ALGORITHMS FOR WEAKLY TRIANGULATED GRAPHS

Arvind Raghunathan †

573 Evans Hall
Computer Science Division
UC Berkeley, CA 94720

ABSTRACT

A graph $G = (V, E)$ is said to be weakly triangulated if neither G nor G^c, the complement of G, contain chordless or induced cycles of length greater than four. Ryan Hayward showed that weakly triangulated graphs are perfect. Later, Hayward, Hoang and Maffray obtained $O(e \cdot v^3)$ algorithms to find a maximum clique and a minimum coloring of a weakly triangulated graph. Performing these algorithms on the complement graph gives $O(v^5)$ algorithms to find a maximum independent set and a minimum clique cover of such a graph.

It was shown in [13-16] that weakly triangulated graphs play a crucial role in polygon decomposition problems. Several polygon decomposition problems can be formulated as the problem of covering a weakly triangulated graph with a minimum number of cliques. Motivated by this, we now improve on the algorithms of Hayward, Hoang and Maffray by providing $O(e \cdot v^2)$ algorithms to find a maximum clique and a minimum coloring of a weakly triangulated graph. We thus obtain an $O(v^4)$ algorithm to find a maximum independent set and a minimum clique cover of such a graph. We also provide $O(v^5)$ algorithms for weighted versions of these problems.

April 18, 1989

† Supported by the Semiconductor Research Corporation under grant SRC-82-11-008.

ALGORITHMS FOR WEAKLY TRIANGULATED GRAPHS

Arvind Raghunathan †

573 Evans Hall
Computer Science Division
UC Berkeley, CA 94720

1. Introduction

Polygon decomposition refers to the breaking up of a polygon into a minimum number of simpler polygons. The simpler polygons most commonly used are convex and star polygons. A decomposition is called a covering if the simpler polygons are allowed to overlap. An orthogonal polygon, P, is a polygon with each of its boundary edges parallel to one of the two Cartesian coordinate axes. Let n denote the number of boundary edges of P. Two problems of interest in computational geometry are to find a minimum cover of P by orthogonally convex polygons (OCP's) and by star polygons [2, 3, 13-15, 17]. The problem of finding a minimum OCP cover of P can be formulated as the problem of finding a minimum clique cover of an associated visibility graph, called the source graph [13, 15, 17]. Similarly, the problem of finding a minimum star cover of P can likewise be formulated as the problem of finding a minimum clique cover of an associated visibility graph, called the star graph [14, 15]. For an important special class of orthogonal polygons called class 3 polygons, the source graph is a weakly traingulated graph on $O(n)$ vertices [13, 15]. For the most general class of orthogonal polygons, the star graph is a weakly traingulated graph on $O(n^2)$ vertices [14]. Therefore, the problem of finding a minimum clique cover of a weakly triangulated graph is of crucial importance in polygon decomposition.

In this paper, we obtain fast algorithms for the problems of finding a maximum clique, a minimum vertex coloring, a maximum independent set and a minimum clique cover of weakly triangulated graphs. Specifically, we present an algorithm which runs in time $O(e \cdot v^2)$ to find a maximum clique and a minimum coloring in weakly triangulated graphs. We also present an $O(e \cdot v^3)$ time algorithm which solves the weighted versions of these problems. Performing these algorithms on the complement graph give solutions to the unweighted and weighted versions, respectively, of the maximum independent set and minimum clique cover problems. By the above results, a minimum OCP cover of a class 3 polygon can be found in $O(n^4)$ time, and a minimum star cover of an orthogonal polygon can be found in $O(n^8)$ time.

A simple undirected graph $G = (V, E)$ is said to be *perfect* [1, 6, 11] if the size of a maximum independent set of every induced subgraph H of G is equal to the size of a minimum clique cover of H. Ryan Hayward [9] defined weakly triangulated graphs and showed that they are a subclass of perfect graphs. Weakly triangulated graphs will be referred to in the rest of this paper as W.T. graphs for short.

Before going any further, let us put the above results in perspective. All the above listed problems are NP-hard for general graphs [4]. However, for the important special class of perfect graphs, each of the above problems can be solved in polynomial time [7]. W.T. graphs are a subclass of perfect graphs [9]; it is this perfectness property that permits polynomially bounded solutions for each of the above problems.

Historically, triangulated graphs (also known as chordal graphs) were the first class of perfect graphs to be studied [1,8]. Triangulated graphs are defined to be graphs that do not contain chordless cycles of length greater than three. It is easy to see that the chordless cycle with five vertices is isomorphic to its complement, and that the complement of every chordless cycle with at least six vertices contains a chordless cycle with four vertices. Hence triangulated graphs are W.T., and the complements of triangulated graphs are also W.T. F. Gavril [5] has provided linear time algorithms to find a maximum clique, a minimum vertex coloring, a maximum independent set and a minimum clique cover of triangulated graphs.

For a perfect graph G, a maximum clique, a minimum coloring, a maximum independent set and a minimum clique cover can be found in polynomial time by an algorithm due to Grötschel, Lov´asz and Schrijver [7]. This algorithm of Grötschel, Lov´asz and Schrijver is based on the Ellipsoid method of linear programming. Since W.T. graphs are known to be perfect, the algorithm of Grötschel, Lov´asz and Schrijver provides polynomial time solutions for the above mentioned problems on W.T. graphs. However, the Ellipsoid method of linear programming is well-known to have severe problems with rounding in computer implementations [18]. Therefore, their algorithm cannot be considered practical by any means. Motivated by this, Hayward, Hoang and Maffray [10] devised $O(e \cdot v^3)$ algorithms for the problems of finding a maximum clique and a minimum coloring in W.T. graphs. They also provided $O(e \cdot v^4)$ algorithms for the weighted versions of these problems. These algorithms are purely combinatorial in nature and therefore do not suffer from the same drawbacks as the Ellipsoid method. In this paper, we retain the basic algorithms of Hayward, Hoang and Maffray while speeding up a crucial step to obtain $O(e \cdot v^2)$ and $O(e \cdot v^3)$ algorithms for the unweighted and weighted cases respectively.

2. W.T. graphs and Two-Pairs

An *even pair* is a pair of (non-adjacent) vertices in a graph G, such that any chordless path that joins the two vertices in G has an even number of edges. Meyniel [12] defined a graph G to be *strict quasi-parity* if every induced subgraph H of G is either a clique or has an even pair. A graph is said to be *quasi-parity* if for every induced subgraph H of G, at least one of H or H^c, the complement of H, is either a clique or has an even pair. Quasi-parity and strict quasi-parity graphs are known to be perfect [12].

A *two-pair* is a pair of (non-adjacent) vertices in a graph G, such that any chordless path between them has exactly two edges. One important theorem of Hayward, Hoang and Maffray [10] states that every W.T. graph that is not a clique contains a two-pair. Their proof is existential in the sense that it does not afford an easy implementation to find such a two-pair in polynomial time. In this paper, we prove a stronger theorem that helps us obtain the desired speedup to

the algorithms of Hayward, Hoang and Maffray.

Theorem 1. Every W.T. graph that is not a clique satisfies the following two properties.

(1) If G has a minimal cutset C that is not a clique, then C contains a two-pair of G,

(2) G contains a two-pair.

Proof: Our proof works by induction on the number of vertices. The base case of (1) is easily established by noting that the smallest graphs that are W.T. and have a non-clique cutset are P_4, the path on four vertices, and C_4, the chordless cycle on four vertices. The base case of (2) is also established by considering the trivial graph on two vertices. By the induction hypothesis, we now have that every W.T. graph with fewer vertices than G satisfies properties (1) and (2).

Let $C \subseteq V$ be a *minimal* cutset of G. Let H_C denote the subgraph induced by C. We have two cases.

CASE I. H_C is a clique. We need to show that G contains a two-pair. Let $B_1, \cdots, B_t$ be the vertex sets of the components of $G - C$. If some $G - B_j$ is not a clique, then by the induction hypothesis $G - B_j$ contains a two-pair. This is a two-pair of G, since every chordless path in G with both endpoints in $G - B_j$ is fully contained in $G - B_j$. Hence, we may assume that each $G - B_j$ is a clique. Thus, $t = 2$, and we have that $\{x, y\}$ is a two-pair whenever $x \in B_1$ and $y \in B_2$.

CASE II. H_C is not a clique. We need to show that C contains a two-pair of G. Here again, we distinguish between two cases. Let ${H_C}^c$ denote the complement of H_C.

Case IIa. Let ${H_C}^c$ be disconnected. Let D be the set of vertices of some component of ${H_C}^c$ with at least two vertices (since C is not a clique, there must be such a set D). Since D is a connected component of ${H_C}^c$, every vertex of $C - D$ is adjacent to every vertex of D. For the same reason, it is clear that D is not a clique. We further assert that D is a minimal cutset of $G - (C - D)$. If not, $D' \subset D$ is a cutset of $G - (C - D)$. We then have that $(C - D) \cup D'$ is a cutset of G, contradicting the minimality of C. By inductive assumption, D contains a two-pair, $\{x, y\}$, of $G - (C - D)$. Since every vertex of D is adjacent to every vertex of $C - D$, $\{x, y\}$ is a two-pair of G.

Case IIb. Let ${H_C}^c$ be connected. Let $B_1, \cdots, B_t$ be the vertex sets of the connected components of $G - C$. Now, it follows from Theorem 1 of [9] that in each of $B_1, \cdots, B_t$, there is some vertex that is adjacent to all of C in G. Let $z(B_j)$ denote one such vertex for each j.

Case IIb.1. First, let $|B_j| = 1$ for all j. Thus, each B_j consists of exactly one vertex, namely $z(B_j)$. Since H_C is a W.T. graph that is not a clique, by inductive assumption H_C contains some two-pair $\{x, y\}$. $\{x, y\}$ is clearly a two-pair of G, as both x and y are adjacent to $z(B_j)$, for all j.

Case IIb.2. Now, let $|B_p| \geq 2$ for some $p \in \{1, \cdots, t\}$. We now confine our attention on this specific B_p. Let us denote by $A(C, B_p)$ the set of vertices of C that are adjacent to some vertex of $B_p - z(B_p)$.

First, suppose that G_A, the subgraph induced by $A\ (C\ , B_p\ )$ is not a clique. It is easy to see that $A\ (C\ , B_p\ )$ is not empty and is a minimal cutset of $G - z\ (B_p\ )$. By inductive assumption, $A\ (C\ , B_p\ )$ contains a two-pair $\{x\ , y\ \}$ of $G - z\ (B_p\ )$. Since $z\ (B_p\ )$ is adjacent to both x and y, we have that $\{x\ , y\ \}$ is a two-pair of G.

Next, suppose that G_A is a clique. Consider the subgraph $G' = G - (B_p - z\ (B_p\ ))$. Since C is a minimal cutset, not a clique, of G', C contains a two-pair $\{x\ , y\ \}$ of G' by the induction hypothesis. We now assert that $\{x\ , y\ \}$ is a two-pair of G. If $\{x\ , y\ \}$ is not a two-pair of G, then there exists a chordless path P in G connecting x and y with at least three edges. Clearly, P has to include vertices of B_p. P cannot use $z\ (B_p\ )$, since $z\ (B_p\ )$ is adjacent to both x and y. Let u_1 and u_2 be the two vertices of B_p on P that are closest to x and y, respectively, along P. If there is exactly one vertex of B_p on P, then u_1 and u_2 can, in fact, be identical. It is clear that u_1 and u_2 are adjacent in P to vertices c_1 and c_2 respectively, such that c_1 and c_2 are non-adjacent in P and $c_1, c_2 \in C$. By definition, c_1 and c_2 are in $A\ (C\ , B_p\ )$. By assumption, G_A is a clique. Thus, edge $<c_1, c_2>$ exists in G, and P is not a chordless path. This contradiction establishes that $\{x\ , y\ \}$ is a two-pair of G.

Q.E.D.

Before we move on to implementation details, we remark that Theorem 1 establishes that W.T. graphs are strict quasi-parity graphs. By Meyniel's theorem, we have thus obtained an alternate proof that W.T. graphs are perfect.

3. An $O\ (e \cdot v\ )$ Algorithm to find a Two-Pair

The algorithm we present in this section to find a two-pair in a W.T. graph is a direct implementation of the proof of Theorem 1. We will assume throughout this paper that a simple undirected graph G is represented in a computer by the adjacency-lists (or edge-lists) of the vertices of G.

Algorithm *Two-Pair (G)*.

Input: A W.T. graph G.

Output: $\{x\ , y\ \}$, a two-pair in G.

Step 1. Find the connected components of G. Let the components be $G_1, \cdots, G_l$. If $l > 1$, return "$\{x\ , y\ \}$ is a two-pair of G", where x is a vertex in G_1 and y is a vertex in G_2. STOP.

Step 2. {G is connected} Check if G is a clique. If yes, then return "G is a clique". STOP.

Step 3. $C \leftarrow$ *Min–Cutset* (G) {C is a minimal cutset of G }

Step 4. Find the connected components of $G - C$. Let their vertex sets be $B_1, \cdots, B_t$.

Step 5. Let H_C be the subgraph induced by C. Run through the edge-list of each vertex in C to check whether H_C is a clique. If H_C is not a clique, go to Step 9. If H_C is a clique and $t > 2$, set $k = 1$ and go to Step 8.

Step 6. {H_C is a clique and $t = 2$} Check if $G - B_1$ is a clique. If not, set $k = 1$ and go to Step 8. Next, check if $G - B_2$ is a clique. If not, set $k = 2$ and go to Step 8.

Step 7. {$t = 2$. $G - B_1$ and $G - B_2$ are both cliques} Return "{x, y} is a two-pair of G", where x is a vertex in B_1 and y is a vertex in B_2. STOP.

Step 8. {$G - B_k$ is not a clique} Return *Two-Pair* ($G - B_k$). STOP.

Step 9. {H_C is not a clique} Let $H_C{}^c$ be the complement of H_C.
$D \leftarrow$ *Connected–Component* (H_C) {D is the vertex set of a connected component of $H_C{}^c$ with at least two vertices}
If $D = C$ go to Step 11.

Step 10. {$H_C{}^c$ is disconnected} Return *Two-Pair* ($G - (C - D)$). STOP.

Step 11. {$H_C{}^c$ is connected} If $| B_p | > 1$ for some $p \in \{1, \cdots, t\}$ go to Step 13.

Step 12. {$| B_j | = 1$, for all j} Return *Two-Pair* (H_C). STOP.

Step 13. {$| B_j | > 1$ for some $j = p$} Run through the edge-list of each vertex in B_p to locate z (B_p) that is adjacent to all of C. Next, run through the edge-list of each vertex in $B_p - z$ (B_p) to find A (C , B_p). {A (C , B_p) is the set of vertices in C that are adjacent to some vertex of $B_p - z$ (B_p)}

Step 14. Check if G_A, the subgraph induced by A (C , B_p), is a clique. If yes, go to Step 16.

Step 15. {G_A is not a clique} Return *Two-Pair* (G_A). STOP.

Step 16. {G_A is a clique} Return *Two-Pair* (G_C). STOP.

Q.E.D.

The correctness of *Two-Pair* is obvious, given the proof of Theorem 1. We now show that *Two-Pair* runs in time bounded by $O(e \cdot v)$.

First, note that the steps that make recursive calls on *Two-Pair* are Steps 8, 10, 12, 15 and 16. At each stage of recursion, only one of these steps is executed. Each time, the recursive call is made on a graph with at least one less vertex. Therefore, the total number of recursive calls is bounded by $O(v)$. Let us now try to estimate the running time of each stage of recursion. Steps 1, 2, 4, 5, 6, 7, 11, 13 and 14 are easily implemented in $O(e+v)$ time, given that G is represented by the edge-lists of the vertices of G. The steps that have been left out so far are Steps 3 and 9. We now show how to implement these two steps in $O(e+v)$ time. We start with Step 3.

Algorithm *Min-Cutset* (G).

Input: A connected, non-clique graph G

Output: C, a minimal cutset of G

Step 1. Find a vertex v_k, such that $\deg(v_k) \neq v-1$. {Since G is not a clique, there is a vertex that is not adjacent to every other vertex of G }

Step 2. Let $u_1, \cdots, u_l$, the set of vertices adjacent to v_k, be denoted by $N(v_k)$. Find $C_1, \cdots, C_q$, the vertex sets of the connected components of $G - N(v_k)$. {Since $\deg(v_k) \neq v-1$, we have $q > 1$}

Step 3. {Initialize} For $i = 1, \cdots, q$, set $\text{rank}(C_i) = 0$. For $j = 1, \cdots, l$, set $\text{label}(u_j) = 0$.

{$\text{rank}(C_i)$ counts the number of distinct vertices in $N(v_k)$ that are adjacent to vertices in C_i. $\text{label}(u_j)$ indicates the highest index of the C_i such that u_j is adjacent to some vertex in the corresponding C_i }

Step 4. For $i = 1, \cdots, q$ repeat the following in order.

For every vertex $v_p \in C_i$, find all the vertices in $N(v_k)$ that it is adjacent to. Call this set $\bar{N}(v_p)$. Now, for every $u_j \in \bar{N}(v_p)$, if $\text{label}(u_j) < i$, make $\text{label}(u_j) = i$ and increment $\text{rank}(C_i)$ by 1.

Step 5. Let C_r be the component with minimum rank. In other words, $\text{rank}(C_r) = \min\{\text{rank}(C_i)\}$. Let C denote the set of vertices in $N(v_k)$ that are adjacent to vertices in C_r. Return C. STOP.

Q.E.D.

It is fairly easy to see that *Min-Cutset* (G) works in $O(e+v)$ time. The following lemma shows that it finds a minimal cutset of G.

Lemma 1. Algorithm *Min-Cutset* (G) returns C, a minimal cutset of G.

Connected-Component, below, is an implementation of Step 9 in $O(e+v)$ time.

Algorithm *Connected-Component* (G).

Input: A non-clique graph G

Output: D, the vertex set of some component (with at least two vertices) of G^c, the complement of G.

Step 1. Find a vertex v_k, such that $\deg(v_k)$ is minimum in G. Let $\delta = \deg(v_k)$.

Step 2. Find the $v - \delta - 1$ vertices in $N^c(v_k)$, the set of vertices that are adjacent to v_k in G^c. Let T be the tree where vertex v_k is adjacent to every vertex in $N^c(v_k)$.

Step 3. Construct the graph H which consists of T and all the edges of G^c that are incident on the vertices in $V - (N^c(v_k) \cup \{v_k\})$.

Step 4. Find a connected component of H. Let its vertex set be D. Return D. STOP.

Q.E.D.

It is easy to see that *Connected-Component* (G) finds D, the vertex set of a connected component of G^c with at least two vertices. The following lemma establishes that it runs in time $O(e+v)$.

Lemma 2. Algorithm *Connected-Component* (G) runs in time bounded by $O(e+v)$.

This completes our analysis of the complexity of *Two-Pair*. We have thus established that its running time is bounded by $O(e \cdot v)$.

4. Identification and Quasi-Identification

The algorithms for W.T. graphs are based on the crucial concepts of identification and quasi-identification. Identification is used in the construction of algorithms for the aforementioned optimization problems without weights and quasi-identification is used in the construction of algorithms for the corresponding problems with weights. Note that given a W.T. graph that is a clique, all of the above optimization problems can be solved easily. If the given W.T. graph is not a clique, the algorithms of [10] repeatedly find a two-pair, each time transforming the graph in question into a smaller W.T. graph by "identifying" the two-pair. Eventually the original graph is transformed into a clique; the optimization problem is solved for the clique, and the two-pair identification process is reversed, transforming the solution of the optimization problem for the clique to the solution of the optimization problem for the original graph. We now formally define identification.

Definition 1. Let x and y be vertices of G. Let $G\ (xy \rightarrow z\ )$ be the graph obtained by replacing vertices x and y of G by vertex z, such that z is adjacent in $G\ (xy \rightarrow z\ )$ to exactly those vertices of $G - \{x\ ,y\ \}$ that were adjacent to either x or y in G. The *identification* of x and y in G is the process of replacing G by $G\ (xy \rightarrow z\ )$.

The following result from [10] states that the W.T. property of G is maintained in $G\ (xy \rightarrow z\ )$, if x and y formed a two-pair in G.

Lemma 3. Let G be a W.T. graph with two-pair $\{x\ ,y\ \}$. Then, $G\ (xy \rightarrow z\ )$ is also W.T.

Having shown that identification preserves the property of being W.T., we next show that it also preserves the size of the maximum clique. In the following, let $\omega(G\ )$ represent the size of a largest clique in G. We first invoke a lemma of Meyniel [12].

Lemma 4. [12] If vertices x and y of a graph G are not joined by any chordless path with exactly three edges, then $\omega(G\ (xy \rightarrow z\ )) = \omega(G\ )$.

Lemma 5 follows as a corollary of Meyniel's result, and states that identifying a two-pair in a W.T. graph leaves the size of a maximum clique unchanged.

Lemma 5. [10] If $\{x\ ,y\ \}$ is a two-pair in a W.T. graph G, then $\omega(G\ (xy \rightarrow z\ )) = \omega(G\ )$.

We now turn out attention to the concept of quasi-identification. Let us denote by $G\ (u \rightarrow vw\ )$ the graph obtained from G by replacing the vertex u by vertices v and w, such that u, v and w are adjacent to the same set of vertices of $G - u$, and by then adding the edge $<v\ ,w>$. This process is called *duplication*.

Definition 2. Let $G\ (xy \rightarrow za\ )$ be the graph $H\ (xb \rightarrow z\ )$, where $H = G\ (y \rightarrow ab\ )$. The process of replacing G with $G\ (xy \rightarrow za\ )$ is called *quasi-identification*.

Our treatment of quasi-identification parallels our treatment of identification. The following lemma of [10] establishes that quasi-identification also preserves the property of being W.T.

Lemma 6. Let G be a W.T. graph with a two-pair $\{x, y\}$. Then $G(xy \rightarrow za)$ is W.T.

As mentioned earlier in this section, quasi-identification is used in the construction of algorithms for the weighted versions of the optimization problems for W.T. graphs. We will formally define these problems in a later section. For the moment, we only introduce the notion of weights and of maximum weighted cliques. For each vertex v_i in G, let us attach a positive integer $w(v_i)$. $w(v_i)$ will be referred to as the weight of vertex v_i. Let $V(K)$ be a subset of the vertices of G such that $V(K)$ induces a clique K in G. $|V(K)|$, the weight of $V(K)$, is defined by $|V(K)| = \sum_{v_i \in V(K)} w(v_i)$. Let $\Omega(G)$ denote the weighted clique number of G, that is, the maximum weight of any subset of the vertex set of G that induces a clique in G. The next lemma of [10] proves that quasi-identification preserves the weighted clique number of G.

Lemma 7. Let G be a W.T. graph with weights on its vertices. Let $\{x, y\}$ be a two-pair of G such that $w(x) \leq w(y)$. Let $F = G(xy \rightarrow za)$ and let $w(z) = w(x)$ and $w(a) = w(y) - w(x)$. Then, $\Omega(F) = \Omega(G)$.

5. The Unweighted Case

In this section, we provide the algorithm of Hayward, Hoang and Maffray [10] to find a maximum clique and a minimum coloring in a W.T. graph G. We then show that the algorithm indeed correctly outputs the desired objects. Finally, we analyze the running time of the algorithm based on our implementation of *Two-Pair* and show that it runs in time bounded by $O(e \cdot v^2)$. This improves on the running time of $O(e \cdot v^3)$ obtained in [10]. Note that running the same algorithm on G^c provides us with a maximum independent set and a minimum clique cover for G.

In the following algorithm, we specify a coloring by a function f_G that assigns some integer from 1 to t to each vertex, such that adjacent vertices are assigned different integers.

Algorithm *Opt* (G).

Input: A W.T. graph G

Output: The vertex set of a maximum clique $V(K_G)$ and a minimum coloring f_G

Step 1. If G is a clique, set $V(K_G) = V$, where V is the vertex set of G. For $i = 1, \cdots, |V|$, set $f_G(v_i) = i$. STOP.

Step 2. {G is not a clique}

$\{x, y\} \leftarrow Two\text{-}Pair\ (G)$

Step 3. {$\{x, y\}$ is a two-pair in G} Let J denote $G\ (xy \rightarrow z)$.

Return $Opt\ (J)$.

{We now have $V\ (K_J)$, the vertex set of a maximum clique in J, and f_J, a minimum coloring of J}

Step 4. If $z \notin V\ (K_J)$, then set $V\ (K_G) = V\ (K_J)$.

If $z \in V\ (K_J)$, then if x is adjacent to all of $V\ (K_J) - z$, set $V\ (K_G) = V\ (K_J) - z + x$, otherwise set $V\ (K_G) = V\ (K_J) - z + y$.

{We will prove below that if $z \in V\ (K_J)$ then one of x, y is adjacent to all of $V\ (K_J) - z$}

Step 5. Set $f_G\ (x) = f_G\ (y) = f_J\ (z)$.

For every other vertex v_i in G, set $f_G\ (v_i) = f_J\ (v_i)$. STOP.

Q.E.D.

We now show that $Opt\ (G)$ correctly computes a maximum clique and a minimum coloring of G.

Lemma 8. Algorithm $Opt(G)$ computes the vertex set of a largest clique and a minimum coloring of G.

A corollary of Lemma 8 is that the size of a maximum clique if G is equal to the size of a minimum coloring of G. G^c, the complement of G is also W.T. By applying this corollary to G^c, we have that the size of a maximum independent set of G is equal to the size of a minimum clique cover of G. Since every induced subgraph of a W.T. graph is W.T., we have that this property holds for every induced subgraph of G. This yields yet another proof that W.T. graphs are perfect.

We now analyze the complexity of *Opt*. Step 2 takes $O\ (e \cdot v)$ time, as we have shown earlier. Steps 1, 4 and 5 clearly take $O(e + v)$ time. Since Step 3 is executed at most v times, the running time of *Opt* is bounded by $O\ (e \cdot v^2)$.

6. The Weighted Case

In this section, we present the algorithm of Hayward, Hoang and Maffray [10] to solve the weighted versions of the problems solved in the previous section. We then show the correctness of their algorithm. Finally, we analyze the algorithm based on our implementation of *Two-Pair* and show that it runs in time $O(e \cdot v^3)$. This improves on the running time of $O(e \cdot v^4)$ obtained by Hayward, Hoang and Maffray.

In each of the following problems, each vertex v_i of G has a positive integer $w(v_i)$ associated with it. $w(v_i)$ is called the *weight* of v_i.

The Maximum Weighted Clique Problem. Find a subset $V(K)$ of the vertices of G, such that $V(K)$ induces a clique K in G and such that the sum of the weights of the vertices in $V(K)$ is maximum over all such subsets that induce cliques in G.

The Maximum Weighted Independent Set Problem. Find an independent set S in G such that the sum of the weights of the vertices in S is maximum over all independent sets in G.

The Minimum Weighted Coloring Problem. Find independent sets $S_1, \cdots, S_t$ and integers $X_1, \cdots, X_t$ such that

(1) For every vertex v_j, the sum of integers X_i of all sets S_i such that $v_j \in S_i$ is at least $w(v_j)$, and

(2) the sum $X_1 + \cdots + X_t$ is minimum over all sets of integers that satisfy property (1).

The Minimum Weighted Clique Cover Problem. Find vertex subsets $V(K_1), \cdots, V(K_t)$, such that each $V(K_i)$ induces a clique in G, and integers $X_1, \cdots, X_t$ such that

(1) For every vertex v_j, the sum of integers X_i of all sets $V(K_i)$ such that $v_j \in V(K_i)$ is at least $w(v_j)$, and

(2) the sum $X_1 + \cdots + X_t$ is minimum over all sets of integers that satisfy property (1).

Note that an algorithm which solves any of the above weighted problems can be used to solve the corresponding unweighted problem. Such a solution is, however, less efficient than the direct implementation shown in the previous section.

We present an algorithm *W-Opt* that solves the maximum weighted clique problem and the minimum weighted coloring problem. As before, running *W-Opt* on G^c provides solutions to the maximum weighted independent set problem and the minimum weighted clique cover problem. In *W-Opt*, the weighted coloring f_G consists of independent sets $S_{G_1}, \cdots, S_{G_t}$ and the associated positive integers $X_{G_1}, \cdots, X_{G_t}$.

Algorithm *W-Opt* (G).

Input: A W.T. graph G and a weight function w on the vertices of G.

Output: The vertex set of a maximum weighted clique $V(K_G)$ and a minimum weighted coloring f_G

Step 1. If G is a clique, set $V(K_G) = V$, where V is the vertex set of G. For $i = 1, \cdots, |V|$, set $S_{G_i} = \{v_i\}$ and $X_{G_i} = w(v_i)$.

Return $V(K_G), f_G$. STOP.

Step 2. {G is not a clique} $\{x, y\} \leftarrow Two\text{-}Pair\ (G)$

Step 3. {$\{x, y\}$ is a two-pair in G}

If $w(x) = w(y)$ then set $J = G(xy \rightarrow z)$ and set $w(z) = w(x)$.

If $w(x) < w(y)$ then set $J = G(xy \rightarrow za)$, $w(z) = w(x)$ and $w(a) = w(y) - w(x)$.

Step 4. Return *W-Opt* (J).

{We now have $V(K_J)$, the vertex set of a maximum weighted clique in J, and f_J, a minimum weighted coloring of J}

Step 5. If $z \notin V(K_J)$, then set $V(K_G) = V(K_J)$.

If $z \in V(K_J)$, then if x is adjacent to all of $V(K_J) - \{a, z\}$,

set $V(K_G) = V(K_J) - \{a, z\} + x$,

otherwise set $V(K_G) = V(K_J) - \{a, z\} + y$.

Step 5. For each set S_{J_i} of f_J perform all of the following.

If $z \in S_{J_i}$ then set $S_{G_i} = S_{J_i} - z + \{x, y\}$. If $z \notin S_{J_i}$ and if $a \in S_{J_i}$ then set $S_{G_i} = S_{J_i} - a + y$. If neither z nor a is in S_{J_i}, then set $S_{G_i} = S_{J_i}$.

Set $X_{G_i} = X_{J_i}$. STOP.

Q.E.D.

The proof of correctness of *W-Opt* parallels the proof of correctness of *Opt*.

Lemma 9. Algorithm *W-Opt*(G) correctly finds a maximum weighted clique and a minimum weighted coloring in a W.T. graph G.

We now analyze the complexity of *W-Opt*. We have established that Step 2 can be done in time $O(e \cdot v)$. Steps 1, 4 and 5 can be done in $O(v)$ time. In Step 3, the graph J is either $G(xy \rightarrow z)$ or $G(xy \rightarrow za)$. In the former case, J has one vertex fewer than G. In the latter case, J has at least one edge more than G. To see this, we reason as follows. z is adjacent to every vertex of $G - \{x, y\}$ that x is adjacent to, a is adjacent to every vertex of $G - \{x, y\}$ that y is adjacent to, and edge $<z, a>$ is in J whereas x is non-adjacent to y. Thus, Step 3 is executed at most $v - 1 + \frac{v \cdot (v-1)}{2} - e$ times. Therefore the running time of *W-Opt* is bounded by $O(e \cdot v^3)$.

7. Conclusions and Open Problems

The main contributions of this paper are Theorem 1 and the implementations of Section 3. We have retained the basic algorithms of Hayward, Hoang and Maffray, while speeding up the crucial step that finds a two-pair in W.T. graphs.

One interesting observation to make is that *Two–Pair* is executed $O(v)$ times. In each execution of *Two–Pair*, all we need to assume is that we are given a W.T. graph as input. However, we observe that the input W.T. graphs in each successive execution of *Two–Pair* are closely related to each other. For instance, in *Opt*, two successive input graphs are identical except that a two-pair in one is identified in the other. Thus, it seems likely that one can find the $v/2$ two-pairs of a W.T. graph in time less than $O(e \cdot v^2)$ by making use of the information obtained in each run of *Two–Pair*. This brings us to the first open problem.

Open Problem 1. Can we find $v/2$ two-pairs of a W.T. graph in $O(e \cdot v)$ time?

The other open problem concerns strict quasi-parity and quasi-parity graphs. As mentioned earlier, these are closely related to W.T. graphs. Further, it has been conjectured [15] that the visibility graph for another version of polygon decomposition problem is a strict quasi-parity graph. Yet, the only known polynomial algorithm to optimize these graphs is the algorithm of Grötschel, Lov´asz and Schrijver. Thus, we pose the following problem.

Open Problem 2. Can we design a combinatorial algorithm to find maximum clique, a minimum coloring, a maximum independent set and a minimum clique cover of strict quasi-parity and quasi-parity graphs that runs in polynomial time?

References

1. C. Berge, "Färbung von Graphen deren sämtliche bzw. ungerade Kreise starrsind (Zusammenfassung)," *Wiss. Z. Martin-Luther-Univ. Halle-Winterberg, Math.-Natur. Reihe*, vol. 114, 1961.
2. J. Culberson and R. Reckhow, "Orthogonally Convex Coverings of Orthogonal Polygons without Holes," *JCSS (to appear).*
3. J. Culberson and R. Reckhow, "Dent Diagrams: A Unified Approach to Polygon Covering Problems," *Tech. Rep. TR 87-14, Dept. of Computing Science, Univ. of Alberta*, July, 1987.
4. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, p. 193, W. H. Freeman and Company, San Francisco, 1979.
5. F. Gavril, "Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques and Maximum Independent Set of a Chordal Graph," *SIAM J. Computing*, vol. 1, 2, pp. 180-187, June 1972.
6. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
7. M. Grötschel, L. Lov´asz, and A. Schrijver, "The Ellipsoid Method and its Consequences on Combinatorial Optimization," *Combinatorica 1*, pp. 169-197, 1981.
8. A. Hajnal and J. Suranyi, "Uber die Auflösung on Graphen in vollständig Teilgraphen," *Ann. Univ. Sci. Budapest Eötvös. Sect. Math.*, vol. 1, pp. 113-121, 1958.
9. R. B. Hayward, "Weakly Triangulated Graphs," *J. of Comb. Theory, Series B 39*, pp. 200-209, 1985.
10. R. B. Hayward, C. Hoang, and F. Maffray, "Optimizing Weakly Triangulated Graphs," *Graphs and Combinatorics (to appear)*, 1987.
11. L. Lov´asz, "Perfect Graphs," *Selected Topics in Graph Theory, 2*, pp. 55-85, Academic Press, Inc., London, 1983. Lowell W. Beineke and Robin J. Wilson, eds.
12. H. Meyniel, "A New Property of Critically Imperfect Graphs and some Consequences," *manuscript.*
13. R. Motwani, A. Raghunathan, and H. Saran, "Perfect Graphs and Orthogonally Convex Covers," *SIAM Journal on Discrete Mathematics (to appear)*, 1988.
14. R. Motwani, A. Raghunathan, and H. Saran, "Covering Orthogonal Polygons with Star Polygons: The Perfect Graph Approach," *Proc. 4th. Annual ACM Symposium on Computational Geometry*, pp. 211-223, 1988.
15. A. Raghunathan, "Polygon Decomposition and Perfect Graphs," *Ph.D. Thesis*, Computer Science Division, University of California, Berkeley, 1988.
16. R. Reckhow, "Covering Orthogonally Convex Polygons with Three Orientations of Dents," *Tech. Rep. TR 87-17*, Dept. of Computing Science, Univ. of Alberta, Aug. 1987.

17. R. Reckhow and J. Culberson, "Covering a Simple Orthogonal Polygon with a Minimum Number of Orthogonally Convex Polygons," *Proc. 3rd. Annual ACM Symposium on Computational Geometry*, pp. 268-277, June 1987.

18. A. Schrijver, *Theory of Linear and Integer Programming*, pp. 170-171, Wiley-Interscience Series in Discrete Mathematics and Optimization, New York, 1986.