RWTH Aachen

Department of Computer Science
Technical Report

A New Intersection Model and Improved Algorithms for Tolerance Graphs

George B. Mertzios
Ignasi Sau
Shmuel Zaks

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2009-05

RWTH Aachen · Department of Computer Science · March 2009

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

A New Intersection Model and Improved Algorithms for Tolerance Graphs

George B. Mertzios*, Ignasi Sau**, and Shmuel Zaks***

Abstract. Tolerance graphs model interval relations in such a way that intervals can tolerate a certain degree of overlap without being in conflict. This class of graphs, which generalizes in a natural way both interval and permutation graphs, has attracted many research efforts since their introduction in [10], as it finds many important applications in constraint-based temporal reasoning, resource allocation, and scheduling problems, among others. In this article we propose the first non-trivial intersection model for general tolerance graphs, given by three-dimensional parallelepipeds, which extends the widely known intersection model of parallelograms in the plane that characterizes the class of bounded tolerance graphs. Apart from being important on its own, this new representation also enables us to improve the time complexity of three problems on tolerance graphs. Namely, we present optimal $\mathcal{O}(n \log n)$ algorithms for computing a minimum coloring and a maximum clique, and an $\mathcal{O}(n^2)$ algorithm for computing a maximum weight independent set in a tolerance graph with n vertices, thus improving the best known running times $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ for these problems, respectively.

Keywords: Tolerance graphs, parallelogram graphs, intersection model, minimum coloring, maximum clique, maximum weight independent set.

1 Introduction

A graph $G = (V, E)$ on n vertices is a *tolerance graph* if there is a set $I = \{I_i \mid i = 1, \ldots, n\}$ of closed intervals on the real line and a set $T = \{t_i \mid i = 1, \ldots, n\}$ of positive real numbers, called *tolerances*, such that for any two vertices $v_i, v_j \in V$, $v_i v_j \in E$ if and only if $|I_i \cap I_j| \geq \min\{t_i, t_j\}$, where $|I|$ denotes the length of the interval I. These sets of intervals and tolerances form a *tolerance representation* of G. If G has a tolerance representation such that $t_i \leq |I_i|$ for $i = 1, \ldots, n$, then G is called a *bounded tolerance graph* and its representation is a *bounded tolerance representation*.

Tolerance graphs were introduced in [10], mainly motivated by the need to solve scheduling problems in which resources that would be normally used exclusively, like rooms or vehicles, can tolerate some sharing among users. Since then, tolerance graphs have been widely studied in the literature [1,2,5,11,12,15,19,22], as they naturally generalize both interval graphs (when all tolerances are equal) and permutation graphs (when $|I_i| = t_i$ for $i = 1, \ldots, n$) [10]. For more details, see [13].

Notation. All the graphs considered in this paper are finite, simple, and undirected. Given a graph $G = (V, E)$, we denote by n the cardinality of V. An edge between vertices u and v is denoted by uv, and in this case vertices u and v are said to be *adjacent*. $\overline{G}$ denotes the *complement* of G, i.e. $\overline{G} = (V, \overline{E})$, where $uv \in \overline{E}$ if and only if $uv \notin E$. Given a subset of vertices $S \subseteq V$, the graph $G[S]$ denotes the graph *induced* by the vertices in S, i.e. $G[S] = (S, F)$, where for any two vertices $u, v \in S$, $uv \in F$ if and only if $uv \in E$. A subset $S \subseteq V$ is an *independent set* in G if the graph $G[S]$ has no edges. For a subset $K \subseteq V$, the induced subgraph $G[K]$ is a *complete subgraph* of G, or a *clique*, if each two of its vertices are adjacent (equivalently, K is

* Department of Computer Science, RWTH Aachen University, Germany. Email: `mertzios@cs.rwth-aachen.de`
** Mascotte joint Project of INRIA/CNRS/UNSA, Sophia-Antipolis, France; and Graph Theory and Combinatorics Group, Applied Maths. IV Dept. of UPC, Barcelona, Spain. Email: `ignasi.sau@sophia.inria.fr`
*** Department of Computer Science, Technion, Haifa, Israel. Email: `zaks@cs.technion.ac.il`

an independent set in $\overline{G}$). The maximum cardinality of a clique in G is denoted by $\omega(G)$ and is termed the *clique number* of G. A *proper coloring* of G is an assignment of different colors to adjacent vertices, which results in a partition of V into independent sets. The minimum number of colors for which there exists a proper coloring is denoted by $\chi(G)$ and is termed the *chromatic number* of G. A partition of V into $\chi(G)$ independent sets is a *minimum coloring* of G.

Motivation and previous work. Besides generalizing interval and permutation graphs in a natural way, the class of tolerance graphs has other important subclasses and superclasses. Let us briefly survey some of them.

Given a class of graphs $\mathcal{G}$, a graph G is a *probe graph* of $\mathcal{G}$ if its vertices can be partitioned into a set $\mathbb{P}$ of probes and an independent set $\mathbb{N}$ of nonprobes, such that G can be extended to a graph of $\mathcal{G}$ by adding edges between certain nonprobes. Probe interval graphs, which are a superclass of interval graphs, have proved to be useful in the Human Genome Project [23]. Tolerance graphs are also a superclass of interval probe graphs [12].

A graph is *perfect* if the chromatic number of every induced subgraph equals the clique number of that subgraph. Perfect graphs include many important families of graphs, and serve to unify results relating colorings and cliques in those families. For instance, in all perfect graphs, the graph coloring problem, maximum clique problem, and maximum independent set problem can all be solved in polynomial time using the Ellipsoid method [14]. Since tolerance graphs were shown to be perfect [11], there exist polynomial time algorithms for these problems. However, these algorithms are not very efficient and therefore, as it happens for most known subclasses of perfect graphs, it makes sense to devise specific fast algorithms for these problems on tolerance graphs.

A *comparability* graph is a graph which can be transitively oriented. A *co-comparability* graph is a graph whose complement is a comparability graph. Bounded tolerance graphs are co-comparability graphs [10], and therefore all known polynomial time algorithms for co-comparability graphs apply to bounded tolerance graphs. This is one of the main reasons why for many problems the existing algorithms have better running time in bounded tolerance graphs than in general tolerance graphs.

A graph $G = (V, E)$ is the *intersection graph* of a family $F = \{S_1, \ldots, S_n\}$ of distinct nonempty subsets of a set S if there exists a bijection $\mu : V \to F$ such that for any two distinct vertices $u, v \in V$, $uv \in E$ if and only if $\mu(u) \cap \mu(v) \neq \emptyset$. In that case, we say that F is an *intersection model* of G. It is easy to see that each graph has a trivial intersection model based on adjacency relations [21]. Some intersection models provide a natural and intuitive understanding of the structure of a class of graphs, and turn out to be very helpful to find efficient algorithms to solve optimization problems [21]. Therefore, it is of great importance to establish non-trivial intersection models for families of graphs. A graph G on n vertices is a *parallelogram graph* if we can fix two parallel lines L_1 and L_2, and for each vertex $v_i \in V(G)$ we can assign a parallelogram $\overline{P}_i$ with parallel sides along L_1 and L_2 so that G is the intersection graph of $\{\overline{P}_i \mid i = 1, \ldots, n\}$. It was proved in [1, 20] that a graph is a bounded tolerance graph if and only if it is a parallelogram graph. This characterization provides a useful way to think about bounded tolerance graphs. However, this intersection model cannot cope with general tolerance graphs, in which the tolerance of an interval can be greater than its length.

Our contribution. In this article we present the first non-trivial intersection model for general tolerance graphs, which generalizes the widely known parallelogram representation of bounded tolerance graphs. The main idea is to exploit the third dimension to capture the information given by unbounded tolerances, and as a result parallelograms are replaced with parallelepipeds. The proposed intersection model is very intuitive and can be efficiently constructed from a tolerance representation (actually, we show that it can be constructed in linear time).

Apart from being important on its own, this new representation proves to be a powerful tool for designing efficient algorithms for general tolerance graphs. Indeed, using our intersection model we improve the best existing running times of three problems on tolerance graphs. We present algorithms to find a minimum coloring and a maximum clique in $\mathcal{O}(n \log n)$ time, which is optimal (as discussed in Section 3.4). The best existing algorithm was $\mathcal{O}(n^2)$ [12,13]. We also present an algorithm to find a maximum weight independent set in $\mathcal{O}(n^2)$ time, whereas the best known algorithm was $\mathcal{O}(n^3)$ [13]. We note that [22] proposes an $\mathcal{O}(n^2 \log n)$ algorithm to find a maximum *cardinality* independent set on a general tolerance graph, and that [13] refers to an algorithm transmitted by personal communication with running time $\mathcal{O}(n^2 \log n)$ to find a maximum weight independent set on a general tolerance graph; to the best of our knowledge, this algorithm has not been published.

It is important to note that the complexity of recognizing bounded and general tolerance graphs is a challenging open problem [3,13,22], and this is the reason why we assume throughout this paper that along with the input tolerance graph we are also given a tolerance representation of it. The only "positive" result in the literature concerning recognition of tolerance graphs is a linear time algorithm for the recognition of bipartite tolerance graphs [3].

Nevertheless, it was shown in [15] that every tolerance graph has a polynomial sized tolerance representation, and hence tolerance graphs recognition is in the class NP. There exist other graph classes closely related to tolerance graphs. If in the definition of tolerance graphs we replace the operation "min" between tolerances with "+", we obtain the class of *sum-tolerance graphs* [17], and if we replace it with "max" we obtain the class of *max-tolerance graphs*. Max-tolerance graphs recognition is known to be NP-hard [18].

Organization of the paper. We provide the new intersection model of general tolerance graphs in Section 2. In Section 3 we present a canonical representation of tolerance graphs, and then show how it can be used in order to obtain optimal $\mathcal{O}(n \log n)$ algorithms for finding a minimum coloring and a maximum clique in a tolerance graph. In Section 4 we present an $\mathcal{O}(n^2)$ algorithm for finding a maximum weight independent set. Finally, Section 5 is devoted to conclusions and open problems.

2 A New Intersection Model for Tolerance Graphs

One of the most natural representations of bounded tolerance graphs is given by parallelograms between two parallel lines in the Euclidean plane [1, 13, 20]. In this section we extend this representation to a three-dimensional representation of general tolerance graphs.

Given a tolerance graph $G = (V, E)$ along with a tolerance representation of it, recall that vertex $v_i \in V$ corresponds to an interval $I_i = [a_i, b_i]$ on the real line with a tolerance $t_i \geq 0$. W.l.o.g. we may assume that $t_i > 0$ for every vertex v_i [13].

Definition 1. *Given a tolerance representation of a tolerance graph $G = (V, E)$, vertex v_i is* bounded *if $t_i \leq |I_i|$. Otherwise, v_i is* unbounded*. V_B and V_U are the sets of bounded and unbounded vertices in V, respectively. Clearly $V = V_B \cup V_U$.*

We can also assume w.l.o.g. that $t_i = \infty$ for any unbounded vertex v_i, since if v_i is unbounded, then the intersection of any other interval with I_i is strictly smaller than t_i. Let L_1 and L_2 be two parallel lines at distance 1 in the Euclidean plane.

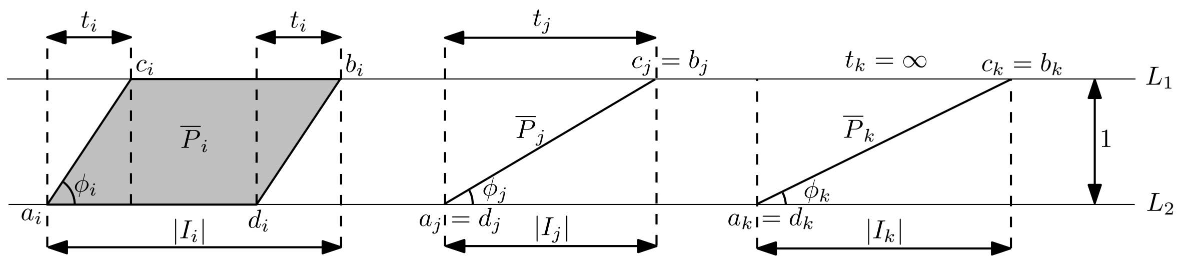

Fig. 1. Parallelograms $\overline{P}_i$ and $\overline{P}_j$ correspond to bounded vertices v_i and v_j, respectively, whereas $\overline{P}_k$ corresponds to an unbounded vertex v_k.

Definition 2. *Given an interval $I_i = [a_i, b_i]$ with tolerance t_i, $\overline{P}_i$ is the parallelogram defined by the points c_i, b_i in L_1 and a_i, d_i in L_2, where $c_i = \min\{b_i, a_i + t_i\}$ and $d_i = \max\{a_i, b_i - t_i\}$. The* slope *$\phi_i$ of $\overline{P}_i$ is $\phi_i = \arctan\left(\frac{1}{c_i - a_i}\right)$.*

An example is depicted in Figure 1, where $\overline{P}_i$ and $\overline{P}_j$ correspond to bounded vertices v_i and v_j, and $\overline{P}_k$ corresponds to an unbounded vertex v_k. Observe that when vertex v_i is bounded, the values c_i and d_i coincide with the *tolerance points* defined in [7,13,16], and $\phi_i = \arctan\left(\frac{1}{t_i}\right)$. On the other hand, when vertex v_i is unbounded, the values c_i and d_i coincide with the endpoints b_i and a_i of I_i, respectively, and $\phi_i = \arctan\left(\frac{1}{|I_i|}\right)$. Observe also that in both cases $t_i = b_i - a_i$ and $t_i = \infty$, parallelogram $\overline{P}_i$ is reduced to a line segment (c.f. $\overline{P}_j$ and $\overline{P}_k$ in Figure 1). Since $t_i > 0$ for every vertex v_i, it follows that $0 < \phi_i < \frac{\pi}{2}$. Furthermore, we can assume w.l.o.g. that all points a_i, b_i, c_i, d_i and all slopes ϕ_i are distinct [7,13,16].

Observation 1 *Let $v_i \in V_U, v_j \in V_B$. Then $|I_i| < t_j$ if and only if $\phi_i > \phi_j$.*

We are ready to give the main definition of this article.

Definition 3. *Let $G = (V, E)$ be a tolerance graph with a tolerance representation $\{I_i = [a_i, b_i], t_i \mid i = 1, \ldots, n\}$. For every $i = 1 \ldots, n$, P_i is the parallelepiped in $\mathbb{R}^3$ defined as follows:*

(a) If $t_i \leq b_i - a_i$ (that is, v_i is bounded), then $P_i = \{(x, y, z) \in \mathbb{R}^3 \mid (x, y) \in \overline{P}_i,\ 0 \leq z \leq \phi_i\}$.
(b) If $t_i > b_i - a_i$ (v_i is unbounded), then $P_i = \{(x, y, z) \in \mathbb{R}^3 \mid (x, y) \in \overline{P}_i,\ z = \phi_i\}$.

The set of parallelepipeds $\{P_i \mid i = 1, \ldots, n\}$ is a parallelepiped representation *of G.*

Observe that for each interval I_i, the parallelogram $\overline{P}_i$ of Definition 2 (see also Figure 1) coincides with the projection of the parallelepiped P_i on the plane $z = 0$. An example of the construction of these parallelepipeds is given in Figure 2, where a set of eight intervals with their associated tolerances is given in Figure 2(a). The corresponding tolerance graph G is depicted in Figure 2(b), while the parallelepiped representation is illustrated in Figure 2(c). In the case $t_i < b_i - a_i$, the parallelepiped P_i is three-dimensional, c.f. P_1, P_3, and P_5, while in the border case $t_i = b_i - a_i$ it degenerates to a two-dimensional rectangle, c.f. P_7. In these two cases, each P_i corresponds to a bounded vertex v_i. In the remaining case $t_i = \infty$ (that is, v_i is unbounded), the parallelepiped P_i degenerates to a one-dimensional line segment above plane $z = 0$, c.f. P_2, P_4, P_6, and P_8.

We prove now that these parallelepipeds form a three-dimensional intersection model for the class of tolerance graphs (namely, that every tolerance graph G can be viewed as the intersection graph of the corresponding parallelepipeds P_i).

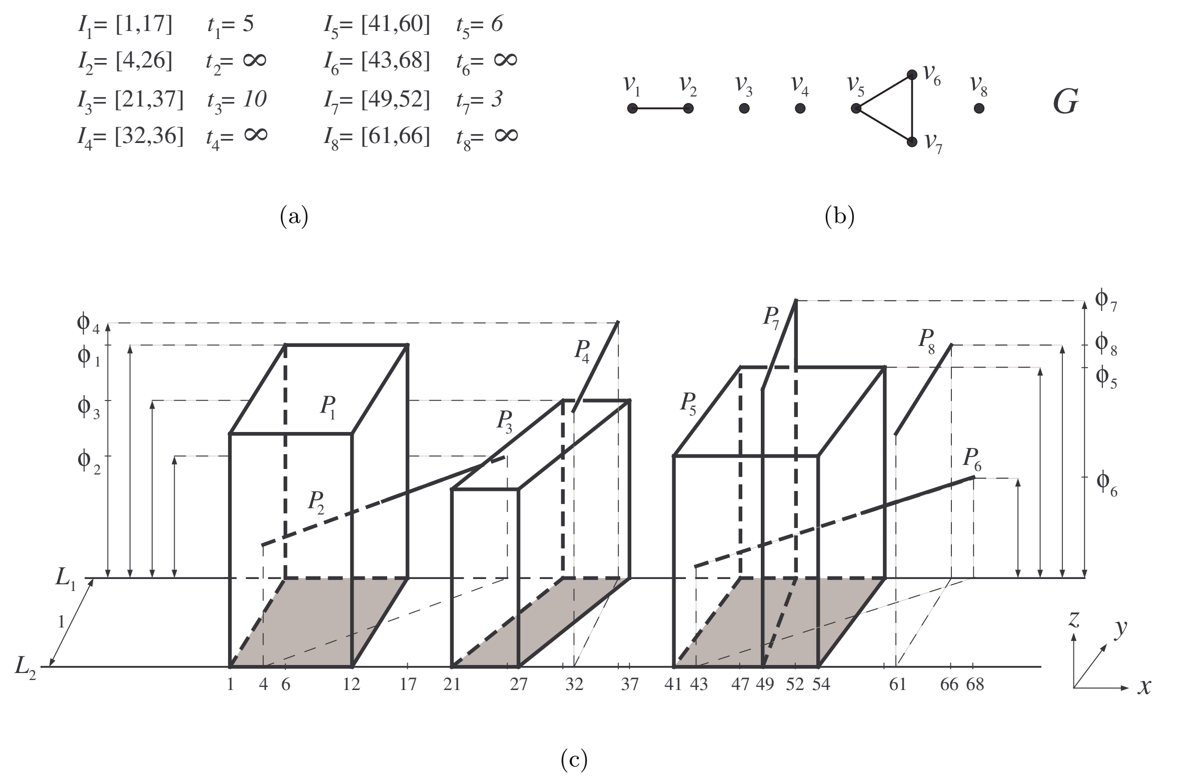

Fig. 2. The intersection model for tolerance graphs: (a) a set of intervals $I_i = [a_i, b_i]$ and tolerances t_i, $i = 1, \ldots, 8$, (b) the corresponding tolerance graph G and (c) a parallelepiped representation of G.

Theorem 1. *Let $G = (V, E)$ be a tolerance graph with a tolerance representation $\{I_i = [a_i, b_i], t_i \mid i = 1, \ldots, n\}$. Then for every $i \neq j$, $v_iv_j \in E$ if and only if $P_i \cap P_j \neq \emptyset$.*

Proof. We distinguish three cases according to whether vertices v_i and v_j are bounded or unbounded:

(a) Both vertices are bounded, that is $t_i \leq b_i - a_i$ and $t_j \leq b_j - a_j$. It follows from [13] that $v_iv_j \in E(G)$ if and only if $\overline{P}_i \cap \overline{P}_j \neq \emptyset$. However, due to the definition of the parallelepipeds P_i and P_j, in this case $P_i \cap P_j \neq \emptyset$ if and only if $\overline{P}_i \cap \overline{P}_j \neq \emptyset$ (c.f. P_1 and P_3, or P_5 and P_7, in Figure 2).

(b) Both vertices are unbounded, that is $t_i = t_j = \infty$. Since no two unbounded vertices are adjacent, $v_iv_j \notin E(G)$. On the other hand, the line segments P_i and P_j lie on the disjoint planes $z = \phi_i$ and $z = \phi_j$ of $\mathbb{R}^3$, respectively, since we assumed that the slopes ϕ_i and ϕ_j are distinct. Thus, $P_i \cap P_j = \emptyset$ (c.f. P_2 and P_4).

(c) One vertex is unbounded (that is, $t_i = \infty$) and the other is bounded (that is, $t_j \leq b_j - a_j$). If $\overline{P}_i \cap \overline{P}_j = \emptyset$, then $v_iv_j \notin E$ and $P_i \cap P_j = \emptyset$ (c.f. P_1 and P_6). Suppose that $\overline{P}_i \cap \overline{P}_j \neq \emptyset$. We distinguish two cases:

  (i) $\phi_i < \phi_j$. It is easy to check that $|I_i \cap I_j| \geq t_j$ and thus $v_iv_j \in E$. Since $\overline{P}_i \cap \overline{P}_j \neq \emptyset$ and $\phi_i < \phi_j$, then necessarily the line segment P_i intersects with the parallelepiped P_j on the plane $z = \phi_i$, and thus $P_i \cap P_j \neq \emptyset$ (c.f. P_1 and P_2).

  (ii) $\phi_i > \phi_j$. Clearly $|I_i \cap I_j| < t_i = \infty$. Furthermore, since $\phi_i > \phi_j$, Observation 1 implies that $|I_i \cap I_j| \leq |I_j| < t_i$. It follows that $|I_i \cap I_j| < \min\{t_i, t_j\}$, and thus $v_iv_j \notin E$. On the other hand, $z = \phi_i$ for all points $(x, y, z) \in P_i$, while $z \leq \phi_j < \phi_i$ for all points $(x, y, z) \in P_j$, and therefore $P_i \cap P_j = \emptyset$ (c.f. P_3 and P_4). □

Clearly, for each $v_i \in V$ the parallelepiped P_i can be constructed in constant time. Therefore,

Lemma 1. *Given a tolerance representation of a tolerance graph G with n vertices, a parallelepiped representation of G can be constructed in $\mathcal{O}(n)$ time.*

3 Coloring and Clique Algorithms in $\mathcal{O}(n \log n)$

In this section we present optimal $\mathcal{O}(n \log n)$ algorithms for constructing a minimum coloring and a maximum clique in a tolerance graph $G = (V, E)$ with n vertices, given a parallelepiped representation of G. These algorithms improve the best known running time $\mathcal{O}(n^2)$ of these problems on tolerance graphs [12,13]. First, we introduce a canonical representation of tolerance graphs in Section 3.1, and then we use it to obtain the algorithms for the minimum coloring and the maximum clique problems in Section 3.2. Finally, we discuss the optimality of both algorithms in Section 3.4.

3.1 A canonical representation of tolerance graphs

We associate with every vertex v_i of G the point $p_i = (x_i, y_i)$ in the Euclidean plane, where $x_i = b_i$ and $y_i = \frac{\pi}{2} - \phi_i$. Since all endpoints of the parallelograms $\overline{P}_i$ and all slopes ϕ_i are distinct, all coordinates of the points p_i are distinct as well. Similarly to [12, 13], we state the following two definitions.

Definition 4. *An unbounded vertex $v_i \in V_U$ of a tolerance graph G is called* inevitable *(for a certain parallelepiped representation), if replacing P_i with $\{(x, y, z) \mid (x, y) \in P_i, 0 \leq z \leq \phi_i\}$ creates a new edge in G. Otherwise, v_i is called* evitable.

Definition 5. *Let $v_i \in V_U$ be an inevitable unbounded vertex of a tolerance graph G (for a certain parallelepiped representation). A vertex v_j is called a* hovering vertex *of v_i if $a_j < a_i$, $b_i < b_j$, and $\phi_i > \phi_j$.*

It is now easy to see that, by Definition 5, if v_j is a hovering vertex of v_i, then $v_i v_j \notin E$. Note that, in contrast to [12], in Definition 4, an isolated vertex v_i might be also inevitable unbounded, while in Definition 5, a hovering vertex might be also unbounded. Definitions 4 and 5 imply the following lemma:

Lemma 2. *Let $v_i \in V_U$ be an inevitable unbounded vertex of the tolerance graph G (for a certain parallelepiped representation). Then, there exists a hovering vertex v_j of v_i.*

Proof. Since v_i is an inevitable unbounded vertex, replacing P_i with $\{(x, y, z) \mid (x, y) \in P_i, 0 \leq z \leq \phi_i\}$ creates a new edge in G; let $v_i v_j$ be such an edge. Then, clearly $\overline{P}_i \cap \overline{P}_j \neq \emptyset$. We will prove that v_j is a hovering vertex of v_i. Otherwise, $\phi_i < \phi_j$, $a_j > a_i$, or $b_i > b_j$. Suppose first that $\phi_i < \phi_j$. If $v_j \in V_U$, then v_i remains not connected to v_j after the replacement of P_i with $\{(x, y, z) \mid (x, y) \in P_i, 0 \leq z \leq \phi_i\}$, since $\phi_i < \phi_j$, which is a contradiction. If $v_j \in V_B$, then v_i is connected to v_j also before the replacement of P_i, since $\phi_i < \phi_j$ and $\overline{P}_i \cap \overline{P}_j \neq \emptyset$, which is again a contradiction. Thus, $\phi_i > \phi_j$. Suppose now that $a_j > a_i$ or $b_i > b_j$. Then, since $\phi_i > \phi_j$, we obtain for both cases that $\overline{P}_i \cap \overline{P}_j = \emptyset$, which is a contradiction. Thus, $a_j < a_i$, $b_i < b_j$, and $\phi_i > \phi_j$, i.e. v_j is a hovering vertex of v_i by Definition 5. $\square$

Definition 6. *A parallelepiped representation of a tolerance graph G is called* canonical *if every unbounded vertex is inevitable.*

For example, in the tolerance graph depicted in Figure 2, v_4 and v_8 are inevitable unbounded vertices, v_3 and v_6 are hovering vertices of v_4 and v_8, respectively, while v_2 and v_6 are evitable unbounded vertices. Therefore, this representation is not canonical for the graph G. However, if we replace P_i with $\{(x, y, z) \mid (x, y) \in P_i, 0 \leq z \leq \phi_i\}$ for $i = 2, 6$, we get a canonical representation for G.

In the following, we present an algorithm that constructs a canonical representation of a given tolerance graph G.

Definition 7. *Let $\alpha = (x_\alpha, y_\alpha)$ and $\beta = (x_\beta, y_\beta)$ be two points in the plane. Then α* dominates *β if $x_\alpha > x_\beta$ and $y_\alpha > y_\beta$. Given a set A of points, the point $\gamma \in A$ is called an* extreme point *of A if there is no point $\delta \in A$ that dominates γ. $Ex(A)$ is the set of the extreme points of A.*

Given a tolerance graph $G = (V, E)$ with the set $V = \{v_1, v_2, \ldots, v_n\}$ of vertices (and its parallelepiped representation), we can assume w.l.o.g. that $a_i < a_j$ whenever $i < j$. Recall that with every vertex v_i we associated the point $p_i = (x_i, y_i)$, where $x_i = b_i$ and $y_i = \frac{\pi}{2} - \phi_i$, respectively. We define for every $i = 1, 2, \ldots, n$ the set $A_i = \{p_1, p_2, \ldots, p_i\}$ of the points associated with the first i vertices of G.

Lemma 3. *Let $v_i \in V_U$ be an unbounded vertex of a tolerance graph G. Then:*

(a) If $p_i \in Ex(A_i)$ then v_i is evitable.
(b) If $p_i \notin Ex(A_i)$ and point p_j dominates p_i for some bounded vertex $v_j \in V_B$ with $j < i$ then v_i is inevitable and v_j is a hovering vertex of v_i.

Proof. (a) Assume, to the contrary, that v_i is inevitable. By Lemma 2 there is a hovering vertex v_j of v_i. But then, $x_i = b_i < b_j = x_j$ and $y_i = \frac{\pi}{2} - \phi_i < \frac{\pi}{2} - \phi_j = y_j$, while $a_j < a_i$, i.e. $j < i$. Therefore $p_j \in A_i$ and p_j dominates p_i, which is a contradiction, since $p_i \in Ex(A_i)$.

(b) Suppose that p_j dominates p_i, for some vertex $v_j \in V_B$ with $j < i$. The ordering of the vertices implies $a_j < a_i$, while $x_i < x_j$ and $y_i < y_j$ imply $b_i < b_j$ and $\phi_i > \phi_j$. Thus v_i is inevitable and v_j is a hovering vertex of v_i. □

The following theorem shows that, given a parallelepiped representation of a tolerance graph G, we can construct in $\mathcal{O}(n \log n)$ a canonical representation of G. This result is crucial for the time complexity analysis of the algorithms of Section 3.2.

Theorem 2. *Every parallelepiped representation of a tolerance graph G with n vertices can be transformed by Algorithm 1 to a canonical representation of G in $\mathcal{O}(n \log n)$ time.*

Proof. We describe and analyze Algorithm 1 that generates a canonical representation of G. First, we sort the vertices $v_1, v_2, \ldots, v_n$ of G such that $a_i < a_j$ whenever $i < j$. Then, we process sequentially all vertices v_i of G. The bounded and the inevitable unbounded vertices will not be changed, while the evitable unbounded vertices will be replaced with bounded ones. At step i we update the set $Ex(A_i)$ of the extreme points of A_i (note that the set A_i remains unchanged during the algorithm). For two points p_{i_1}, p_{i_2} of $Ex(A_i)$, $x_{i_1} > x_{i_2}$ if and only if $y_{i_1} < y_{i_2}$. We store the elements of $Ex(A_i)$ in a list P, in which the points p_j are sorted increasingly according to their x values (or, equivalently, decreasingly according to their y values). Due to Lemma 3(a), and since during the algorithm the evitable unbounded vertices of G are replaced with bounded ones, after the process of vertex v_i, all points in the list P correspond to bounded vertices of G in the current parallelepiped representation.

We distinguish now the following cases:

Algorithm 1 Construction of a canonical representation of a tolerance graph G

Input: A parallelepiped representation R of a given tolerance graph G with n vertices
Output: A canonical representation R' of G

```
Sort the vertices of G, such that a_i < a_j whenever i < j
ℓ_0 ← min{x_i : 1 ≤ i ≤ n}; r_0 ← max{x_i : 1 ≤ i ≤ n}
p_s ← (ℓ_0 − 1, π/2); p_t ← (r_0 + 1, 0)
P ← (p_s, p_t); R' ← R
for i = 1 to n do
   Find the point p_j having the smallest x_j with x_j > x_i
   if y_j < y_i then {no point of P dominates p_i}
      Find the point p_k having the greatest x_k with x_k < x_i
      Find the point p_ℓ having the greatest y_ℓ with y_ℓ < y_i
      if x_k ≥ x_ℓ then
         Replace points p_ℓ, p_{ℓ+1} ..., p_k with point p_i in the list P
      else
         Insert point p_i between points p_k and p_ℓ in the list P
      if v_i ∈ V_U then {v_i is an evitable unbounded vertex}
         Replace P_i with {(x, y, z) | (x, y) ∈ P_i, 0 ≤ z ≤ φ_i} in R'
   else {y_j > y_i; p_j dominates p_i}
      if v_i ∈ V_U then {v_i is an inevitable unbounded vertex}
         v_j is a hovering vertex of v_i
return R'
```

Case 1. v_i is bounded. If there exists a point of P that dominates p_i then $p_i \notin Ex(A_i)$. Thus, we do not change P, and we continue to the process of v_{i+1}. If no point of P dominates p_i then $p_i \in Ex(A_i)$. Thus, we add p_i to P and we remove from P all points that are dominated by p_i.

Case 2. v_i is unbounded. If there exists a point $p_j \in P$ that dominates p_i then $p_i \notin Ex(A_i)$, while Lemma 3(b) implies that v_i is inevitable and v_j is a hovering vertex of v_i. Thus, similarly to Case 1, we do not change P, and we continue to the process of v_{i+1}. If no point of P dominates p_i then $p_i \in Ex(A_i)$. Thus, we add the point p_i to P and remove from P all points that are dominated by p_i. In this case, v_i is evitable by Lemma 3(a). Hence, we replace P_i with $\{(x,y,z) \mid (x,y) \in P_i, 0 \le z \le \phi_i\}$ in the current parallelepiped representation of G and we consider from now on v_i as a bounded vertex.

It follows that after the process of each vertex v_i (either bounded or unbounded) the list P stores the points of $Ex(A_i)$. Furthermore, at every iteration of the algorithm, all points of the list P correspond to bounded vertices in the current parallelepiped representation of G.

The processing of vertex v_i is done by executing three binary searches in the list P as follows. Let $\ell_0 = \min\{x_i \mid 1 \le i \le n\}$ and $r_0 = \max\{x_i \mid 1 \le i \le n\}$. For convenience, we add two dummy points $p_s = (\ell_0 - 1, \frac{\pi}{2})$ and $p_t = (r_0 + 1, 0)$. First, we find the point $p_j \in P$ with the smallest value x_j, such that $x_j > x_i$ (see Figure 3). Note that $p_i \in Ex(A_i)$ if and only if $y_j < y_i$. If $y_j > y_i$ then p_j dominates p_i (see Figure 3(a)). Thus, if $v_i \in V_U$, Lemma 3(b) implies that v_i is an inevitable unbounded vertex and v_j is a hovering vertex of v_i. In the opposite case $y_j < y_i$, we have to add p_i to P. In order to remove from P all points that are dominated by p_i, we execute binary search two more times. In particular, we find the points p_k and p_ℓ of P with the greatest values x_k and y_ℓ, respectively, such that $x_k < x_i$ and $y_\ell < y_i$ (see Figure 3(b)). If there are some points of P that are dominated by p_i, then p_k and p_ℓ have the greatest and smallest values x_k and x_ℓ among them, respectively, and $x_k \ge x_\ell$. In this case, we replace all points $p_\ell, p_{\ell+1}, \ldots, p_k$ with the point p_i in the list P. Otherwise, if no point of P is dominated by p_i, then $x_k < x_\ell$. In this case, we remove no point from P and we insert p_i between p_k and p_ℓ in P.

Finally, after processing all vertices v_i of G, we return a canonical representation of the given tolerance graph G, in which every vertex that remains unbounded has a hovering vertex

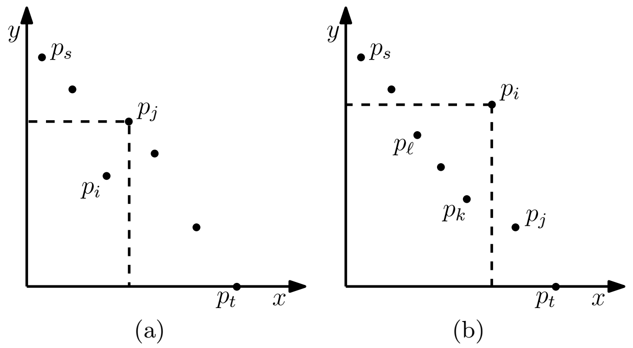

Fig. 3. The cases where the associated point p_i to the currently processed vertex v_i is (a) dominated by the point p_j in A_i and (b) an extreme point of the set A_i.

assigned to it. Since the processing of every vertex can be done in $\mathcal{O}(\log n)$ time by executing three binary searches, and since the sorting of the vertices can be done in $\mathcal{O}(n \log n)$ time, the running time of Algorithm 1 is $\mathcal{O}(n \log n)$. □

3.2 Minimum coloring

In the next theorem we present an optimal $\mathcal{O}(n \log n)$ algorithm for computing a minimum coloring of a tolerance graph G with n vertices, given a parallelepiped representation of G. The informal description of the algorithm is identical to the one in [12], which has running time $\mathcal{O}(n^2)$; the difference is in the fact that we use our new representation, in order to improve the time complexity.

Algorithm 2 Minimum coloring of a tolerance graph G

Input: A parallelepiped representation of a given tolerance graph G
Output: A minimum coloring of G

Construct a canonical representation of G by Algorithm 1, where a hovering vertex is associated with every inevitable unbounded vertex
Color $G[V_B]$ by the algorithm of [6]
for every inevitable unbounded vertex $v_i \in V_U$ **do**
  Assign to v_i the same color as its hovering vertex in $G[V_B]$

Theorem 3. *A minimum coloring of a tolerance graph G with n vertices can be computed in $\mathcal{O}(n \log n)$ time.*

Proof. We present Algorithm 2 that computes a minimum coloring of G. Given a parallelepiped representation of G, we construct a canonical representation of G in $\mathcal{O}(n \log n)$ time by Algorithm 1. V_B and V_U are the sets of bounded and inevitable unbounded vertices of G in the latter representation, respectively. In particular, Algorithm 1 associates a hovering vertex $v_j \in V_B$ with every inevitable unbounded vertex $v_i \in V_U$. We find a minimum proper coloring of the bounded tolerance graph $G[V_B]$ in $\mathcal{O}(n \log n)$ time using the algorithm of [6]. Finally, we associate with every inevitable unbounded vertex $v_i \in V_U$ the same color as that of its hovering vertex $v_j \in V_B$ in the coloring of $G[V_B]$.

Consider an arbitrary inevitable unbounded vertex $v_i \in V_U$ and its hovering vertex $v_j \in V_B$. Following Definition 5, $\overline{P}_i \cap \overline{P}_j \neq \emptyset$ and $\phi_i > \phi_j$. Consider a vertex v_k of G, such that $v_i v_k \in E$. It follows that $v_k \in V_B$, since no two unbounded vertices are adjacent in G. Furthermore, since

$v_i v_k \in E$, it follows that $\overline{P}_i \cap \overline{P}_k \neq \emptyset$ and $\phi_k > \phi_i$. Then $\overline{P}_j \cap \overline{P}_k \neq \emptyset$, and thus $P_j \cap P_k \neq \emptyset$, i.e. $v_j v_k \in E$, since both v_j and v_k are bounded vertices. It follows that v_k does not have the same color as v_j in the proper coloring of $G[V_B]$, and thus the resulting coloring of G is proper. Finally, since both colorings of $G[V_B]$ and of G have the same number of colors, it follows that this proper coloring of G is minimum. Since the coloring of $G[V_B]$ can be done in $\mathcal{O}(n \log n)$ time and the coloring of all inevitable unbounded vertices $v_i \in V_U$ can be done in $\mathcal{O}(n)$ time, Algorithm 2 returns a minimum proper coloring G in $\mathcal{O}(n \log n)$ time. □

3.3 Maximum clique

In the next theorem we prove that a maximum clique of a tolerance graph G with n vertices can be computed in optimal $\mathcal{O}(n \log n)$ time, given a parallelepiped representation of G. This theorem follows from Theorem 2 and from the clique algorithm presented in [6], and it improves the best known $\mathcal{O}(n^2)$ running time mentioned in [12].

Theorem 4. *A maximum clique of a tolerance graph G with n vertices can be computed in $\mathcal{O}(n \log n)$ time.*

Proof. We compute first a canonical representation of G in $\mathcal{O}(n \log n)$ time by Algorithm 1. The proof of Theorem 3 implies that $\chi(G) = \chi(G[V_B])$, where $\chi(H)$ denotes the chromaric number of a given graph H. Since tolerance graphs are perfect graphs [11], $\omega(G) = \chi(G)$ and $\omega(G[V_B]) = \chi(G[V_B])$, where $\omega(H)$ denotes the clique number of a given graph H. It follows that $\omega(G) = \omega(G[V_B])$. We compute now a maximum clique Q of the bounded tolerance graph $G[V_B]$ in $\mathcal{O}(n \log n)$ time. This can be done by the algorithm presented in [6] that computes a maximum clique in a trapezoid graph, since bounded tolerance graphs are trapezoid graphs [13]. Since $\omega(G) = \omega(G[V_B])$, Q is a maximum clique of G as well. □

3.4 Optimality of the running time

In this section we use permutation graphs [13]. Given a sequence $S = a_1, a_2, \ldots, a_n$ of numbers, a *subsequence* of S is a sequence $S' = a_{i_1}, a_{i_2}, \ldots, a_{i_k}$, where $a_{i_j} \in S$ for every $j \in \{1, 2, \ldots, k\}$, and $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. S' is called an *increasing subsequence* of S, if $a_{i_1} < a_{i_2} < \ldots < a_{i_k}$. Clearly, increasing subsequences in a permutation graph G correspond to independent sets of G, while increasing subsequences in the complement $\overline{G}$ of G correspond to cliques of G, where $\overline{G}$ is also a permutation graph. Since $\Omega(n \log n)$ is a lower time bound for computing the length of a longest increasing subsequence in a permutation [6, 8], the same lower time bound holds for computing a maximum clique and a maximum independent set in a permutation graph G. Furthermore, since permutation graphs are perfect graphs [9], the chromatic number $\chi(G)$ of a permutation graph G equals the clique number $\omega(G)$ of G. Thus, $\Omega(n \log n)$ is a lower time bound for computing the chromatic number of a permutation graph. Finally, since the class of permutation graphs is a subclass of tolerance graphs [13], the same lower bounds hold for tolerance graphs. It follows that the algorithms in Theorems 3 and 4 for computing a minimum coloring and a maximum clique in tolerance graphs are optimal.

4 Weighted Independent Set Algorithm in $\mathcal{O}(n^2)$

In this section we present an algorithm for computing a maximum weight independent set in a tolerance graph $G = (V, E)$ with n vertices in $\mathcal{O}(n^2)$ time, given a parallelepiped representation of G, and a weight $w(v_i) > 0$ for every vertex v_i of G. The proposed algorithm improves the running time $\mathcal{O}(n^3)$ of the one presented in [13]. In the following, consider as above the partition of the vertex set V into the sets V_B and V_U of bounded and unbounded vertices of G, respectively.

Similarly to [13], we add two isolated bounded vertices v_s and v_t to G with weights $w(v_s) = w(v_t) = 0$, such that the corresponding parallelepipeds P_s and P_t lie completely to the left and to the right of all other parallelepipeds of G, respectively. Since both v_s and v_t are bounded vertices, we augment the set V_B by the vertices v_s and v_t. In particular, we define the set of vertices $V'_B = V_B \cup \{v_s, v_t\}$ and the tolerance graph $G' = (V', E)$, where $V' = V'_B \cup V_U$. Since $G'[V'_B]$ is a bounded tolerance graph, it is a co-comparability graph as well [11,13]. A transitive orientation of the comparability graph $\overline{G'[V'_B]}$ can be obtained by directing each edge according to the upper left endpoints of the parallelograms $\overline{P}_i$. Formally, let $(V'_B, \prec)$ be the partial order defined on the bounded vertices V'_B, such that $v_i \prec v_j$ if and only if $v_iv_j \notin E$ and $c_i < c_j$. Recall that a *chain* of elements in a partial order is a set of mutually comparable elements in this order [4].

Observation 2 ([13]) *The independent sets of $G[V_B]$ are in one-to-one correspondence with the chains in the partial order $(V'_B, \prec)$ from v_s to v_t.*

For the sequel, recall that for every unbounded vertex $v_k \in V_U$ the parallelepiped P_k degenerates to a line segment, while the upper endpoints b_k and c_k of the parallelogram $\overline{P}_k$ coincide, i.e. $b_k = c_k$.

Definition 8. *For every $v_i, v_j \in V'_B$ with $v_i \prec v_j$, $L_i(j) = \{v_k \in V_U \mid b_i < c_k < c_j, v_iv_k \notin E\}$ and its weight $w(L_i(j)) = \sum_{v \in L_i(j)} w(v)$.*

Definition 9. *For every $v_j \in V'_B$, $R_j = \{v_k \in V_U \mid c_j < c_k < b_j, v_jv_k \notin E\}$ and its weight $w(R_j) = \sum_{v \in R_j} w(v)$.*

For every pair of bounded vertices $v_i, v_j \in V'_B$ with $v_i \prec v_j$, the set $L_i(j)$ consists of those unbounded vertices $v_k \in V_U$, for which $v_iv_k \notin E$ and whose upper endpoint $b_k = c_k$ of $\overline{P}_k$ lies between $\overline{P}_i$ and $\overline{P}_j$. Furthermore, $v_jv_k \notin E$ for every vertex $v_k \in L_i(j)$. Indeed, in the case where $\overline{P}_k \cap \overline{P}_j \neq \emptyset$, it holds $\phi_k > \phi_j$, since $b_k = c_k < c_j$, and thus $P_k \cap P_j = \emptyset$. Similarly, the set R_j consists of those unbounded vertices $v_k \in V_U$, for which $v_jv_k \notin E$ and whose upper endpoint $b_k = c_k$ of $\overline{P}_k$ lies between the upper endpoints c_j and b_j of $\overline{P}_j$. Furthermore, $v_iv_k \notin E$ for every vertex $v_k \in R_j$ as well. Indeed, since $v_jv_k \notin E$, it follows that $\phi_k > \phi_j$, and thus, $\overline{P}_i \cap \overline{P}_k = \emptyset$ and $P_i \cap P_k = \emptyset$. In particular, in the example of Figure 4, $L_1(2) = \{v_3, v_5\}$ and $R_2 = \{v_6\}$. In this figure, the line segments that correspond to the unbounded vertices v_4 and v_7, respectively, are drawn with dotted lines to illustrate the fact that $v_4v_1 \in E$ and $v_7v_2 \in E$.

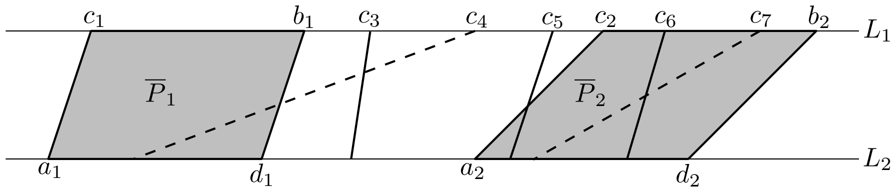

Fig. 4. The parallelograms $\overline{P}_i$, $i = 1, 2, \ldots, 7$ of a tolerance graph with the sets $V_B = \{v_1, v_2\}$ and $V_U = \{v_3, v_4, \ldots, v_7\}$ of bounded and unbounded vertices, respectively. In this graph, $L_1(2) = \{v_3, v_5\}$, $R_2 = \{v_6\}$ and $S(v_1, v_2) = \{v_3, v_5, v_6\}$.

Definition 10 ([13]). *For every $v_i, v_j \in V'_B$ with $v_i \prec v_j$, $S(v_i, v_j) = \{v_k \in V_U \mid v_iv_k, v_jv_k \notin E, b_i < c_k < b_j\}$.*

Algorithm 3 Maximum weight independent set of a tolerance graph G

Input: A parallelepiped representation of a given tolerance graph G
Output: The value of a maximum weight independent set of G

Add the dummy bounded vertices v_s, v_t to G, such that P_s and P_t lie completely to the left and to the right of all other parallelepipeds of G, respectively
$V'_B \leftarrow V_B \cup \{v_s, v_t\}$
Construct the partial ordering $(V'_B, \prec)$ of the bounded vertices V'_B
Sort the bounded vertices V'_B, such that $c_i < c_j$ whenever $i < j$

for $j = 1$ to $|V'_B|$ **do**
  $W(v_j) \leftarrow 0$
  Compute the value $w(R_j)$

for $i = 1$ to $|V'_B|$ **do**
  for every $v_j \in V'_B$ with $v_i \prec v_j$ **do**
    Update the value $w(L_i(j))$
    if $W(v_j) < (w(v_j) + w(R_j)) + W(v_i) + w(L_i(j))$ **then**
      $W(v_j) \leftarrow (w(v_j) + w(R_j)) + W(v_i) + w(L_i(j))$
return $W(v_t)$

Observation 3 *For every pair of bounded vertices $v_i, v_j \in V'_B$ with $v_i \prec v_j$,*

$$S(v_i, v_j) = L_i(j) \cup R_j \tag{1}$$

Furthermore, $L_i(j) \subseteq L_i(\ell)$ for every triple $\{v_i, v_j, v_\ell\}$ of bounded vertices, where $v_i \prec v_j$, $v_i \prec v_\ell$ and $c_j < c_\ell$.

In particular, in the example of Figure 4, $S(v_1, v_2) = L_1(2) \cup R_2 = \{v_3, v_5, v_6\}$.

Lemma 4 ([13]). *Given a tolerance graph G with a set of positive weights for the vertices of G, any maximum weight independent set of G consists of a chain of bounded vertices $v_{x_1} \prec v_{x_2} \prec \ldots \prec v_{x_k}$ together with the union of the sets $\cup\{S(v_{x_i}, v_{x_{i+1}}) \mid i = 0, 1, \ldots, k\}$, where $v_{x_0} = v_s$ and $v_{x_{k+1}} = v_t$.*

Now, using Lemma 4 and Observation 3, we can present Algorithm 3, which improves the running time $\mathcal{O}(n^3)$ of the one presented in [13].

Theorem 5. *A maximum weight independent set of a tolerance graph G with n vertices can be computed using Algorithm 3 in $\mathcal{O}(n^2)$ time.*

Proof. We present Algorithm 3 that computes the value of a maximum weight independent set of G. A slight modification of Algorithm 3 returns a maximum weight independent set of G, instead of its value. First, we construct the partial order $(V'_B, \prec)$ defined on the bounded vertices $V'_B = V_B \cup \{v_s, v_t\}$, such that $v_i \prec v_j$ whenever $v_i v_j \notin E$ and $c_i < c_j$. This can be done in $\mathcal{O}(n^2)$ time. Then, we sort the bounded vertices of V'_B, such that $c_i < c_j$ whenever $i < j$. This can be done in $\mathcal{O}(n \log n)$ time. As a preprocessing step, we compute for every bounded vertex $v_j \in V'_B$ the set R_j and its weight $w(R_j)$ in linear $\mathcal{O}(n)$ time by visiting at most all unbounded vertices $v_k \in V_U$. Thus, all values $w(R_j)$ are computed in $\mathcal{O}(n^2)$ time.

We associate with each bounded vertex $v_j \in V'_B$ a cumulative weight $W(v_j)$ defined as follows:

$$\begin{aligned} &W(v_s) = 0 \\ &W(v_j) = (w(v_j) + w(R_j)) + \max_{v_i \prec v_j}\{W(v_i) + w(L_i(j))\}, \text{ for every } v_j \in V'_B \setminus \{v_s\} \end{aligned} \tag{2}$$

The cumulative weight $W(v_j)$ of an arbitrary bounded vertex $v_j \in V'_B$ equals the maximum weight of an independent set S of vertices v_k (both bounded and unbounded), for which $b_k \leq b_j$ and $v_j \in S$. Initially all values $W(v_j)$ are set to zero.

In the main part of Algorithm 3, we process sequentially all bounded vertices $v_i \in V'_B$. For every such vertex v_i, we update sequentially the cumulative weights $W(v_j)$ for all bounded vertices $v_j \in V'_B$ with $v_i \prec v_j$ by comparing the current value of $W(v_j)$ with the value $(w(v_j) + w(R_j)) + W(v_i) + w(L_i(j))$, and by storing the greatest of them in $W(v_j)$. After all bounded vertices of V'_B have been processed, the value of the maximum weight independent set of G is stored in $W(v_t)$, due to Lemma 4 and Observation 3.

While processing the bounded vertex v_i, we compute the values $w(L_i(j))$ sequentially for every j, where $v_i \prec v_j$, as follows. Let v_{j_1}, v_{j_2} be two bounded vertices that are visited consecutively by the algorithm, during the process of vertex v_i. Then, due to Observation 3, we compute the value $w(L_i(j_2))$ by adding to the previous value $w(L_i(j_1))$ the weights of all unbounded vertices $v_k \in V_U$, whose upper endpoints c_k lie between c_{j_1} and c_{j_2}.

Since we visit all bounded and all unbounded vertices of the graph at most once during the process of v_i, this can be done in $\mathcal{O}(n)$ time. Thus, since there are in total at most $n+2$ bounded vertices $v_i \in V'_B$, Algorithm 3 returns the value of the maximum weight independent set of G in $\mathcal{O}(n^2)$ time. Finally, observe that, storing at every step of Algorithm 3 the independent sets that correspond to the values $W(v_i)$, and removing at the end the vertices v_s and v_t, the algorithm returns at the same time a maximum weight independent set of G, instead of its value. $\square$

5 Conclusions and Further Research

In this article we proposed the first non-trivial intersection model for general tolerance graphs, given by parallelepipeds in the three-dimensional space. This representation generalizes the parallelogram representation of bounded tolerance graphs. Using this representation, we presented improved algorithms for computing a minimum coloring, a maximum clique, and a maximum weight independent set on a tolerance graph. The running times of the first two algorithms are optimal. It can be expected that this representation will prove useful in improving the running time of other algorithms for the class of tolerance graphs.

As mentioned in Section 1, the complexity of the recognition problem for tolerance and bounded tolerance graphs is possibly the main open problem in this class of graphs. Even when the input graph is known to be a tolerance graph, it is not known how to obtain a tolerance representation for it [22]. Moreover, given a tolerance graph, it is not known how to decide in polynomial time whether it is a bounded tolerance graph [22].

References

1. K. P. Bogart, P. C. Fishburn, G. Isaak, and L. Langley. Proper and unit tolerance graphs. *Discrete Applied Mathematics*, 60(1-3):99–117, 1995.
2. A. H. Busch. A characterization of triangle-free tolerance graphs. *Discrete Applied Mathematics*, 154(3):471–477, 2006.
3. A. H. Busch and G. Isaak. Recognizing bipartite tolerance graphs in linear time. In *33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 12–20, 2007.
4. R. Diestel. *Graph Theory*. Springer-Verlag, Berlin, 3rd edition, 2005.
5. S. Felsner. Tolerance graphs and orders. *Journal of Graph Theory*, 28:129–140, 1998.
6. S. Felsner, R. Müller, and L. Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Applied Mathematics*, 74:13–32, 1997.
7. P. Fishburn and W. Trotter. Split semiorders. *Discrete Mathematics*, 195:111–126, 1999.
8. M. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11:29–35, 1975.

9. M. Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol. 57)*. North-Holland Publishing Co., 2004.
10. M. Golumbic and C. Monma. A generalization of interval graphs with tolerances. In *Proceedings of the 13th Southeastern Conference on Combinatorics, Graph Theory and Computing, Congressus Numerantium 35*, pages 321–331, 1982.
11. M. Golumbic, C. Monma, and W. Trotter. Tolerance graphs. *Discrete Applied Mathematics*, 9(2):157–170, 1984.
12. M. Golumbic and A. Siani. Coloring algorithms for tolerance graphs: Reasoning and scheduling with interval constraints. In *Joint International Conferences on Artificial Intelligence, Automated Reasoning, and Symbolic Computation (AISC/Calculemus)*, pages 196–207, 2002.
13. M. Golumbic and A. Trenk. *Tolerance Graphs*. Cambridge Studies in Advanced Mathematics, 2004.
14. M. Grötshcel, L. Lovász, and A. Schrijver. The Ellipsoid Method and its Consequences in Combinatorial Optimization. *Combinatorica*, 1:169–197, 1981.
15. R. B. Hayward and R. Shamir. A note on tolerance graph recognition. *Discrete Applied Mathematics*, 143(1-3):307–311, 2004.
16. G. Isaak, K. Nyman, and A. Trenk. A hierarchy of classes of bounded bitolerance orders. *Ars Combinatoria*, 69, 2003.
17. M. Jacobson and F. McMorris. Sum-tolerance proper interval graphs are precisely sum-tolerance unit interval graphs. *Journal of Combinatorics, Information and System Science*, 16:25–28, 1991.
18. M. Kaufmann, J. Kratochvíl, K. A. Lehmann, and A. R. Subramanian. Max-tolerance graphs as intersection graphs: cliques, cycles, and recognition. In *17th annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 832–841, 2006.
19. J. M. Keil and P. Belleville. Dominating the complements of bounded tolerance graphs and the complements of trapezoid graphs. *Discrete Applied Mathematics*, 140(1-3):73–89, 2004.
20. L. Langley. *Interval tolerance orders and dimension*. PhD thesis, Dartmouth College, June 1993.
21. T. McKee and F. McMorris. *Topics in Intersection Graph Theory*. Society for Industrial and Applied Mathematics (SIAM), 1999.
22. G. Narasimhan and R. Manber. Stability and chromatic number of tolerance graphs. *Discrete Applied Mathematics*, 36:47–56, 1992.
23. P. Zhang, E. A. Schon, S. G. Fischer, E. Cayanis, J. Weiss, S. Kistler, and P. E. Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. *CABIOS*, 10:309–317, 1994.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2003-01 * Jahresbericht 2002

2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting

2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations

2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs

2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard

2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates

2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung

2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs

2004-01 * Fachgruppe Informatik: Jahresbericht 2003

2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic

2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting

2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming

2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming

2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming

2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination

2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information

2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity

2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules

2005-01 * Fachgruppe Informatik: Jahresbericht 2004

2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"

2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions

2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots

2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut

2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures

2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts

2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises "Features"

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-10 Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes

2007-11 Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

2007-15 Volker Stolz: Temporal assertions for sequential and concurrent programs

2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks

2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting

2007-18 Uwe Naumann: Call Tree Reversal is NP-Complete

2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control

2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems

2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains

2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets

2008-01 * Fachgruppe Informatik: Jahresbericht 2007

2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing

2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René :Thiemann, Harald Zankl: Maximal Termination

2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler

2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations

2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs

2008-07 Alexander Nyßen, Horst Lichter:: The MeDUSA Reference Manual, Second Edition

2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs

2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

2008-10 George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time

2008-11 George B. Mertzios: Fast Convergence of Routing Games with Splittable Flows

2008-12 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang's method of stages

2008-13 Beatriz Alarcón, Fabian Emmes, Carsten Fuhs, Jürgen Giesl, Raúl Gutiérrez, Salvador Lucas, Peter Schneider-Kamp, René Thiemann: Improving Context-Sensitive Dependency Pairs

2008-14 Bastian Schlich: Model Checking of Software for Microcontrollers

2008-15 Joachim Kneis, Alexander Langer, Peter Rossmanith: A New Algorithm for Finding Trees with Many Leaves

2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study

2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving

2008-18 Falk Salewski: Empirical Evaluations of Safety-Critical Embedded Systems

2009-03 Alexander Nyßen: Model-Based Construction of Embedded & Real-Time Software - A Methodology for Small Devices

* These reports are only available as a printed version.

Please contact biblio@informatik.rwth-aachen.de to obtain copies.