i'. Denote by G the elementary graph obtained from H by augmenting each x_iy_i with H_i, and by G' the graph obtained from H by augmenting each x_iy_i with H_i', for $i = 1, 2, \ldots, k$. (Note that H_i' may not be connected when some X_{ij} or Y_{ij} is empty, yet in this case we still augment each x_iy_i with H_i' as described in Definition 1.1.)

Lemma 3.2. *The following statements hold for the above construction:*

(i) *G' is also the line graph of a bipartite multigraph;*
(ii) *Suppose each vertex v of G is associated with a nonnegative integral weight $w(v)$ and suppose $X_{i1} \cup Y_{i1}$ is a maximum weighted clique of H_i with respect to the weight function w, for $i = 1, 2, \ldots, k$. Then $w(G', w) = w(G, w)$.*

Proof: (i) Denote by B a bipartite multigraph such that $H = L(B)$. Without loss of generality, we may assume that B is connected. We prove the statement by induction on k.

For $k = 1$, let ab, ac be the edges of B corresponding to vertices x_1 and y_1 of H, respectively. Note that a is of degree two in B, for otherwise x_1y_1 would be contained in a triangle in $L(B)$, contradicting the hypothesis that x_1y_1 is a flat edge of H. If $b = c$ then, similarly, b is also of degree two. Thus $H = x_1y_1$ and hence $G' = H'_1$, which is obviously the line graph of a bipartite multigraph. So we assume $b \neq c$. Let B' be the graph obtained from B by first deleting vertex a, then adding two new vertices a_1, a_2, and finally adding $|X_{11}|$ parallel edges between a_1 and b, $|Y_{11}|$ parallel edges between a_1 and c, $|X_{12}|$ parallel edges between a_2 and b, and $|Y_{12}|$ parallel edges between a_2 and c. It is a routine matter to check that G', the graph obtained from H by augmenting x_1y_1 with H'_1, is nothing but $L(B')$. So the statement holds for the induction base.

Suppose the statement is valid for $k = t$. Let us proceed to the induction step for $k = t+1$. Let G'' be the graph obtained from H by augmenting each x_iy_i with H'_i, for $i = 1, 2, \ldots, t$.

Then the induction hypothesis guarantees that G'' is the line graph of a bipartite multigraph. Since G' can be obtained from G'' by augmenting $x_{t+1}y_{t+1}$ with H'_{1+1}, from the induction base it follows instantly that G' is the line graph of a bipartite multigraph, completing the proof of (i).

(ii) To prove $w(G', w) \geq w(G, w)$, let C be a maximum weighted clique of G with respect to w, that is, $w(C) = w(G, w)$. If there exists a subscript i with $1 \leq i \leq k$ such that $C \cap X_i \neq \emptyset$ and $C \cap Y_i \neq \emptyset$, then $C \subseteq X_i \cup Y_i$ since x_iy_i is a flat edge of H and H_i is an augment along x_iy_i. By hypothesis, $X_{i1} \cup Y_{i1}$ is a maximum weighted clique of H_i with respect to the weight function w, so $w(C) = w(X_{i1} \cup Y_{i1}) \leq w(G', w)$. On the other hand, if for each i we have $C \cap X_i = \emptyset$ or $C \cap Y_i = \emptyset$, then C is also a clique in G', and so $w(C) \leq w(G'w)$. Combining these two cases, we get the desired inequality.

To establish the reverse inequality $w(G, w) \geq w(G', w)$, observe that $X_{i1} \cup Y_{i1}$ is also a maximum weighted clique of H'_i with respect to w for each i. (Indeed, since X_i, Y_i and $X_{i1} \cup Y_{i1}$ are all cliques in H_i, comparing their total weights we obtain $w(X_{i2}) \leq w(Y_{i1})$ and $w(Y_{i2}) \leq w(X_{i1})$ for $X_{i1} \cup Y_{i1}$ is a maximum clique of H_i by hypothesis, so the desired statement follows.) The rest of the proof goes along the same line as in the preceding paragraph. □

Let G and G' be two elementary graphs (as defined above Lemma 3.2) that satisfy the hypothesis described in Lemma 3.2(ii). The key ingredient of the algorithm for a minimum weighted coloring of G is to color G' first, using Algorithm (2.4), and then transform the minimum weighted coloring ϕ of G' into the desired one of G. By Lemma 3.2(i) and Lemma 2.4, ϕ is formed by a collection $\mathcal{T}$ of *O(n)* stable sets and nonnegative integers $x(S)$ for all $S \in \mathcal{T}$, where $n|V(G)| = |V(G')|$.

(3.2) Algorithm for transforming a minimum weighted coloring ϕ of G' into one of G.

Step 1. For $i = 1, 2, \ldots, k$, construct a graph F_i by adding to H_i two adjacent vertices a_i and b_i such that a_i is adjacent to all vertices in X_i and b_i is adjacent to all vertices in Y_i. Set $j = 1$.

Step 2. Let $\mathcal{T}_{j1}$ (resp. $\mathcal{T}_{j2}$) be the collection of all $S \in \mathcal{T}$ such that $S \cap X_i \neq \emptyset$ and $S \cap Y_j = \emptyset$ (resp. $S \cap X_j = \emptyset$ and $S \cap Y_j \neq \emptyset$), and let $\mathcal{T}_{j3}$ be the collection of all $S \in \mathcal{T}$ such that $S \cap X_j \neq \emptyset S \cap Y_j$. Set $w(a_j) \sum_{S \in \mathcal{T}j2} x(S)$ and $W(b_j) = \sum_{S \in \mathcal{T}j1} x(S)$. Find a minimum

weighted coloring ϕ_j of F_j with respect to w by using Algorithm (2.1). Let $\mathcal{R}_{j1}$ (resp. $\mathcal{R}_{j2}$) denote all the stable sets S in ϕ_j such that $b_j \in S$ but $a_j \notin S$ (resp. $a_j \in S$ but $b_j \notin S$), and let $\mathcal{R}_{j3}$ denote all the stable sets in ϕ_j such that $S \cap \{a_j, b_j\} = \emptyset$. Set $\ell = 1$.

Step 3. If both $\mathcal{T}_{j\ell}$ and $\mathcal{R}_{j\ell}$ are nonempty, take $\mathcal{T} \in \mathcal{T}_{j\ell}$ and $\mathcal{R} \in \mathcal{R}_{j\ell}$. Set $S = (T \backslash (X_j \cup Y_j)) \cup (R \backslash \{a_j, b_j\})$ and $x(S) = \min\{x(T), x(R)\}$. Set $x(T) = x(T) - x(S)$, $x(R) = x(R) - x(S)$, $\mathcal{T}_{j\ell} = \mathcal{T}_{j\ell} \backslash T$ if $x(T) = 0$, and $\mathcal{R}_{j\ell} = \mathcal{R}_{j\ell} \backslash R$ if $x(R) = 0$. Set $\mathcal{T} = \mathcal{T} \cup \{S\}$ if $x(T) \neq 0$ and $\mathcal{T} = \mathcal{T} \cup \{S\} \backslash T$ if $x(T) = 0$. Repeat the process until one of $\mathcal{T}_{j\ell}$ and $\mathcal{R}_{j\ell}$ is empty.

Step 4. Set $\ell = \ell + 1$. If $\ell \leq 3$, return to Step 3. Else, set $j = j + 1$. If $j \leq k$, go back to Step 2. Else, stop: $\mathcal{T}$ together with $x(S)$ for each $S \in \mathcal{T}$ form a minimum weighted coloring of G.

Lemma 3.3. *Algorithm* (3.2) *correctly transforms a minimum weighted coloring of G' into one of G with $O(n^2)$ stable sets in $O(n^4)$ time, where $n = |V|$.*

Proof: Since ϕ is a minimum weighted coloring of G' with respect to w and G' is perfect, at the beginning of the algorithm we have

(a) $\sum_{S \in \mathcal{T}} x(S) = w(G', w) = w(G, w)$ by Lemma 3.2.

The construction of H'_i implies that in Step 2

(b) $\sum_{S \in \mathcal{T}_{j1} \cup \mathcal{T}_{j2} \cup \mathcal{T}_{j3}} x(S) \geq w(H'_j, w) = w(H_j, w)$ for $j = 1, 2, \ldots, k$.

(c) $w(F_j, w) = \sum_{S \in \mathcal{T}_{j1} \cup \mathcal{T}_{j2} \cup \mathcal{T}_{j3}} x(S)$.

To justify (c), let C be an arbitrary maximal clique (with respect to set-inclusion) of F_j. If $a_j \in C$ but $b_j \notin C$, then it follows from the construction of F_j that $C = X_j \cup \{a_j\}$. Thus by Definition 2.1 we have $w(C) = w(X_j) + w(a_j) = \sum_{S \in \mathcal{T}_{j1} \cup \mathcal{T}_{j3}} x(S) + \sum_{S \in \mathcal{T} j2} x(S) = \sum_{S \in \mathcal{T}_{j1} \cup \mathcal{T}_{j2} \cup \mathcal{T}_{j3}} x(S)$. Similarly, we can prove the same statement when $b_j \in C$ but $a_j \notin C$. If $\{a_j, b_j\} \subseteq C$ then, by the construction of F_j, we have $C = \{a_j, b_j\}$. Thus $w(C) = w(a_j) + w(b_j) = \sum_{S \in \mathcal{T}_{j2}} x(S) + \sum_{S \in \mathcal{T}_{j_1}} x(S) \leq \sum_{S \in \mathcal{T} j_1 \cup \mathcal{T} j_2 \cup \mathcal{T} j_3} x(S)$. if $C \cap \{a_j, b_j\} = \emptyset$, then C is a clique of H_j and so, by (b), $w(C) \leq w(H_j, w) \leq \sum_{S \in \mathcal{T}_{j1} \cup \mathcal{T}_{j2} \cup \mathcal{T}_{j3}} x(S)$. Hence (c) holds.

Since F_j is a cobipartite graph, it is perfect. Hence $w(F_j, w) = \sum_{S \in \mathcal{R}_{j1} \cup \mathcal{R}_{j2} \cup \mathcal{R}_{j3}} x(S)$. By (c), Definition 2.1, and the assignment of $w(a_j)$ and $w(b_j)$, we obtain

(d) $\sum_{S \in \mathcal{T}_{j\ell}} x(S) = \sum_{S \in R_{jl}} x(S)$ for $l = 1, 2, 3$.

The purpose of Step 3 is to combine the stable sets in $\mathcal{T}_{j\ell}$ and those in $\mathcal{R}_{j\ell}$. It follows from (d) that at the end of Step 3, $\mathcal{T}_{j\ell} = \mathcal{R}_{j\ell} = \emptyset$ for $\ell = 1, 2, 3$.

From Step 3 we see that $\sum_{S \in \mathcal{T}} x(S)$ remains the same throughout the algorithm. Hence, by virtue of (a), $\mathcal{T}$ together with $x(S)$ for each $S \in \mathcal{T}$ output by the algorithm form a minimum weighted coloring of G. Therefore the correctness of the algorithm is established.

(e) At the end of the algorithm, we have $|\mathcal{T}| = O(n^2)$.

To justify (e), recall that at the beginning of the algorithm $|\mathcal{T}| = O(n)$. Since F_j is a cobipartite graph, by Lemma 2.1 there are $O(|V(F_j)|^2) = O(|V(H_j)|^2)$ stable sets involved in ϕ_j in Step 2. Thus it can be deduced from Step 3 that there are $O(n) + \sum_{j=1}^{k} O(|V(H_j)|^2) = O(n^2)$ stable sets in $\mathcal{T}$ at the end of the algorithm. So (e) follows.

Clearly the complexity of the algorithm is dominated by that of Steps 2 and 3. To figure out $\mathcal{T}_{jl}$ for $\ell = 1, 2, 3$, we can simply check the vertices contained in the stable sets in $\mathcal{T}$ by the enumeration method, which by (e) takes $O(n^3)$ time. Recalling Lemma 2.1, it takes $O(|V(F_j)|^3) = O(|V(H_j)|^3)$ time to get ϕ_j in Step 2. Hence in the whole algorithm Step 2 takes $O(n^4)$ time. Since it takes $O(n)$ time to get the S in Step 3, we need $O(n(|\mathcal{T}_{j\ell}|+|\mathcal{R}_{j\ell}|))$ time to combine $\mathcal{T}_{j\ell}$ and $\mathcal{R}_{j\ell}$ for each ℓ. So in the whole algorithm Step 3 also takes $O(n^4)$ time. Therefore the whole algorithm runs in $O(n^4)$ time. □

(3.3) Algorithm for finding a minimum weighted coloring of an elementary graph $G = (V, E)$ such that each vertex v of G is associated with a nonnegative integral weight $w(v)$.

Step 1. Apply the Maffray-Reed algorithm (Maffray and Reed, 1999) to find a line graph H of a bipartite multigraph, pairwise non-incident flat edges of H denoted by $x_1y_1, x_2y_2, \ldots, x_ky_k$ for some k, and pairwise disjoint connected cobipartite graphs $H_1 = (X_1, Y_1; E_1)$, $H_2 = (X_2, Y_2; E_2), \ldots, H_k = (X_k, Y_k; E_k)$ (which are also disjoint from H) such that G can be obtained from H by augmenting each x_iy_i with H_i, for $i = 1, 2, \ldots, k$.

Step 2. For $i = 1, 2, \cdots, k$, find a maximum weighted clique C_i in H_i with respect to w using Algorithm (2.1). Set $X_{i1} = X_i \cap C$, $X_{i2} = X_i \backslash C$, $Y_{i1} = Y_i \cap C$, and $Y_{i2} = Y_i \backslash C$. Let $H_i' = (X_i, Y_i; E_i')$ be the cobipartite graph such that X_i, Y_i, $X_{i1} \cup Y_{i1}$, and $X_{i2} \cup Y_{i2}$ are all cliques and that there is no other edge in H_i', and let G' the graph obtained from H by augmenting each x_iy_i with H_i', for $i = 1, 2, \ldots, k$.

Step 3. Find a minimum weighted coloring ϕ of G' using Algorithm (2.4), and then transform ϕ into a minimum weighted coloring φ of G using Algorithm (3.2). Return φ, stop.

Lemma 3.4. *Algorithm* (3.3) *correctly finds a minimum weighted coloring φ of $G = (V, E)$ with $O(n^2)$ stable sets in $O(n^4)$ time, where $n = |V|$.*

Proof: The correctness of the algorithm follows directly from Lemma 3.3. As analyzed in Maffray and Reed (1999), the running time of the Maffray-Reed algorithm employed in Step 1 is $O(m^2)$, where $m = |E|$. Since, by Lemma 2.1, it takes $O(|V(H_i)|^3)$ time to find C_i, Step 2 takes $\sum_{i=1}^{k} O(|V(H_i)^3) = O(n^3)$ time. In view of Lemma 2.4, a minimum weighted coloring ϕ of G' in Step 3 can be found in $O(n^3)$ time, and by Lemma 3.3, ϕ can be transformed into a minimum weighted coloring φ of G with $O(n^2)$ stable sets in $O(n^4)$ time. So Algorithm (3.3) runs in $O(n^4)$ time. □

Let $G = (V, E)$ be a perfect graph such that each vertex v of G is associated with a non-negative integral weight $w(v)$. Suppose C is a clique-cutset of G and A is a component of $G\backslash C$. Set $G_1 = G[A \cup C]$ and $G_2 = G[V\backslash A]$. Suppose ϕ_i is a minimum weighted coloring of G_i which is formed by a collection S_i of stable sets in G_i and nonnegative integers $x(S)$ for $S \in S_i$. Let us now produce a minimum weighted coloring of G.

(3.4) Algorithm for gluing ϕ_1 and ϕ_2 to form a minimum weighted coloring of G.

Step 1. Set $\mathcal{T} = S_1 \cup S_2$. Denote $C = \{1, 2, \cdots, k\}$. For $i = 1, 2$ and each $j \in C$, define $\mathcal{S}_{ij}$ to be the collection of all stable sets in $\mathcal{S}_i$ containing j. Set $j = 1$.

Step 2. If $j \geq k+1$, go to Step 4. Else, if both S_{1j} and S_{2j} are nonempty, take $T_i \in S_{ij}$ for $i = 1, 2$. Set $T = T_1 \cup T_2$ and $x(T) = \min\{x(T_1), x(T_2)\}$, For $i = 1, 2$, set $x(T_i) = x(T_i) - x(T)$, and set $S_{ij} = S_{ij} \backslash \{T_i\}$ if $x(T_i) = 0$, and set $\mathcal{S}_i = \mathcal{S}_i \backslash \{T_i\}$ if $x(T_i) = 0$. Set $\mathcal{T}$ to be the collection of stable sets obtained from $\mathcal{T} \cup \{T\}$ by deleting each $T_i \in \{T_1, T_2\}$ with $x(T_i) = 0$. Repeat Step 2 until one of S_{1j} and S_{2j} is empty.

Step 3. Set $j = j + 1$, return to Step 2.

Step 4. If both $\mathcal{S}_1$ and $\mathcal{S}_2$ are nonempty, take $T_i \in \mathcal{S}_i$ for $i = 1, 2$. Set $T = T_1 \cup T_2$ and $x(T) = \min\{x(T_1), x(T_2)\}$. For $i = 1, 2$, set $x(T_i) = x(T_i) - x(T)$ and set $\mathcal{S}_i = \mathcal{S}_i \backslash \{T_i\}$ if $x(T_i) = 0$. Set $\mathcal{T}$ to be the collection of stable sets obtained from $\mathcal{T} \cup \{T\}$ by deleting each $T_i \in \{T_1, T_2\}$ with $x(T_i) = 0$. Repeat the process until one of $\mathcal{S}_1$ and $\mathcal{S}_2$ is empty, stop: $\mathcal{T}$ together with $x(T)$ for each $T \in \mathcal{T}$ form a minimum weighted coloring of G.

Lemma 3.5. *Algorithm* (3.4) *correctly finds a minimum weighted coloring of* $G = (V, E)$ *with at most* $|S_1| + |S_2|$ *stable sets in* $O(n(|\mathcal{S}_1| + |\mathcal{S}_2|))$ *time, where* $n = |V|$.

Proof: Initially, $|\mathcal{T}| = |\mathcal{S}_1| + |\mathcal{S}_2|$. In both Step 3 and Step 4, at least one of $x(T_1)$ and $x(T_2)$ becomes zero after each iteration. So $|\mathcal{T}|$ has never increased throughout the algorithm. Therefore $\mathcal{T}$ consists of at most $|\mathcal{S}_1| + |\mathcal{S}_2|$ stable sets at the end of the algorithm.

Since ϕ_i is a minimum weighted coloring of G_i with respect to w, at the beginning of the algorithm we have $\sum_{v \in T \in \mathcal{S}_i} x(T) = w(v)$ by Definition 2.1 for each $v \in V(G_i)$ and $i = 1, 2$. Hence for each j in C and $\mathcal{S}_{ij}$ defined in Step 1, we have $\sum_{T \in \mathcal{S}_{ij}} x(T) = \sum_{T \in \mathcal{S}_{2j}} x(T) = w(j)$. It follows from the algorithm that $\mathcal{S}_{ij} = \emptyset$ at the end of Step 3 for each $i = 1, 2$ and each $j = 1, 2, \ldots, k$. Let $\bar{\mathcal{T}}$, $\bar{\mathcal{S}}_i$, and $\bar{x}(T)$ stand for $\mathcal{T}$, $\mathcal{S}_i$, and $x(T)$ at the end of Step 3, respectively. Then $\sum_{T \in \bar{\mathcal{T}}} \bar{x}(T) = \sum_{T \in \mathcal{S}_1} x(T) + \sum_{T \in \mathcal{S}_2} x(T) - \sum_{v \in C} w(v)$ and $\sum_{T \in \bar{\mathcal{S}}_i} \bar{x}(T) = \sum_{T \in \mathcal{S}_i} x(T) - \sum_{v \in C} w(v)$. The sum $\sum_{T \in \mathcal{T}} x(T)$ at the end of Step 4 is clearly equal to $\sum_{T \in \bar{\mathcal{T}}} \bar{x}(T) - \min\{\sum_{T \in \bar{\mathcal{S}}_1} \bar{x}(T), \sum_{T \in \bar{\mathcal{S}}_2} \bar{x}(T)\} = \max\{\sum_{T \in \mathcal{S}_1} x(T), \sum_{T \in \mathcal{S}_1} x(T)\}$, which implies that $\mathcal{T}$ together with $x(T)$ for each $T \in \mathcal{T}$ form a minimum weighted coloring of G as $\sum_{T \in \mathcal{S}_i} x(T)$ is the total weight of a minimum weighted coloring of G_i for $i = 1, 2$.

By enumerating all the vertices in each $T \in \mathcal{S}_i$, we can get $\mathcal{S}_{ij}$ for each $i = 1, 2$ and $j = 1, 2, \ldots k$. So Step 1 takes $O(n(|\mathcal{S}_1| + |\mathcal{S}_2|))$ time. Since we can eliminate at least one T_i in $\mathcal{S}_1 \cup \mathcal{S}_2$ in each iteration of Step 2 and Step 4 and since it takes $O(n)$ time to get $T = T_1 \cup T_2$, the total running time of Steps 2–4 is $O(n(|\mathcal{S}_1| + |\mathcal{S}_2|))$. Hence the algorithm runs in $O(n(|\mathcal{S}_1| + |\mathcal{S}_2|))$ time. □

The graph decomposition technique plays an important role in the design of our main algorithm. Let $G = (V, E)$ be an arbitrary graph. Suppose G has a clique-cutset C and A is the vertex-set of one component of $G \backslash C$. Then we can decompose G into $G_1 = G[A \cup C]$ and $G_2 = G[V \backslash A]$. By decomposing G_1 and G_2 in the same way and repeating until no further decomposition is possible, we decompose G into a collection of indecomposable induced subgraphs of G, which are called *atoms* by Tarjan (1985). The atoms fit together in a hierarchy to form G. We can represent this hierarchy by a binary tree: each external vertex represents an atom and each internal vertex represents a clique-cutset. We call such a tree a *binary decomposition tree*. Notice that each atom corresponds to a leaf (i.e. degree-one vertex) of this tree. The following theorem was obtained by Tarjan.

Theorem 3.1 (Tarjan, 1985)**.** *For any graph $G = (V, E)$, a binary decomposition tree of G has at most $n - 1$ atoms and can be found in $O(mn)$ time, where $n = |V|$ and $m = |E|$.*

Now we are ready to present the algorithm for a minimum weighted coloring of an arbitrary claw-free perfect graph $G = (V, E)$. Recall Theorem 1.1, an application of Tarjan's algorithm (Tarjan, 1985) to G will lead to atoms which are elementary graphs or peculiar graphs.

(3.5) Algorithm for coloring a claw-free perfect graph $G = (V, E)$.

Step 1. Apply Tarjan's algorithm (Tarjan, 1985) to find a binary decomposition tree T of G. Let r be the root of T, let p be the maximum possible distance from r to a vertex in T, and let $G_1, G_2, \ldots, G_k$ be all the atoms of T, where $k \leq n - 1$.

Step 2. For $i = 1, 2, \ldots, k$, test if G_i is a peculiar graph using Algorithm (2.5). If yes, apply Algorithm (3.1) to output a minimum weighted coloring ϕ_i of G_i. Else (G_i is an elementary graph), output a minimum weighted coloring ϕ_i of G_i using Algorithm (3.3). Set $\mathcal{G}_p = \{G_1, G_2, \ldots, G_k\}$ and $\mathcal{F}_p = \{\phi_1, \phi_2, \ldots, \phi_k\}$. Set $\ell = p - 1$.

Step 3. If $\ell = -1$, stop: ϕ in $\mathcal{F}_0$ is a minimum weighted coloring of G. Else, let C be a clique-cutset of G such that C is a vertex of T and that the distance between r and C on T is precisely ℓ. Let G_i and G_j be the elements of $\mathcal{G}_{\ell+1}$ that contain C, and let H denote the subgraph of G induced by all the vertices in $V(G_i) \cup V(G_j)$. Produce a minimum weighted coloring ϕ of H by gluing ϕ_i and ϕ_j in $\mathcal{F}_{\ell+1}$ using Algorithm (3.4). Let $\mathcal{G}_\ell$ denote the collection obtained from $\mathcal{G}_{\ell+1}$ by adding all these possible H and then deleting all the corresponding G_i and G_j, and let $\mathcal{F}_\ell$ denote the collection obtained from $\mathcal{F}_{\ell+1}$ by adding all these possible ϕ and then deleting all the corresponding ϕ_i and ϕ_j.

Step 4. Set $\ell = \ell - 1$, go to Step 3.

Theorem 3.2. *Algorithm* (3.5) *correctly finds a minimum weighted coloring of a claw-free perfect graph G with $O(n^3)$ stable sets in $O(n^5)$ time, where $n = |V|$.*

Proof: The correctness of the algorithm follows instantly from Lemma 3.1, Lemma 3.4 and Lemma 3.5. Let α stand for the number of all the degree-one vertices of T and let β stand for the number of all other vertices (each of them has degree three except for the root, which has degree two). Since T has $\alpha + \beta - 1$ edges, by the handshaking theorem $\alpha + 3(\beta - 1) + 2 = 2(\alpha + \beta - 1)$, so $\beta = \alpha - 1$. Since α equals the number of all the atoms in the graph decomposition, by Theorem 3.1, $\alpha \leq n - 1$ and so $\beta \leq n - 2$. It follows that there are at most $n - 2$ clique-cutsets involved in the graph decomposition, and hence in Algorithm (3.5), we only need to apply Algorithm 3.4 at most $n - 2$ times. Since each ϕ_i produced in Step 2 consists of $O(n^2)$ stable sets (recall Lemmas 3.1 and 3.4), repeated applications of Lemma 3.5 imply that the output of the algorithm contains $O(n^3)$ stable sets.

To estimate the running time of the algorithm, note that Step 1 takes $O(mn) = O(n^3)$ time to decompose G using Tarjan's algorithm by Theorem 3.1. Since it takes $O(n^4)$ time to produce a ϕ_i with $O(n^2)$ stable sets for each G_i (recall Lemmas 3.1 and 3.4), Step 2 takes $O(n^5)$ time. Let $\mathcal{S}_i$ stand for the collection of all stable sets produced in ϕ_i (we

have $|\mathcal{S}_i| = O(n^2)$), and let p_i be the distance between the root r of T and atom G_i. Using Lemma 3.5 and induction on the number of vertices in the binary decomposition tree T, it is easy to see that Step 3 and Step 4 take $O(n\sum_{i=1}^{k} p_i|\mathcal{S}_i|)$ time. Since $p_i \leq \beta \leq n-2$, $O(n\sum_{i=1}^{k} p_i|\mathcal{S}_i|) = O(n^5)$. So the whole algorithm takes $O(n^5)$ time. This completes the proof. □

Acknowledgment

The authors are grateful to an anonymous referee for his/her invaluable comments and suggestions.

References

C. Berge and V. Chvátal (Eds.), *Topics on Perfect Graphs*. North-Holland, Amsterdam, 1984.

M. Chudnovsky, N. Robertson, P.D. Seymour, and R. Thomas, "The strong perfect graph theorem," *Ann. of Math.*, to appear.

V. Chvátal, *Linear Programming*. W.H. Freeman and Company: New York, 1983.

V. Chvátal and N. Sbihi, Bull-free Berge graphs are perfect. *Graphs Combin*, vol. 3, pp. 127–139, 1987.

V. Chvátal and N. Sbihi, "Recognizing claw-free perfect graphs," *J. Combin. Theory Ser. B*, vol. 44, pp. 154–176, 1988.

D.G. Degiorgi and K. Simon, "A dynamic algorithm for line graph recognition," *Lecture Notes in Comput. Sci.*, vol. 1017, pp. 37–48, 1995.

L.R. Ford, Jr. and D.R. Fulkerson, "Maximal flow through a network," *Canad. J. Math.*, vol. 8, pp. 399–404, 1956.

M.R. Garey and D.S. Johnson, *Computers and Intractability*. W.H. Freeman and Company, New York, 1979.

F. Gavril, "Algorithms on clique separable graphs," *Discrete Math.*, vol. 55, pp. 245–254, 1985.

T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," *J. Assoc. Comput. Mach.*, vol. 23, pp. 665–679, 1976.

M. Grötschel, L. Lovász, and A. Schrijver, "Polynomial algorithms for perfect graphs," in *Topics on Perfect Graphs* C. Berge and V. Chvátal (Eds.) North-Holland, Amsterdam, 1984, pp. 325–356

J.E. Hopcroft and R.M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput.*, vol. 2, pp. 225–231, 1973.

W.L. Hsu, "How to color claw-free perfect graphs," *Ann. Discrete Math.*, vol. 11, pp. 189–197, 1981.

W.L. Hsu and G. Nemhauser, "Algorithms for maximum weight cliques, minimum weighted clique covers and minimum colorings of claw-free perfect graphs," in *Topics on Perfect Graphs* C. Berge and V. Chvátal (Eds.) North-Holland, Amsterdam, 1984, pp. 357–369.

W.L. Hsu and G. Nemhauser, "Algorithms for minimum covering by cliques and maximum clique in claw-free perfect graphs," *Discrete Math.*, vol. 37, pp. 181–191, 1981.

W.L. Hsu and G. Nemhauser, "A polynomial algorithm for the minimum weighted clique cover problem on claw-free perfect graphs," *Discrete Math.*, vol. 38, pp. 65–71, 1982.

A.V. Karzanov, "Determining the maximal flow in a network by the method of preflows," *Soviet Math. Dokl.*, vol. 15, pp. 434–437, 1974.

P.G.H. Lehot, "An optimal algorithm to detect a line graph and output its root graph," *J. Assoc. Comput. Mach.*, vol. 21, pp. 569–575, 1974.

L. Lovász, Normal hypergraphs and the perfect graph conjecture," *Discrete Math.*, vol. 2, pp. 253–267, 1972.

F. Maffray and B. Reed, A description of claw-free perfect graphs," *J. Combin. Theory Ser. B*, vol. 75, pp. 134–156, 1999.

K.R. Parthasarathy and G. Ravindra, "The strong perfect graph conjecture is true for $K_{1,3}$-free graphs," *J. Combin. Theory Ser. B*, vol. 21, pp. 212–223, 1976.

J.L. Ramfrez-Alfonsín and B.A. Reed (Eds.), *Perfect Graphs*. John Wiley & Sons, Ltd., Chichester, 2001.

N.D. Roussopoulos, "A max{m, n} algorithm for determining the graph H from its line graph G," *Inform. Process. Lett*., vol. 2, pp. 108–112, 1973.

A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

R.E. Tarjan, "Decomposition by clique separators," *Discrete Math*., vol. 55, pp. 221–232, 1985.

R.E. Tarjan, *Data Structures and Network Algorithms*, SIAM, Philadelphia, 1983.

S.H. Whitesides, "An algorithm for finding clique cut-sets," *Inform. Process. Lett*., vol. 12, pp. 31–32, 1981.