Stability Number and Chromatic Number of Tolerance Graphs

Giri Narasimhan
and
Rachel Manber

Computer Sciences Technical Report #788

September 1988

STABILITY NUMBER AND CHROMATIC NUMBER OF TOLERANCE GRAPHS

Giri Narasimhan[1] Rachel Manber[2]

Computer Sciences Department
University of Wisconsin-Madison
Madison, Wisconsin-53706.

August 19, 1988

ABSTRACT

An undirected graph $G(V, E)$ is a *tolerance graph* if there exists a collection $\mathcal{I} = \{\overline{v} | v \in V\}$ of closed intervals on a line and a multiset $\mathcal{T} = \{t_v | v \in V\}$ such that $(x, y) \in E \iff |\overline{x} \cap \overline{y}| \geq \min\{t_x, t_y\}$. Here $|\overline{x}|$ denotes the length of interval $\overline{x}$. We present algorithms to compute the chromatic number, the stability number, the clique number, and the clique cover number of tolerance graphs.

[1]Research Supported By A Grant From Wisconsin Alumni Research Foundation.
[2]Research Supported by an NSF grant # DMS-8702930

1 Introduction

Many sub-classes 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 [Go80]. Recently, Golumbic and Monma [GM82] introduced a new sub-class of perfect graphs called tolerance graphs.

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

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

Here $|\overline{x}|$ denotes the length of interval $\overline{x}$. The number t_v is the *Tolerance* of $\overline{v}$. 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 $\langle \mathcal{I}, \mathcal{T} \rangle$ such that

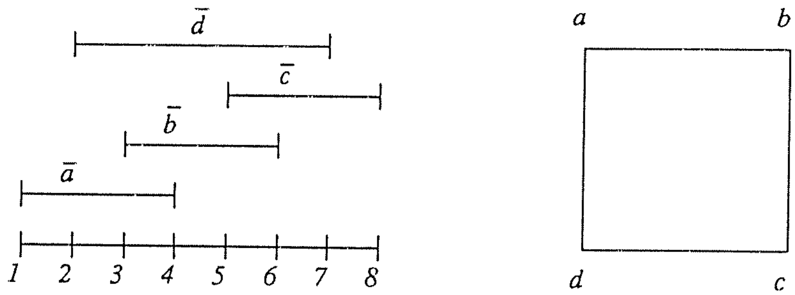
$$(x, y) \in E \iff \overline{x} \quad \text{and} \quad \overline{y} \quad \text{conflict.}$$

The pair $\langle \mathcal{I}, \mathcal{T} \rangle$ 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 Figure 1.

Trotter has shown that all tolerance graphs are perfect [GMT84]. In fact, the class of tolerance graphs properly contains both interval graphs and permutation graphs [GM82]. 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 [GMT84].

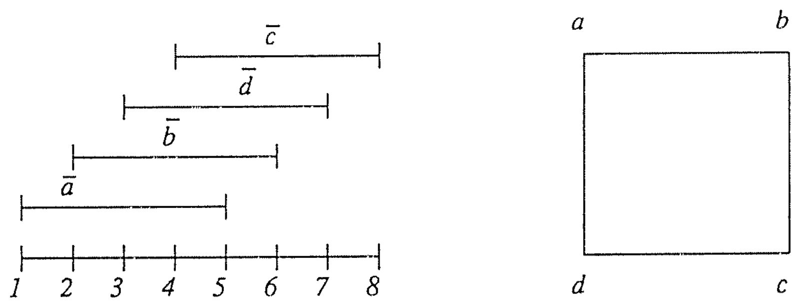
An interval in a tolerance representation is *bounded* if its tolerance does not exceed its length, otherwise it is *unbounded.* A tolerance representation is *bounded* if all its intervals are bounded. A tolerance graph is a *bounded tolerance* graph if it admits a bounded tolerance representation. The tolerance representation for C_4 given in Figure 1, was not a bounded tolerance representation since the tolerance of interval $\overline{b}$ was 6, while its length was only 3. However, C_4 is a bounded tolerance graph since it admits a bounded tolerance representation (see Figure 2). Golumbic and Monma [GM82] showed that every bounded tolerance graph is a co-comparability graph [3].

[3] *A co-comparability graph is a graph whose complement is a comparability graph; that is,*



$t_a = t_c = 1 \quad t_b = t_d = 6.$

Figure 1: A tolerance representation for C_4.



$t_a = t_c = 2 \quad t_b = t_d = 4.$

Figure 2: A bounded tolerance representation for C_4.

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:

- *the stability number* — the size of the largest independent set,
- *the clique number* — The size of the largest clique,
- *the chromatic number*,
- *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 [GLS81] 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 [Go80]. Similarly, the algorithm to determine the chromatic number of comparability graphs does in fact produce an optimal coloration [Go80]. 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

it's complement can be transitively oriented.

a tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$ of G. The interval corresponding to a vertex $v \in V$ is $\overline{v} = (l(v), r(v))$. Following [GMT84] we further assume that the tolerance representation satisfies the following properties: (a) the end points of the intervals are distinct; (b) the tolerances are all strictly positive; (c) any tolerance that is larger than the length of its corresponding interval is set to infinity; (d) the tolerances are all distinct (except those set to infinity); and (e) the intersection of all intervals is a nonempty interval. A tolerance representation satisfying the above five assumptions is a *regular* representation. The tolerance representations of C_4 given in Figures 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 $\overline{B} = \{\overline{b_1}, \ldots, \overline{b_p}\}$ consists of all bounded interval and the other set $\overline{U} = \{\overline{u_1}, \ldots, \overline{u_q}\}$ consists of all the unbounded intervals. Without loss of generality, we assume that $r(b_1) < r(b_2) < \ldots < r(b_p)$, and $r(u_1) < r(u_2) < \ldots < r(u_q)$. 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 [Go80]. Consequently, a maximum clique in a comparability graph corresponds to the longest path in its transitive orientation. Although the longest path problem is in general NP-complete, it can be computed in linear time for a digraph obtained as a transitive orientation of a comparability graph since this digraph is acyclic [Go80]. A transitive orientation of a comparability graph can be computed in $\mathcal{O}(\delta|E|)$, where δ is the maximum degree of a vertex in G [Go80]. Thus, the time complexity to determine the stability number of a co-comparability graph is $\mathcal{O}(|V|^2 + \delta|E|)$. 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 [GMT84], we define the *right end point 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 $\overline{x}$ terminates before interval $\overline{y}$. It is not hard to see that a right end point orientation of a bounded tolerance graph is transitive [GMT84]. 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 $\mathcal{O}(|V|^2)$.

We extend the procedure to find a maximum independent set for bounded tolerance graphs and present an algorithm to find the largest 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 end point orientation of the complement of G_B 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. One may think of s (resp. t) as representing an interval that starts and terminates before (resp. after) all the intervals in $\mathcal{I}$ and whose tolerance equals its length. Let G' be the graph obtained from G by adding to it the independent set $\{s, t\}$ and set $t_s = |s|$ and $t_t = |t|$. Extend the tolerance representation of G to a tolerance representation for G' by adding the intervals $\overline{s}$ and $\overline{t}$ such that $\overline{s}$ (resp. $\overline{t}$) starts and terminates before (resp. after) all intervals in $\mathcal{I}$. Then $H(G)$ is simply the right end point 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 $\overline{x}$ terminates before interval $\overline{y}$.

We associate a set-valued function $S(e)$ and a weight function $w(e)$ with each directed edge $e = (x, y)$ in $H(G)$. The set $S(e)$ consists of all unbounded vertices $u_k \in G$ which are not adjacent either to x or to y and whose corresponding unbounded intervals $\overline{u_k}$ 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 $\overline{u_k}$ terminate before $\overline{b}$ does. A similar statement holds for edges directed toward 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 Figure 3 in which

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

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

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 $\overline{b_i}$ and $\overline{b_j}$ relative to each other.

Lemma 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 $\overline{b_i}$ and $\overline{b_j}$ do not conflict in the tolerance representation of G. This, together with the fact that both $\overline{b_i}$ and $\overline{b_j}$ are bounded intervals, imply that neither one of them can contain the other. Since the vertices in B are ordered by their right end point, and since containment is excluded, the right end point orientation implies that $l(b_i) < l(b_j)$ and $r(b_i) < r(b_j)$. ■

Let $P = \{e_0 = (s, b_{i_1}), e_1 = (b_{i_1}, b_{i_2}), \ldots, e_k = (b_{i_k}, t)\}$ be a directed path from s to t in $H(G)$. The set of *internal vertices* of P, $\{b_{i_1}, b_{i_2}, \ldots, b_{i_k}\}$, is denoted by B_P. The next lemma follows from the construction of $H(G)$.

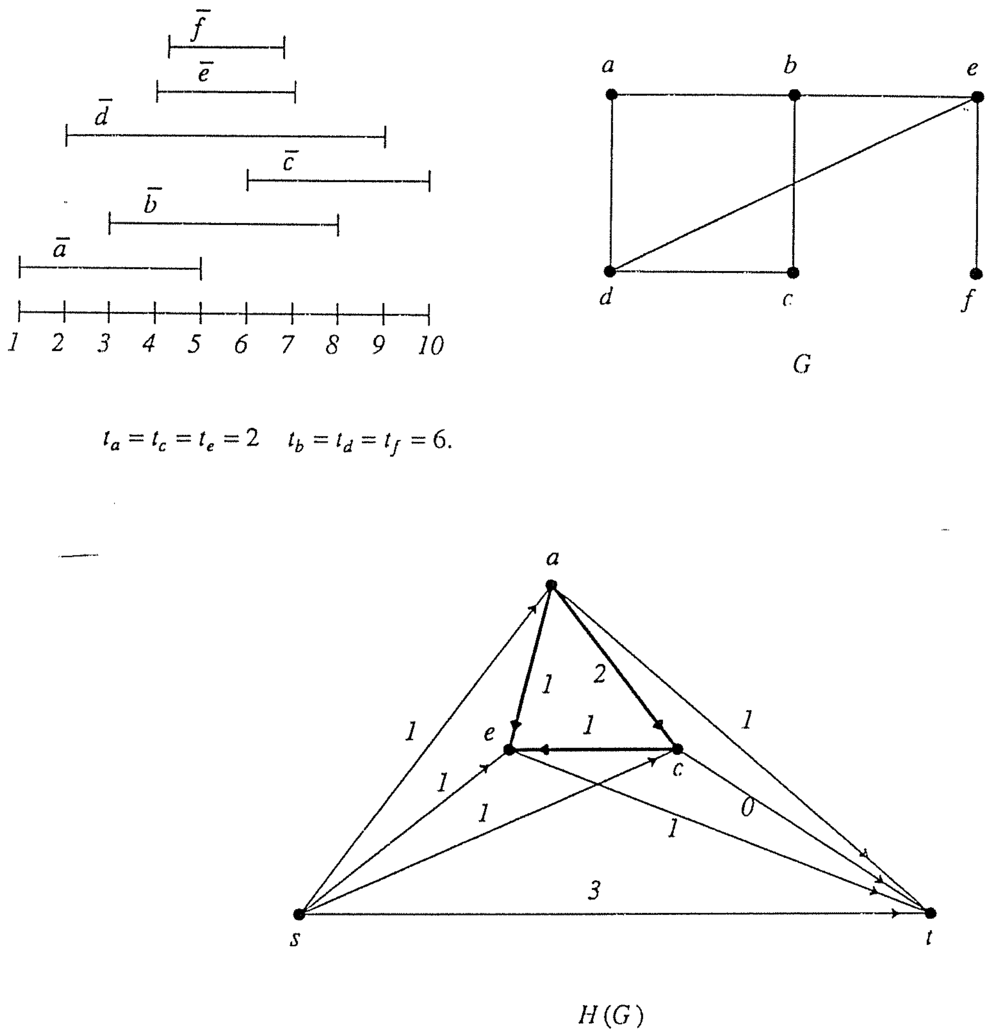


Figure 3: The construction of $H(G)$.

Lemma 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 3 *Let $P = \{e_0 = (s, b_{i_1}), e_1 = (b_{i_1}, b_{i_2}), \ldots, e_k = (b_{i_k}, t)\}$ 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 end point 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 $b_{i_l} \in B_P$ $(1 \leq l \leq k)$ then the vertices u and b_{i_l} are not adjacent in G. In other words, we need to show that the intervals $\overline{u}$ and $\overline{b_{i_l}}$ do not conflict.

Assume first that $1 \leq l < j < k$. Since b_{i_l} and b_{i_j} are internal vertices of P and $l < j$, Lemmas 1 and 2 imply that $l(b_{i_l}) < l(b_{i_j})$ and $r(b_{i_l}) < r(b_{i_j})$. The fact that u is in $S(e_j)$ means that $\overline{u}$ does not conflict with either $\overline{b_{i_j}}$ or $\overline{b_{i_{j+1}}}$. In particular this implies that $\overline{u}$ cannot contain either $\overline{b_{i_j}}$ or $\overline{b_{i_{j+1}}}$. It follows that the left end points and the right end points of intervals $\overline{b_{i_l}}$, $\overline{b_{i_j}}$, and $\overline{u}$ are ordered as follows: $l(b_{i_l}) < l(b_{i_j}) < l(u)$, and $r(b_{i_l}) < r(b_{i_j}) < r(u)$. Hence $|\overline{u} \cap \overline{b_{i_l}}| < |\overline{b_{i_l}} \cap \overline{b_{i_j}}|$. Furthermore, the intervals $\overline{b_{i_j}}$ and $\overline{b_{i_l}}$ do not conflict and hence $|\overline{b_{i_l}} \cap \overline{b_{i_j}}| < \min\{t_{b_{i_l}}, t_{b_{i_j}}\} \leq t_{b_{i_l}} = \min\{t_{b_{i_l}}, t_u\}$. It follows that $\overline{u}$ and $\overline{b_{i_l}}$ do not conflict and hence u and b_{i_l} are not adjacent in G.

The case $j = k$ follows verbatim if we let $t = b_{i_{k+1}}$. In this case it is more 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 $S(e_j)$, $(u, b_{i_l}) \notin E(G)$. ∎

Lemma 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 $I_P = B_P \cup S(e_0) \cup S(e_1) \ldots \cup S(e_k)$ in G. In the example of Figure 3 there are two paths of total weight 3, $P_1 = ((s,t))$ and $P_2 = ((s,a),(a,c),(c,t))$. Their corresponding independent sets are $I_{P_1} = \{b, d, f\}$ and $I_{P_2} = \{a, c, f\}$. Thus, selecting edge $e = (b_i, b_j)$ $1 \le j \le k$ to be included in a path P from s to t in $H(G)$ is equivalent to selecting $S(e) \cup \{b_j\}$ 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 1 *Given a tolerance graph $G = (V, E)$ and a regular tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$ of G. There is an $\mathcal{O}(|B|^2 \log(|U| + 1))$ 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_0} and denote t by $b_{i_{k+1}}$. We claim that the largest independent set in G is

$$S = \bigcup_{i=0}^{k} S(e_i) \cup B_P.$$

Lemma 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 $S_B = \{s, b_{i_1}, b_{i_2}, \ldots, b_{i_k}, t\}$ be an ordered set consisting of s, t, and all bounded vertices in S ordered by the right end points 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, $e_j = (b_{i_j}, b_{i_{j+1}})$, joining consecutive vertices in S_B form a directed path P from s to t in the right end point 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 $0 \le j \le k$. In this partition a vertex $u \in S_U$ belongs to $S(e_j)$ if $r(b_{i_j}) < r(u) < r(b_{i_{j+1}})$.

The construction of the unweighted right end point orientation of $H(G)$ can be done in $\mathcal{O}(|B^2|)$. The weight function w can be computed in time

$\mathcal{O}(|B|^2 \log(|U|+1))$, assuming, as we did, that U is sorted. 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 $H(G)$, [Go80]. The size of $H(G)$ is $\mathcal{O}(|B|^2 + \log|w|) = \mathcal{O}(|B|^2 + \log|U|)$, where $|w|$ is the largest weight in $H(G)$. All these steps combined yield a total time complexity of $\mathcal{O}(|B|^2 + |B|^2 \log|U|) = \mathcal{O}(|B|^2 \log(|U|+1))$. ∎

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.

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

Corollary 2 *Given a tolerance graph $G = (V, E)$ and a regular tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$ of G. There is an $\mathcal{O}(|V|^2)$ algorithm to find the clique cover number of G.*

3 Coloring

In this section we show how to color a bounded tolerance graph and how to find the chromatic number of a general tolerance graph.

The chromatic number of a bounded tolerance graph equals the clique cover number of its complement. Since the complement of a bounded tolerance graph is a comparability graph, we first examine an algorithm to find the clique cover number of a comparability graph [Go80]. Let G be a comparability graph and let H be the right endpoint transitive orientation of G. We transform H into a transportation network N by adding a super source S joined to all the sources in H and a super sink T joined to all the sinks in H. Each vertex is assigned a lower capacity 1. The value of a minimum flow of N equals the clique cover number of G. There are many known network flow algorithms with varying time complexities. The choice as to which one to choose depends upon the density of the digraph and space considerations. For simplicity we state here a worst case upper bound for the complexity of the network flow problem as $\mathcal{O}(|V|^3)$. For a discussion of the different methods and their associated time complexity see [Tar83]. Thus, the complexity of finding the chromatic number of a bounded tolerance graph is $\mathcal{O}(|V|^3)$. We note that this algorithm can be extended to find an optimal coloring of a bounded tolerance graph.

Theorem 3 *There is an $\mathcal{O}(|V|^3)$ algorithm to color a bounded tolerance graph with an optimal number of colors. (We assume that the input consists of a bounded tolerance graph together with a regular tolerance representation for it.)*

Proof: Let G be the complement of the given bounded tolerance graph. We first construct a network N as above and find a minimum flow F for N. We then use the transitivity of H to find a minimum flow f that pushes exactly one unit of flow through each vertex in H, except for its sources and sinks. The procedure to find f is based on a breadth first search. Suppose that following an edge (v, w) we discover that w has already been visited by our breadth first search. If no such edge exists then $f := F$. Moreover suppose that at least one unit of flow is pushed along a path from w to a sink t of H (there is at least one sink with this property). Then we can redirect the flow along (v, w) to the edge (v, t), whose existence is guaranteed by the transitivity of H.

The flow f induces a partition of the vertices into $\chi(G)$ vertex disjoint paths from a source of H to a sink of H, each with unit flow. (These path may have a sink or a source in common.) The internal vertices of each path form a clique in G and an independent set in its complement. A source s (resp. a sink t) of H can be now assigned to an arbitrary clique corresponding to a path starting (resp. terminating) with s (resp. t). The result is a partition of the vertex set of G into $\chi(G)$ sets whose induces subgraphs are cliques. This partition is in fact a partition of the input graph G^c into $\chi(G)$ independent sets, which in turn is an optimal coloring of the given bounded tolerance graph. The flow f can be found in linear time, given any minimum flow on N. We note that an implementation of the algorithm can incorporate this last stage into the search for a minimum flow. ∎

Theorem 4 *Given a tolerance graph $G = (V, E)$ and a regular tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$ of G. There is an $\mathcal{O}(q|V|^3)$ algorithm to find the chromatic number of G, where q is the number of unbounded intervals in the tolerance representation of G.*

Since tolerance graphs are perfect, Theorem 4 follows from Theorem 5 below.

Theorem 5 *Given a tolerance graph $G = (V, E)$ and a regular tolerance representation $\langle \mathcal{I}, \mathcal{T} \rangle$ of G. There is an $\mathcal{O}(q|V|^3)$ 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 $\{k_u + 1 \quad | u \in U\}$.

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 $\mathcal{O}(|V|^3)$, the total time complexity of our algorithm is $\mathcal{O}(q|V|^3)$. ∎

4 References

[GM82] Golumbic, M., and Monma, C., A Generalization of Interval Graphs with Tolerances, *Proc. 13th SouthEastern Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium*, **35**, Utilitas Math., Winnipeg, 1982, 321-331.

[GLS81] M. Grotshcel, L. Lovasz, and A. Schrijver, The Ellipsoid Method and its Consequences in Combinatorial Optimization, *Combinatorica*, **1**, (1981), 169-197.

[GMT84] Golumbic, M., Monma, C., and Trotter, W., Tolerance Graphs, *Disc. Appl. Math.*, **9**, 1984, 157-170.

[Go80] M. C. Golumbic, "Algorithmic Graph Theory And Perfect Graphs", Academic Press, New York, 1980.

[Tar83] R. E. Tarjan, "Data structures and network algorithms", Siam, Philadephia, 1983.