Stability number and chromatic number of tolerance graphs

Giri Narasimhan* and Rachel Manber**

Computer Sciences Department, University of Wisconsin-Madison, Madison, WI 53706, USA

Received 30 August 1988
Revised 1 August 1989

Abstract

Golumbic and Monma [3] introduced a subclass of perfect graphs called tolerance graphs. In this paper, we present algorithms to compute the stability number, the clique number, the chromatic number, and the clique cover number of a tolerance graph.

1. Introduction

Many subclasses of perfect graphs frequently appear in real-life applications. These include, among others, the classes of interval graphs, permutation graphs, comparability graphs, and co-comparability graphs [2]. Recently, Golumbic and Monma [3] introduced a new subclass of perfect graphs called tolerance graphs.

An undirected graph $G(V, E)$ is a tolerance graph if there exists a collection $\mathcal{I} = \{I \mid I \in V\}$ of closed intervals on a line and a multiset $\mathcal{J} = \{t_v \mid v \in V\}$ such that

$$(x, y) \in E \iff |x \cap y| \geq \min(t_x, t_y).$$

Here $|x|$ denotes the length of interval $x$. The number $t_v$ is the tolerance of $v$. We say that two intervals conflict if their intersection rises above a threshold, which is equal to the minimum of the tolerances of the two intervals. Thus, a graph is a tolerance graph if there exists a pair $\langle \mathcal{I}, \mathcal{J}\rangle$ such that

$$(x, y) \in E \iff x \text{ and } y \text{ conflict}.$$
The pair \( (\mathcal{A}, \mathcal{F}) \) is called a **tolerance representation** of \( G \). For example, \( C_4 \) - the simple cycle of length 4, is a tolerance graph. Its tolerance representation is given in Fig. 1.

Golumbic et al. [4] have shown that the complement of a tolerance graph is perfect orderable, thus proving that all tolerance graphs are perfect. It was also shown in [3] that the class of tolerance graphs properly contains the classes of interval graphs and permutation graphs. Like interval graphs, tolerance graphs have applications in scheduling. Tolerance graphs can model situations in which the intervals can tolerate a certain degree of overlap. Specific examples can be found in [4].

An interval in a tolerance representation is **bounded** if its tolerance does not ex-
ceed its length, otherwise it is \textit{unbounded}. A tolerance representation is \textit{bounded} if all its intervals are bounded. A tolerance graph is a \textit{bounded tolerance} graph if it admits a bounded tolerance representation. The tolerance representation for $C_4$ given in Fig. 1, was not a bounded tolerance representation since the tolerance of interval $5$ was $6$, while its length was only $3$. However, $C_4$ is a bounded tolerance graph since it admits a bounded tolerance representation (see Fig. 2). Golumbic and Monma [3] showed that every bounded tolerance graph is a co-comparability graph. A \textit{co-comparability graph} [2] is a graph whose complement is a comparability graph; that is, it's complement can be transitively oriented.

The algorithmic aspects of tolerance graphs have not been studied. Since tolerance graphs are perfect we know how to find in polynomial time the following four parameters:

- \textit{the stability number}—the size of the largest independent set,
- \textit{the clique number}—the size of the largest clique,
- \textit{the chromatic number}—the least number of stable sets needed to cover the vertex set, and
- \textit{the clique cover number}—the fewest number of cliques needed to cover the vertex set.

In fact, since for perfect graphs the chromatic number equals the clique number and the stability number equals the clique cover number, it suffices to compute only two of these parameters. The algorithms to compute these parameters for perfect graphs use the ellipsoid method [5] and hence they are not very efficient. For most known subclasses of perfect graphs there exist more efficient algorithms to determine the values of these parameters. Moreover, in some cases the algorithms are constructive. For example, the algorithm to compute the stability number for a co-comparability graph finds an independent set of maximum size [2]. Similarly, the algorithm to determine the chromatic number of comparability graphs does in fact produce an optimal coloration [2]. Since bounded tolerance graphs are co-comparability graphs, all known algorithms on co-comparability graphs will apply. These include polynomial time algorithms for computing all the four parameters mentioned above. We present polynomial time algorithms to solve these problems for general tolerance graphs.

Given a tolerance representation, its corresponding tolerance graph can be constructed in polynomial time. In contrast, the recognition problem for the class of tolerance graphs is yet unsolved. Even when the input graph is known to be a tolerance graph, it is not known how to obtain a tolerance representation for it. Moreover, given a tolerance graph it is not known how to decide in polynomial time whether it is a bounded tolerance graph. In view of these remarks, we assume that along with the input graph $G = (V, E)$, we are given a tolerance representation $(\mathcal{I}, \mathcal{T})$ of $G$. The interval corresponding to a vertex $v \in V$ is $I(v) = (l(v), r(v))$. Following [4] we assume that the tolerance representation has intervals all of whose endpoints are distinct. Such a tolerance representation we call a \textit{regular} representation. The tolerance representations of $C_4$ given in Figs. 1 and 2 are both regular.
2. Maximum independent set

The intervals in a tolerance representation of a tolerance graph can be partitioned into two sets of intervals. One set $B = \{b_1, \ldots, b_p\}$ consists of all bounded intervals and the other set $U = \{u_1, \ldots, u_q\}$ consists of all the unbounded intervals. Without loss of generality, we assume that $r(b_1) < r(b_2) < \cdots < r(b_p)$, and $r(u_1) < r(u_2) < \cdots < r(u_q)$. This partition induces a partition of the vertices into the two sets of vertices $B = \{b_1, \ldots, b_p\}$ and $U = \{u_1, \ldots, u_q\}$. We refer to vertices in $B$ as bounded vertices and to vertices of $U$ as unbounded vertices. The subgraphs induced by these sets are $G_B$, which is a co-comparability graph, and $G_U$ which is an independent set. Our algorithm to find the largest independent set in a tolerance graph $G$ transforms its bounded part into a weighted directed graph whose weights are a function of the unbounded part of $G$. We first describe the algorithm to compute the stability number of a co-comparability graph.

The algorithm to compute the stability number of a co-comparability graph $G$ computes the clique number of its transitively orientable complement $G^c$. The computation of the clique number of a comparability graph is based on the fact that a clique in a comparability graph corresponds to a directed path in its transitive orientation [2]. Consequently, a maximum clique in a comparability graph corresponds to the longest path in its transitive orientation. Although the longest path problem is NP-complete on general graphs, it can be computed in linear time for a digraph obtained as a transitive orientation of a comparability graph since this digraph is acyclic [2]. A transitive orientation of a comparability graph can be computed in $O(\delta |E|)$, where $\delta$ is the maximum degree of a vertex in $G$ [2]. Thus, the time complexity to determine the stability number of a co-comparability graph is $O(|V|^2 + \delta |E|)$. In fact, the algorithm can actually find the largest independent set in the given co-comparability graph since we can easily recover the list of vertices along the longest path in the transitive orientation of its complement.

We can find the largest independent set in a bounded tolerance graph $G$ in polynomial time simply because they are co-comparability graphs. However, since along with $G$ we are given a tolerance representation for $G$, we can use it to transitively orient the complement of $G$ in linear time. Following [4], we define the right endpoint orientation of $G^c$ as follows. An edge $(x, y)$ is oriented from vertex $x$ to vertex $y$ if in the given tolerance representation of $G$, interval $x$ terminates before interval $y$. It is not hard to see that a right endpoint orientation of a bounded tolerance graph is transitive [4]. Thus, a transitive orientation of a bounded tolerance graph can be found in time linear in the size of $G^c$. It follows that the maximum independent set in a bounded tolerance graph can be found in $O(|V|^2)$.

We now extend the procedure for bounded tolerance graphs by presenting an algorithm to find the maximum independent set in a general tolerance graph. We reduce the problem of finding the maximum independent set in a tolerance graph $G$ to that of finding the longest (heaviest) path in an acyclic weighted directed graph $H(G)$. The digraph $H(G)$ consists of the right endpoint orientation of the comple-
ment of $G_B$ (which is the subgraph induced by the bounded vertices) together with two additional vertices, a source $s$ and a sink $t$. The source is joined to all the vertices in $B \cup \{t\}$ and the edges are oriented from $s$. The sink is joined to all the vertices of $B$ and the edges are oriented to $t$. Let $G'$ be the graph obtained from $G$ by adding to it the independent set $\{s, t\}$ and set $t_s = 0$ and $t_t = 0$. Extend the tolerance re-

\[ t_a = t_c = t_e = 2 \quad t_b = t_d = t_f = 8 \]

Graph $G$

\[ H(G) \]

Fig. 3. (a) Tolerance representation of (b) tolerance graph $G$, and (c) the constructed directed graph $H(G)$. 
presentation of \( G \) to a tolerance representation for \( G' \) by adding the intervals \( s \) and \( \bar{t} \) such that \( s \) (respectively \( \bar{t} \)) starts and terminates before (respectively after) all intervals in \( \mathcal{S} \). Then \( H(G) \) is simply the right endpoint orientation of the complement of \( G'_b \). In other words, the vertex set of \( H(G) \) is \( B \cup \{s, t\} \) and there is a directed edge from \( x \) to \( y \) if \( x \) and \( y \) are not adjacent in \( G' \), and interval \( x \) terminates before interval \( y \).

We associate a set valued function \( S(e) \) and weight function \( w(e) \) with each directed edge \( e = (x, y) \) in \( H(G) \). The set \( S(e) \) consists of all unbounded vertices \( u_k \in G \) that are not adjacent either to \( x \) or \( y \) and whose corresponding unbounded intervals \( \bar{u}_k \) terminate after \( r(x) \) and before \( r(y) \). Note that for a directed edge \( e = (s, b) \) in \( H(G) \), \( S(e) \) consists of those unbounded vertices \( u_k \in G \) which are not adjacent to \( b \) and whose corresponding unbounded intervals \( \bar{u}_k \) terminate before \( \bar{b} \) does. A similar statement holds for edges directed towards the sink \( t \), with the word "after" replacing the word "before". For the special edge joining \( s \) and \( t \) we have \( S((s, t)) = U \). It follows that each set \( S(e) \) is an independent set. The weight function \( w \) is defined as follows:

\[
w(e) = \begin{cases} 
|S(e)|, & \text{if } e = (b, t); \\
|S(e)| + 1, & \text{otherwise}.
\end{cases}
\]

The motivation for the definition of the weight function will become apparent later.

The construction of \( H(G) \) for a given tolerance graph \( G \) is illustrated in Fig. 3 in which

\[
S((s, c)) = S((a, c)) = S((a, t)) = \{f\}, \quad S((s, t)) = \{b, d, f\},
\]

and for all other edges \( S(e) = \emptyset \).

We first show a few properties of \( H(G) \). This is done in the next three lemmas. The first of these lemmas shows the relation between an edge \( e = (b_i, b_j) \) in \( H(G) \) and the positions of the intervals \( \bar{b}_i \) and \( \bar{b}_j \) relative to each other.

**Lemma 2.1.** Let \( e = (b_i, b_j) \) be an edge in \( H(G) \) with \( i < j \). Then \( l(b_i) < l(b_j) \) and \( r(b_i) < r(b_j) \).

**Proof.** The fact that \( e \) is an edge of \( H(G) \) implies that \( \bar{b}_i \) and \( \bar{b}_j \) do not conflict in the tolerance representation of \( G \). This, together with the fact that both \( \bar{b}_i \) and \( \bar{b}_j \) are bounded intervals, imply that neither one of them can contain the other. Since the vertices in \( B \) are ordered by their right endpoint, and since containment is excluded, the right endpoint orientation implies that \( l(b_i) < l(b_j) \) and \( r(b_i) < r(b_j) \).

Let \( P = \{e_0 = (s, b_i), e_1 = (b_i, b_j), \ldots, e_k = (b_k, t)\} \) be a directed path from \( s \) to \( t \) in \( H(G) \). The set of internal vertices of \( P \), \( \{b_i, b_j, \ldots, b_k\} \), is denoted by \( B_P \). The next lemma follows from the construction of \( H(G) \).
Lemma 2.2. Every two internal vertices of a path in $H(G)$ are joined by an edge in $H(G)$.

Since $B_p$ is transitive, $B_p$ is an independent set in $G$. The next lemma shows that we can extend the independent set $B_p$ which consists of bounded vertices to include unbounded vertices.

Lemma 2.3. Let $P = \{e_0 = (s, b_i), e_1 = (b_i, b_i), \ldots, e_k = (b_i, t)\}$ be a directed path from $s$ to $t$ in $H(G)$. Then $S(e_j) \cup B_p$ is an independent set in $G$, $1 \leq j \leq k$.

Proof. The sets $S(e_j)$ and $B_p$ are each independent sets in $G$. So we need only to show that there is no edge with one endpoint in $S(e_j)$ and the other endpoint in $B_p$. That is, it suffices to show that if $u \in S(e_j)$ ($0 \leq j \leq k$) and $b_i \in B_p$ ($1 \leq i \leq k$), then the vertices $u$ and $b_i$ are not adjacent in $G$. In other words, we need to show that the intervals $u$ and $b_i$ do not conflict.

Assume first that $1 \leq l < j < k$. Since $b_i$ and $b_j$ are internal vertices of $P$ and $l < j$, Lemmas 2.1 and 2.2 imply that $l(b_i) < l(u)$ and $r(u) < r(b_i)$. The fact that $u$ is in $S(e_j)$ means that $u$ does not conflict with either $b_i$ or $b_{j+1}$. In particular this implies that $u$ cannot contain either $b_i$ or $b_{j+1}$. It follows that the left endpoints and the right endpoints of intervals $b_i$, $b_j$, and $u$ are ordered as follows: $l(b_i) < l(b_j) < l(u)$, and $r(u) < r(b_i) < r(b_j)$. Hence $|u \cap b_j| < |b_j \cap b_i|$. Furthermore, the intervals $b_j$ and $b_i$ do not conflict and hence $|b_i \cap b_j| < \min\{t_{b_i}, t_{b_j}\} = t_{b_i}$. It follows that $u$ and $b_i$ do not conflict and hence $u$ and $b_i$ are not adjacent in $G$.

The case $j = k$ follows verbatim if we let $t = b_{j+1}$. In this case it is consistent with our definitions to replace $G$ by $G'$. This assumption is justified since the vertex $t$ represents a bounded interval with tolerance 0 which starts (and terminates) after all other intervals. A dual argument can be used to handle the case $1 \leq j + 1 < l \leq k$. Finally, if $l = j$ or if $l = j + 1$, then by the definition of $S(e_j)$, $(u, b_i) \in E(G)$.

Lemma 7.3 implies that an independent set in $G_B$ consisting of internal vertices of a path $P$ from $s$ to $t$ in $H(G)$, can be extended to an independent set $I_P = B_p \cup S(e_0) \cup S(e_1) \cup \cdots \cup S(e_k)$ in $G$. In the example of Fig. 3 there are three paths of total weight 3, $P_1 = ((s, t), (s, a), (a, c), (c, t))$, and $P_2 = ((s, a), (a, c), (c, t))$. Their corresponding independent sets are $I_{P_1} = \{b, d, f\}$, $I_{P_2} = \{a, c, f\}$, and $I_{P_3} = \{a, c, e\}$. Thus, selecting edge $e = (b_i, b_j)$, $1 \leq j \leq k$ to be included in a path $P$ from $s$ to $t$ in $H(G)$ is equivalent to selecting $S(e) \cup \{b_i\}$ to be included in the independent set $I_P$. Since the sets $S(e_j)$ are disjoint, this means that each edge in the path, except the last edge, identifies $|S(e)| + 1 = w(e)$ vertices in the corresponding independent set. The last edge identifies only $|S(e)| = w(e)$ vertices. In other words, each path from $s$ to $t$ in $H(G)$ corresponds to an independent set in $G$ whose size is the sum of the weights of the edges in the path. This observation is the basis for
our algorithm to find an independent set of maximum size in a general tolerance graph.

**Theorem 2.4.** Given a tolerance graph \( G = (V, E) \) and a regular tolerance representation \( \langle \mathcal{A}, \mathcal{F} \rangle \) of \( G \). There is an \( O(|B|^2 \log(|U| + 1)) \) algorithm to find the largest independent set in \( G \), where \( |B| \) and \( |U| \) are the numbers of bounded and unbounded vertices in \( G \).

**Proof.** We first construct the weighted directed graph \( H(G) \). Let \( P \) be the longest weighted path from \( s \) to \( t \) in \( H(G) \). Denote \( s \) by \( b_i \) and denote \( t \) by \( b_{k+1} \). We claim that the largest independent set in \( G \) is

\[
S = \bigcup_{i=0}^{k} S(e_i) \cup B_p.
\]

Lemma 2.3 implies that \( S \) is an independent set. In order to show that \( S \) is an independent set of maximal cardinality, it suffices to show that every independent set \( S \) in \( G \) is equal to \( \bigcup_{i=0}^{k} S(e_i) \cup B_p \) for some path \( P \) in \( H(G) \). So let \( S \) be an independent set in \( G \). Let \( S_B = \{s, b_1, b_2, \ldots, b_k, t\} \) be an ordered set consisting of \( s, t \) and all bounded vertices in \( S \) ordered by the right endpoints of the corresponding intervals. Let \( S_U \) be the set of all unbounded intervals in \( S \). If \( S_B \) has no internal vertices, then \( P \) consists of the single edge \((s,t)\) and \( S = S_U = U \). Otherwise, the independent set \( S_B \) in \( G' \) induces a clique in \( G'^c \). Moreover, the directed edges, \( e_j = (b_{j-1}, b_j, \ldots, b_1, t) \), joining consecutive vertices in \( S_B \) form a directed path \( P \) from \( s \) to \( t \) in the right endpoint orientation of \( G'^c \). That is \( P \) is a path in \( H(G) \). We can now partition \( S_U \) into \( k+1 \) subsets \( S(e_j) \) for \( 0 \leq j \leq k \). In this partition a vertex \( u \in S_U \) belongs to \( S(e_j) \) if \( r(b_j) < r(u) < r(b_{j+1}) \).

The construction of the unweighted right endpoint orientation of \( H(G) \) can be done in \( O(|B|^2) \). The weight function \( w \) can be computed in time \( O(|B|^2 \log(|U| + 1)) \), assuming, as we did, that \( U \) is ordered according to their right endpoints. If \( U \) is empty, then this stage can be done in constant time. The longest weighted path in \( H(G) \) can be found in time linear in the size of \( H(G) \) [2]. The size of \( H(G) \) is \( O(|B|^2 + \log |w|) = O(|B|^2 + \log |U|) \), where \( |w| \) is the largest weight in \( H(G) \). All these steps combined yield a total time complexity of \( O(|B|^2 + |B|^2 \log |U|) = O(|B|^2 \log(|U| + 1)) \). \( \square \)

Note that when the input graph is a bounded tolerance graph, all the weights in \( H(G) \) are 1 except for edges joined to the sink. In this case our algorithm reduces to finding the longest path in an unweighted digraph, i.e., the usual algorithm for co-comparability graphs.

The next corollary follows from the fact that tolerance graphs are perfect.

**Corollary 2.5.** Given a tolerance graph \( G = (V, E) \) and a regular tolerance represen-
3. Coloring

In this section we discuss the coloring problem on tolerance graphs, and we show how to find the chromatic number of a general tolerance graph.

Since the complement of a bounded tolerance graph is a comparability graph, to compute the chromatic number of a bounded tolerance graph, we apply the corresponding algorithm designed for co-comparability graphs [2]. It is not hard to see that the problem of computing the optimal coloring of a bounded tolerance graph can be reduced to a minimum flow problem in a network obtained by adding a source and a sink vertex to the "right endpoint orientation" of the bounded tolerance graph \( G \) (also see [1]). The above observation implies that there is an \( O(|V|^3) \) algorithm to color a bounded tolerance graph with an optimal number of colors.

Before we show how to compute the chromatic number of a general tolerance graph, we first prove Theorem 3.1, which shows how to efficiently compute the clique number of a general tolerance graph.

**Theorem 3.1.** Given a tolerance graph \( G = (V, E) \) and a regular representation \( (\mathcal{I}, \mathcal{F}) \) of \( G \). There is an \( O(q|V|^3) \) algorithm to find the clique number of \( G \), where \( q \) is the number of unbounded intervals in the tolerance representation of \( G \).

**Proof.** Let \( U \) be the set of unbounded vertices of \( G \). Then \( U \) is an independent set and hence any clique in \( G \) contains at most one vertex from \( U \). Let \( K \) be a set of vertices that form a clique in \( G \). If \( K \) contains an unbounded vertex, then \( K \setminus \{u\} \) forms a clique in the neighborhood \( N(u) \) of \( u \). It follows that a maximum clique in \( G \) is either a maximum clique in \( G_B \), or it consists of a maximum clique in \( N(u) \), together with \( u \) for some \( u \in U \). Both \( G_B \) and \( N(u) \) are co-comparability graphs. As discussed earlier, the clique number of co-comparability graphs can be computed via a network flow algorithm. These observations lead to a simple algorithm to compute the clique number in general tolerance graphs.

The algorithm finds the clique number \( k_0 \) of \( G_B \), and the clique numbers \( k_u \) of \( N(u) \) for every \( u \in U \). The clique number of \( G \) is updated to the maximum of \( k_0 \) and \( \{k_u + 1 \mid u \in U\} \). The algorithm performs \( q + 1 \) iterations of the algorithm to compute the clique number of co-comparability graphs. Since the latter has time complexity of \( O(|V|^3) \), the total time complexity of our algorithm is \( O(q|V|^3) \).

Since tolerance graphs are perfect, Corollary 3.2 follows from Theorem 3.1 above.

**Corollary 3.2.** Given a tolerance graph \( G = (V, E) \) and a regular tolerance representation \( (\mathcal{I}, \mathcal{F}) \) of \( G \). There is an \( O(|V|^3) \) algorithm to find the clique cover number of \( G \).
There is an $O(q|V|^3)$ algorithm to find the chromatic number of $G$, where $q$ is the number of unbounded intervals in the tolerance representation of $G$.

4. Discussion

It may be noted that the optimal coloring of a general tolerance graph can be computed using the results in [5]. However, this algorithm would make use of the ellipsoid algorithm. Even though we have shown that the chromatic number of a general tolerance graph can be computed efficiently, it remains open to design an efficient algorithm that will compute the optimal coloring.

References