Coloring Algorithms for Tolerance Graphs: Reasoning and Scheduling with Interval Constraints

Martin Charles Golumbic[1] and Assaf Siani[2]

[1] Dept. of Computer Science, University of Haifa, Haifa, Israel
golumbic@cs.haifa.ac.il
[2] Dept. of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
siani@cs.biu.ac.il

Abstract. Interval relations play a significant role in constraint-based temporal reasoning, resource allocation and scheduling problems. For example, the intervals may represent events in time which may conflict or may be compatible, or they may represent tasks to be performed according to a timetable which must be assigned distinct resources like processors or people. In previous work [G93, GS93, G98], we explored the interaction between the interval algebras studied in artificial intelligence and the interval graphs and orders studied in combinatorial mathematics, extending results in both disciplines.
In this paper, we investigate algorithmic problems on tolerance graphs, a family which generalizes interval graphs, and which therefore have broader application. Tolerance graph models can represent qualitative and quantitative relations in situations where the intervals can tolerate a certain degree of overlap without being in conflict. We present a coloring algorithm for a tolerance graph on n vertices whose running time is $O(n^2)$, given the tolerance representation, thus improving previously known results. The coloring problem on intervals has direct application to resource allocation and scheduling temporal processes.

Keywords and Topics: AI, OR applications, reasoning, coloring tolerance graphs

1 Introduction

Graph $G=(V,E)$ is a *tolerance graph* if each vertex $v \in V$ can be assigned an interval on the real line that represents it, denoted I_v, and a real number $t_v > 0$ referred to as its tolerance, such that for each pair of adjacent vertices, $uv \in E$ if and only if $|I_u \cap I_v| \geq \min\{t_u, t_v\}$. The intervals represented by the vertices and the tolerances assigned to the intervals form the *tolerance representation* of graph G [see Figure 1]. If the graph has a tolerance representation for which the tolerance t_v of the interval I_v representing each vertex $v \in V$ is smaller than or equal to the interval's length, i.e., $t_v \leq |I_v|$, then that graph is called a *bounded tolerance graph* and its representation is a *bounded representation.* The family of tolerance graphs was first introduced by

J. Calmet et al. (Eds.): AISC-Calculemus 2002, LNAI 2385, pp. 196–207, 2002.

Golumbic and Monma [GM82]. Their motivation was the need to solve scheduling problems in which resources that normally we would use exclusively, like rooms, vehicles, etc. can tolerate some sharing. Since then, properties of this model and quite a number of variations of it have appeared, and are the topic of a forthcoming monograph [GT02]. The tolerance graphs are a generalization of *probe graphs*, a graph family used for Genome research, which is in itself a generalization of the well-known family of *interval graphs*. Tolerance graphs are a sub-family of the Perfect Graphs [GMT84], and shares the latter's variety of mathematical and algorithmic properties [G80]. We are particularly interested in finding algorithms to calculate specific problems concerning these graphs, like graph coloring or maximum clique cover, because of their application to constraint-based temporal reasoning, resource allocation and scheduling problems.

As part of our algorithmic research, we present a coloring algorithm for a tolerance graph on n vertices whose running time is $O(n^2)$, given the interval tolerance representation. This is an improvement over previously known results of [NM92]. Another problem we deal with in this research is finding all the maximal cliques in a tolerance graph. Let k be the number of a maximal cliques of a graph G. Since it is known that k is not polynomially bounded by n for tolerance graphs, we present an algorithm for iteratively generating all maximal cliques in a graph, which uses as a subroutine an efficient algorithm for finding all the maximal cliques in a bounded tolerance graph.

2 Basic Definitions and Background

Let $G = (V, E)$ be an undirected graph. We call $N(v)=\{w \mid (v,w)\in E\}$ the *(open) neighborhood* of vertex v, and we call $N[v] = N(v)\cup\{v\}$ the *closed neighborhood* of v. The pair $(v,w)\in E$ is an *edge* and we say that w is adjacent to v, and v and w are *endpoints* of the edge (v,w). When it is clear, we will often drop the parenthesis and the comma when denoting an edge. Thus $xy\in E$ and $(x,y)\in E$ will have the same meaning.

We define the *complement* of G to be the graph $\overline{G} = (V, \overline{E})$, where $\overline{E}=\{(x,y)\in V \times V \mid x \neq y$ and $xy\notin E\}$. Given a subset $A\subseteq V$ of the vertices, we define the *subgraph induced* by A to be $G_A=(A,E_A)$, where $E_A = \{xy\in E \mid x\in A$ and $y\in A\}$.

A graph in which every pair of distinct vertices are adjacent is called a *complete graph*. K_n denotes the complete graph on n vertices. A subset $A\subseteq V$ of r vertices is an *r-clique* if it induces a complete subgraph, i.e., if G_A isomorphic to K_r. A single vertex is a 1-clique. A clique A is *maximal* if there is no clique of G, which properly contains A as a subset. A clique is *maximum* if there is no clique of G of larger cardinality. We denote by $\omega(G)$ the number of vertices in a maximum clique of G; it is called the *clique number* of G.

A *clique cover* of size k is a partition of the vertices $V=A_1+A_2+...+A_k$ such that each A_i is a clique. We denote by $k(G)$ the size of the smallest possible clique cover of G and is called the *clique cover number* of G.

A *stable set* (or *independent set*) is a subset X of vertices no two of which are adjacent. We denote $\alpha(G)$ to be the number of vertices in a stable set of maximum cardinality; it is called the *stability number* of G.

A *proper c-coloring* is a partition of the vertices $V=X_1+X_2+...+X_c$ such that each X_i is a stable set. In such a case, the members of X_i are "painted" with the color i and adjacent vertices will receive different colors. We denote by $\chi(G)$ the smallest possible c for which there exists a proper c-coloring of G; it is called the *chromatic number* of G.

A graph G is a *perfect graph* if G has the property that for every induced subgraph G_A of G, its chromatic number equals its clique number (i.e., $\chi(G_A) = \alpha(G_A)$). Due to a theorem of Lovász [L72] perfect graphs may be defined alternatively, G has the property that for every induced subgraph G_A of G, its stability number equals its clique cover number (i.e., $\alpha(G_A) = k(G_A)$). Perfect graphs are important for their applications and because certain decision problems that are NP-complete in general have polynomial-time algorithms when the graphs under consideration are perfect. Tolerance graphs were shown to be perfect in [GMT84].

Let F be a family of nonempty sets. The *intersection* graph of F is obtained by representing each set in F by a vertex and connecting two vertices by an edge if and only if their corresponding sets intersect. The intersection graph of a family of intervals on a linearly ordered set is called an *interval graph*, see Figure 2. The interval graph's real world applications vary from classic computer science problems such as scheduling or storage, to a chemical, biological and even archeological problems (refer to [G80] for details).

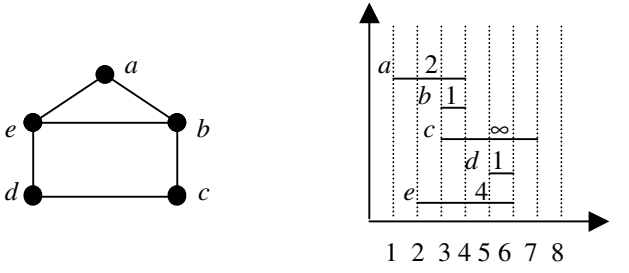

Fig. 1. A tolerance graph and a tolerance representation for it.

It can be easily shown that an interval graph is the special case of a tolerance graph where the tolerance t_v of interval I_v for all $v \in V$ equals some very small constant $\varepsilon > 0$.

Being an interval graph is a hereditary property, i.e., an induced subgraph of an interval graph is an interval graph. Another hereditary property of interval graph is that every simple cycle of length strictly greater than 3 possesses a chord. Graphs that satisfy this property are called *chordal graphs*. The graph in Figure 2 is chordal, but the graph in Figure 1 is not chordal because it contains a chordless 4-cycle. Therefore, it is not an interval graph. It is, however, a tolerance graph, which can be seen by the tolerance representation in Figure 1.

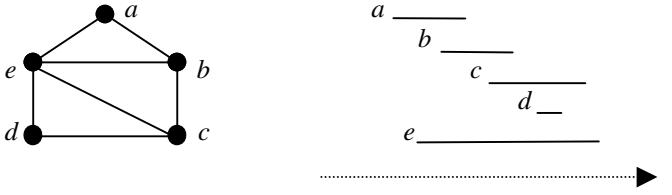

Fig. 2. An interval graph and an interval representation for it.

A graph said to be *weakly chordal* if it contains no induced subgraph isomorphic to C_n or to the complement of C_n for $n \geq 5$. Golumbic, Monma and Trotter [GMT84] showed that tolerance graphs are weakly chordal. Hayward [H85] showed that weakly chordal graphs are perfect, and gave some polynomial time algorithms for problems relating with the chromatic number and the stability number for weakly chordal graphs. Raghunathan [R89] improve these results and provide $O(|E||V|^2)$ algorithms to find a maximum clique and a minimum coloring of a weakly chordal graph. He also obtained an $O(|V|^4)$ algorithm to find a maximum independent set and a minimum clique cover of such a graph**,** and $O(|V|^5)$ algorithms for weighted versions of these problems

A graph G is a *comparability graph* if each edge in E can be assigned a direction so that the resulting oriented graph (V,F) satisfies the following condition: $ij \in F$ and $jk \in F$ implies that $ik \in F$ for all $i,j,k \in V(G)$. Such an orientation is called a *transitive orientation* of G and when such an orientation exists we say that G is *transitively orientable*. A *co-comparability graph* is a graph whose complement is a comparability graph.

Theorem 2.1 **[GM82].** Bounded tolerance graphs are co-comparability graphs.

One proof of this theorem is given in [BFIL95] and uses parallelogram graphs, which provide another useful way to think about bounded tolerance graphs. A graph G is a *parallelogram graph* if we can fix two parallel lines L_1 and L_2 with L_1

above L_2, and for each vertex $i \in V(G)$ we can assign a parallelogram P_i with parallel sides along L_1 and L_2 so that G is the intersection graph of $\{P_i \mid i \in V(G)\}$.

Lemma 2.2 **[BFIL95].** A graph is a bounded tolerance graph if and only if it is a parallelogram graph.

Finally, it is sometimes convenient to assume, without loss of generality as shown in [GMT84], that any tolerance graph has a *regular* tolerance representation which satisfies the following addition properties:

1. Any tolerance larger than the length of its corresponding interval is set to infinity.
2. All tolerances are distinct (except for those set to infinity).
3. No two different intervals share an endpoint.

3 Coloring Tolerance Graphs

The problem of coloring a graph has many applications. Basically, a graph can represent a collection of objects: the vertices, which relate to each other through the edges. It is very common to use this structure to model objects that consume resources, whereby the edges symbolize some restriction on two objects that cannot consume the same resource simultaneously. The act of coloring a graph, i.e., labeling the vertices of the graph such that each of any two adjacent vertices receives a different label, is actually the act of allocating a resource (the label) to an object (the vertex). Minimal coloring in that sense is therefore utilizing minimal resources without violating any restriction. For example, consider a model in which lectures are represented by vertices, any two lectures which take place simultaneously cause an edge. A minimal coloring of the graph may signify a minimal allocation of classrooms for this set of lectures. The graph coloring problem is NP-complete for general graphs, however, efficient polynomial time algorithms are known for many classes of perfect graphs.

Narasimhan and Manber [NM92] suggested a simple method for computing the chromatic number $\chi(G)$ of a tolerance graph. This method is based on the facts that (1) the bounded intervals in the tolerance model form a co-comparability graph (Theorem 2.1) and that there exists a known algorithm for computing the chromatic number of co-comparability graphs. Their algorithm results in $O(qn^3)$ time, where q is the number of unbounded intervals in the tolerance model, n is the number of vertices and $O(n^3)$ is the time required for computing the chromatic number of a co-comparability graph. However, as was shown subsequently, bounded tolerance graphs are parallelogram graphs (Lemma 2.2), and Felsner et al. [FMW97] suggested an algorithm for computing the chromatic number of trapezoid graphs (which contain parallelogram graphs) in optimal $O(n\log n)$ time. Hence, we can use this algorithm for computing the chromatic number of the bounded intervals, and using the same idea of [NM92] obtain the chromatic number $\chi(G)$ of a tolerance graph in $O(qn\log n)$ time. However, their method does not give a coloring for the graph.

We took this problem a step forward to produce a coloring of the graph. Our algorithm has complexity $O(n^2)$ for general tolerance. In all these algorithms, it is assumed that a tolerance representation for the graph is given as input, which is typically the case in most applications. The problem of recognizing tolerance graphs is a longstanding open question.

3.1 Definitions, Terminology, and Lemmas

Let $G=(V,E)$ be a tolerance graph with a regular tolerance representation $<I,t>$. We define the subset of bounded vertices of V as $V_B=\{v\in V \mid t_v \le |I_v|\}$, and the unbounded vertex subset of V as $V_U=\{v\in V \mid t_v>|I_v|\}$. Similarly, the induced bounded subgraph is $G_B=(V_B,E)$, and the induced unbounded subgraph is $G_U=(V_U,E)$.

***Definition* 3.1.** An unbounded tolerance vertex $v\in V$ is labeled an *inevitable unbounded* vertex in G (for a certain tolerance representation), if the following holds:
(1) v is not an isolated vertex.
(2) $t_v > |I_v|$.
(3) Setting v's tolerance to its length (i.e. $t_v=|I_v|$) creates a new edge in the representation (i.e., the representation is no longer a tolerance representation for G).

***Definition* 3.2.** An *inevitable unbounded* tolerance representation for G is a tolerance representation, where every unbounded tolerance vertex is an inevitable unbounded vertex.

***Lemma* 3.3.** Every tolerance representation can be transformed into an inevitable unbounded representation in $O(n^2)$ time.

Proof. We will scan the representation from left to right. At any point during the algorithm we are aware of the *active* intervals, i.e., the intervals whose left endpoint has been scanned but whose right endpoint hasn't. Whenever a left endpoint of unbounded interval I is scanned, we check whether there is, among the active intervals, an interval which contains the interval I (i.e., its right endpoint is larger than $r(I)$ and the interval's tolerance is larger than $|I|$). If such an interval exists, then reducing I's tolerance to its length will cause a new edge in the representation between I and that interval. If no such interval exists, reduce I to its length. Otherwise, check whether there is some interval that forms an edge with I. If no such interval exists, then I is a isolated interval. In this case, make it the rightmost interval and make its tolerance equal to its length. If there is an interval adjacent to I, then I is left with unbounded tolerance. The scan is continued with the modified representation. □

***Definition* 3.4.** Let $v\in V$ be an inevitable unbounded vertex in G (for some representation $<I,t>$), then there is (at least) one bounded vertex $u\in V$ for which $uv\notin E$ and $I_v\subset I_u$ ($t_u > |I_v|$). The vertex u is called a *hovering vertex* for v and I_u is called a *hovering interval* for I_v.

***Definition* 3.5.** We define the *hovering vertex set* (abbreviated *HVS*) for some inevitable unbounded vertex $v \in V$ to be the set of all hovering vertices of v, i.e., $HVS(v)=\{u \mid u$ is a hovering vertex for $v\}$.

***Lemma* 3.6.** Let $v \in V$ be an inevitable unbounded vertex in G (for some representation $\langle I,t \rangle$). The set *HVS* always contains (at least) one bounded interval.

Proof. Since v is an inevitable unbounded vertex, hence there is some vertex $w_1 \in V$ where $I_v \subset I_{w_1}$. Supposing $t_{w_1} = \infty$, then w_1 is an inevitable unbounded vertex and there is some vertex $w_2 \in V$ where $I_{w_1} \subset I_{w_2}$. By induction this continues until we get $I_v \subset I_{w_1} \subset I_{w_2} \subset \ldots \subset I_{w_k}$ where w_k is a bounded vertex and $|I_{w_k}| > |I_{w_{k-1}}| > \ldots > |I_{w_1}| > |I_v|$, hence $w_k v \notin E$ and therefore $I_{w_k} \in HVS(v)$. □

***Lemma* 3.7.** Let $v \in V$ be an inevitable unbounded vertex in G (for a representation $\langle I,t \rangle$). We can find some bounded interval I_w, $I_w \in HVS(v)$, to be a representative interval for the set $HVS(v)$ in $O(n^2)$ time.

Proof. We will scan the representation from left to right. At any point during the algorithm we are aware of the *active* intervals, i.e., the intervals whose left endpoint has been scanned but their right endpoint hasn't. We will search amongst the active intervals for some bounded interval I_w that contains I_v, which is not adjacent to I_v (we know that such an interval exists from Lemma 3.6). We choose this interval to be a representative for $HVS(v)$. □

3.2 Algorithm for Coloring Tolerance Graphs

On one hand, if every maximum clique in G contains an unbounded vertex, then we can color G_B as a co-comparability graph (or parallelogram graph), give the set of vertices V_U a different color, and we are done. If, on the other hand, there exists a maximum clique in G which does not contain an unbounded vertex, then we should do the following: (1) color G_B as a co-comparability graph or parallelogram graph, (2) for every vertex $v \in V_U$, insert v into the color-set of a representative vertex $w \in HVS(v)$. This is justified since $vw \notin E$ and so $N(v) \subseteq N(w)$, hence no neighbor of v is present in the color-set of w, and thus this is a proper coloring of G .

Thus, a preliminary algorithm could require that we find all maximum cliques in the graph and check whether they contain an unbounded tolerance vertex or not. However, for every inevitable unbounded tolerance vertex $v \in V$, $HVS(v)$ contains some bounded vertex which would be colored when coloring G_B, and thus can determine a color for v. It follows that we do not need the maximum cliques after the transformation of Lemma 3.3 has been applied. Knowing the above, we conclude the following algorithm:

***Algorithm* 3.1.**

Color G_B as a co-comparability graph (or parallelogram graph).
For every inevitable unbounded vertex $v \in V_U$,
insert v into the color-set of some representative vertex $w \in HVS(v)$.

Correctness

Clearly, the coloring for G_B is proper. For every unbounded interval I_i we have some bounded interval I_j such that $I_i \subset I_j$ but $v_i v_j \notin E$, which implies that $t_j > |I_i|$. Clearly, every interval I_k which forms an edge with I_i has to be bounded, and thus we get $|I_j \cap I_k| \geq |I_k \cap I_i| \geq \min\{t_k, t_i\} = t_k$, implying that $v_j v_k \in E$, and that v_k is not in the color set of v_j. Therefore, if we insert v_i into the color set of v_j, there will be no vertex in that color set to form an edge with v_i, and thus the coloring is proper.

Complexity

The algorithm consists of three stages: the first stage is a transformation of the tolerance representation into an inevitable unbounded one. The transformation takes $O(n^2)$ time (because we need to check each intersecting interval of every unbounded tolerance interval). The second stage is finding a representative interval for the *HVS* set of every unbounded tolerance interval which also takes $O(n^2)$ time, for the reasons mentioned above. Finally, the third stage is the coloring itself, which is also divided into three parts: the first part is a transformation of the tolerance representation (of the bounded intervals) into a parallelogram representation, which takes linear time. The second part is the coloring of the parallelogram representation, which is done in $O(n \log n)$ time [FMW97]. The third part is the coloring of each unbounded tolerance interval; this takes linear time, for the procedure is simply labeling the unbounded intervals with the color of their *HVS*'s set representative. The total time is therefore $O(n^2)$.

We conclude by noting that since tolerance graphs are perfect, $\chi(G) = \omega(G)$, and our coloring algorithm can easily find a maximum clique at the same time.

4 All Maximal Cliques of Tolerance Graphs

A maximal clique in a graph G is a complete set X of vertices such that no superset of X is a clique of G. Finding all maximal cliques is sometimes necessary to find a solution for other problems. For example, cut vertices, bridges and vertex-disjoint paths can all be determined easily once the maximal cliques are known. In this section, we will present a method for computing all maximal cliques of tolerance graphs, given the tolerance model of the graph. For solving the all-maximal cliques problem we used the same method as with the coloring problem in section 3, i.e., we solve the problem for the induced graph containing only the bounded vertices and

later we deal with the unbounded vertices. Unfortunately, the number of maximal cliques for the induced bounded intervals may be exponential. However, for interval graphs, as with all chordal graphs, the number of maximal cliques is at most $|V|$. Our algorithm especially suits subfamilies of tolerance graphs where computing all maximal cliques for the induced subgraph containing only bounded vertices is done in polynomial time.

4.1 All Maximal Cliques of a Tolerance Graph

Let $G=(V,E)$ be a tolerance graph and let $<I,t>$ be a regular tolerance representation for G. Let V_B be the set of all bounded vertices in V in this representation, and let V_U be the set of all unbounded vertices in V. We will use K_X to denote the set of all maximal cliques of any set of vertices X, $X \subseteq V$. We define the set K_G to be the set of all maximal cliques in G, and let K_B denote the set of all maximal cliques in G_B. For every $u \in V_U$, let $K_{N(u)}$ denote the set of all maximal cliques in the graph $G_{N(u)}$, which is the subgraph induced from G by the neighbors of the vertex u. Clearly, for every $u \in V_U$ and $Y \in K_{N(u)}$, $Y \cup \{u\}$ is a maximal clique in G. Finally, let $K_{N[u]}$ denote the set of all maximal cliques in $G_{N[u]}$, i.e., $K_{N[u]}=\{Y \cup \{u\} \mid Y \in K_{N(u)}\}$.

***Theorem* 4.1.** $K_G = \left\{ K_B \setminus \left\{ \bigcup_{u \in U} K_{N(u)} \right\} \right\} \cup \left\{ \bigcup_{u \in U} K_{N[u]} \right\}$.

Proof. ("$\subseteq$" part.) Let Y be a maximal clique in G, i.e. $Y \in K_G$. If Y does not contain any unbounded vertex, then $Y \in K_B$ and is not in any $K_{N[u]}$ where $u \in U$. Otherwise, Y contains an unbounded vertex $u \in V_U$ (Y may contain only one such vertex since Y is a clique). $Y \setminus \{u\}$ is a maximal clique in $G_{N(u)}$ and $Y \in K_{N[u]}$.

("$\supseteq$" part.) Assume $Y \in \left\{ K_B \setminus \left\{ \bigcup_{u \in U} K_{N(u)} \right\} \right\} \cup \left\{ \bigcup_{u \in U} K_{N[u]} \right\}$.

Suppose to the contrary that there is some $Y' \in K_G$ such that $Y \subset Y'$. If Y' does not contain an unbounded vertex, then $Y \in K_B$ - a contradiction. Hence, Y' contains an unbounded vertex $u \in U$. But in that case $Y \setminus \{u\} \in K_{N(u)}$, and $Y' \in K_{N[u]}$, a contradiction. □

Now, if G_B belongs to a known hereditary graph family, for which we have an algorithm for computing all maximal cliques of that family, we can use that algorithm to compute K_B and $K_{N[u]}$ for every $u \in U$, since $G_{N(u)}$ is also a graph of the same family because all the vertices in $N(u)$ are bounded.

Recall from section 3, that we defined the terms *inevitable unbounded representation* and the *HVS* set of an inevitable unbounded interval/vertex. Also recall that each neighbor vertex of some unbounded vertex $u \in V_U$ is a neighbor vertex of every vertex $v \in HVS(u)$. Algorithm 4.1 is based on the following lemma.

***Lemma* 4.2.** Let $v \in B$ be a hovering vertex of u, i.e., $v \in HVS(u)$. For every $Y \in K_{N(u)}$, there is some $Y' \in K_{N[v]}$ for which $Y \subseteq Y'$.

***Proof*.** Clearly, for every vertex $w \in N(u)$, $w \in N(v)$. Hence, for every maximal clique $Y \in K_{N(u)}$, either Y is maximal clique of $K_{N[v]}$ or there is some other maximal clique $Y' \in K_{N[v]}$ such that $Y \subset Y'$. □

Let $v \in B$ be a hovering vertex of u. By lemma 4.2, we can compute all the maximal cliques of $G_{N(u)}$ by taking all the maximal cliques in G containing v and deleting any vertex which is not adjacent to u, as we do next in algorithm 4.1.

Algorithm 4.1

Let $G=(V,E)$ be a tolerance graph and let $\langle I,t \rangle$ be an inevitable unbounded representation for G.

(1) Compute K_B, let $K_B^{Computed} \leftarrow K_B$.

(2) For any unbounded vertex $u \in V_U$,

(2.1) Let $w \in V_B$ be a hovering vertex of u, i.e., w is some representative of $HVS(u)$.

(2.2) For any maximal clique $Y \in K_B$ containing w, compute the set $Y' = \{Y \setminus \{w\}\} \cap N(u)$.

(2.3) If $Y' \neq \phi$ then $K_U^{Computed} \leftarrow K_U^{Computed} \cup \{Y' \cup \{u\}\}$.

***Theorem* 4.3.** $K_G = K_B^{Computed} \cup K_U^{Computed}$

***Proof*.** . ("$\subseteq$" part) Let $Y \in K_G$ be a maximal clique in G. If Y does not contain any unbounded vertex, then $Y \in K_B^{Computed}$. Otherwise, Y contains some unbounded vertex $u \in U$ (Y may contain only one such vertex since Y is a clique). For each $v \in HVS(u)$, there exists some maximal clique Y' such that $Y \setminus \{u\} \subset Y'$, hence by the algorithm $Y \in K_U^{Computed}$.

("$\supseteq$" part) Assume that $Y \in K_B^{Computed} \cup K_U^{Computed}$. Suppose to the contrary that there is some $Y' \in K_G$ where $Y \subset Y'$. If Y' contains only bounded tolerance vertices it is a contradiction for $K_B^{Computed}$ being the set of all maximal cliques of G_B. Hence, Y' must contain some $u \in U$. The set $Y \setminus \{u\}$ is a set of bounded vertices, hence there is a clique set $Y'' \in K_B^{Computed}$ such that $Y \setminus \{u\} \subset Y''$, and Y'' contains $v \in HVS(u)$ (v is the reason for the proper inclusion of $Y \setminus \{u\}$ in Y''). But this is a contradiction since $Y' \setminus \{v\} \cap N(u) \in K_U^{Computed}$. □

4.2 The Number of All Maximal Cliques in a Tolerance Graph

As previously mentioned, the number of all maximal cliques of tolerance graph may be exponential. Bounded tolerance graphs, a subfamily of tolerance graphs, contains the permutation graph family. Consider the permutation diagram in Figure 3, which has $2n$ lines, such that each line intersects with all the other lines except for one line. We can choose one line of any two parallel lines, and the result would be a maximal clique in the permutation graph. It is easy to see that we have 2^n maximal cliques in this graph. Hence, the number of maximal cliques of a tolerance graph may be exponential.

As a result, the algorithm presented for computing all maximal cliques of tolerance graphs is one, which iteratively generates them. In the case where the bounded vertices of the graph form a subgraph of a structured family whose maximal cliques may be computed in polynomial time using a known algorithm, as with the family of probe graphs, the method then becomes polynomially efficient.

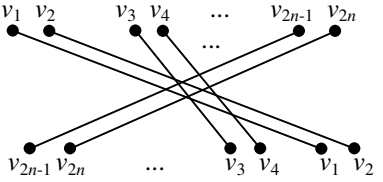

Fig. 3. Permutation diagram whose permutation graph has $2n$ vertices and exactly 2^n maximal cliques.

References

[BFIL95] K. Bogart, P. Fishburn, G. Isaak and P. Langley, Proper and unit tolerance graphs, *Discrete Applied Math.* 60 (1995) 37-51.

[F98] S. Felsner, Tolerance graphs and orders, *J. of Graph Theory* 28 (1998) 129-140.

[FMW97] S. Felsner, R. Müller, L. Wernisch, Trapezoid graphs and generalizations, geometry and algorithms, *Discrete Applied Math.* 74 (1997) 13-32.

[G80] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[G93] M.C. Golumbic, Reasoning about time, Invited talk, AISMC-1 Karlsruhe, Germany, August 3-6, 1992, abstract in *LNCS* 737 (1993) p. 276.

[G98] M.C. Golumbic, Reasoning about time, in *"Mathematical Aspects of Artificial Intelligence"*, F. Hoffman, ed., American Math. Society, *Proc. Symposia in Applied Math.*, vol. 55, 1998, pp. 19-53.

[GM82] M.C. Golumbic and C.L. Monma, A generalization of interval graphs with tolerances, in: Proceedings 13th Southeastern Conference on Combinatorics, Graph Theory and Computing, *Congressus Numerantium* 35 (1982) 321-331.

[GMT84] M.C. Golumbic, C.L. Monma and W.T. Trotter Jr., Tolerance graphs, *Discrete Applied Math.* 9 (1984) 157-170.

[GS93] M.C. Golumbic and R. Shamir, Complexity and algorithms for reasoning about time: a graph-theoretic approach, *J. Assoc. Comput. Mach.* 40 (1993), 1108-1133.

[GT02] M.C. Golumbic and Ann N. Trenk, *Tolerance Graphs*, monograph in preparation.

[H85] R. Hayward, Weakly triangulated graphs, *J. Combin. Theo. Ser. B* 39 (1985) 200-209.

[HHM90] R. Hayward, C. Hoàng, and F. Maffray, Optimizing weakly triangulated graphs, *Graphs and Combinatorics* 6 (1990) 33-35. Erratum to *ibid*, 5:339-349.

[L72] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Math.* 2 (1972), 253-267.

[NM92] G. Narasimhan and R. Manber, Stability number and chromatic number of tolerance graphs. *Discrete Applied Math.* 36 (1992) 47-56.

[R89] A. Raghunathan, Algorithms for weakly triangulated graphs, Univ. of Calif. Berkeley, Technical Report CSD-89-503 (April 1989).