RWTH Aachen

Department of Computer Science
Technical Report

A New Intersection Model for Multitolerance Graphs, Hierarchy, and Efficient Algorithms

George B. Mertzios

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2010-07

RWTH Aachen · Department of Computer Science · March 2010

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 for Multitolerance Graphs, Hierarchy, and Efficient Algorithms

George B. Mertzios*

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 has attracted many research efforts, mainly due to its interesting structure and its numerous applications, while a number of variations of this model has appeared. In particular, one of the most natural generalizations of tolerance graphs, namely *multitolerance* graphs, has been introduced in 1998 [25], where two tolerances are allowed for each interval. These two tolerances – one from the left and one from the right side of the interval – define an infinite number of the so called *tolerance-intervals*, which make the multitolerance model inconvenient to cope with. The main subclass of multitolerance graphs, namely bounded multitolerance graphs, coincide with the widely known class of trapezoid graphs that has been extensively studied. In this article we introduce the first non-trivial intersection model for general multitolerance graphs, given by objects in the three-dimensional space called *trapezoepipeds*, which unifies in a simple and intuitive way the trapezoid representation for bounded multitolerance graphs and the recently introduced parallelepiped representation for tolerance graphs [22]. Apart from being important on its own, this new intersection model proves to be a powerful tool for designing efficient algorithms. Given a multitolerance graph with n vertices and m edges, we present three new algorithms that compute a minimum coloring and a maximum clique in optimal $O(n \log n)$ time, and a maximum weight independent set in $O(m + n \log n)$ time – this algorithm also improves the best known running time of $O(n^2)$ for the same problem on tolerance graphs [22]. Moreover, we prove several structural results on the class of multitolerance graphs, complementing thus the hierarchy of perfect graphs given in [11]. The resulting hierarchy of classes of perfect graphs is *complete*, i.e. all inclusions are strict.

Keywords: Multitolerance graphs, tolerance 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 exists a collection $I = \{I_v \mid v \in V\}$ of closed intervals on the real line and a set $t = \{t_v \mid v \in V\}$ of positive numbers, such that for any two vertices $u, v \in V$, $uv \in E$ if and only if $|I_u \cap I_v| \geq \min\{t_u, t_v\}$, where $|I|$ denotes the length of the interval I. The pair $\langle I, t \rangle$ is called a *tolerance representation* of G. If G has a tolerance representation $\langle I, t \rangle$, such that $t_v \leq |I_v|$ for every $v \in V$, then G is called a *bounded tolerance* graph and $\langle I, t \rangle$ a *bounded tolerance representation* of G.

Tolerance graphs have been introduced in [8], in order to generalize some of the well known applications of interval graphs. The main motivation was in the context of resource allocation and scheduling problems, in which resources, such as rooms and vehicles, that normally would be used exclusively by one user, can tolerate sharing among users [11]. These graphs find numerous applications in constrained-based temporal reasoning, data transmission through networks to efficiently scheduling aircraft and crews, as well as contributing to genetic analysis and studies of the brain [10, 11]. Since their introduction, tolerance graphs have attracted many research efforts [1, 4, 5, 9–11, 14, 17, 22–24] as they generalize in a natural way both interval and permutation graphs [8]. Moreover, quite a number of variations of the model of tolerance graphs has appeared, such as *max-tolerance*, *sum-tolerance* etc. For a detailed survey we refer to the book of tolerance graphs [11].

*Department of Computer Science, RWTH Aachen, Germany. Email: mertzios@cs.rwth-aachen.de

In particular, a natural generalization of tolerance graphs, namely *multitolerance* graphs, has been introduced in 1998 [25]. The main idea is to allow two different tolerances to each interval, one to the left and one to the right side. That is, in the context of scheduling and resource allocation, a user can tolerate differently the sharing of a resource from the left and from the right side. It has been proved that the difference between the pathwidth and the treewidth of a multitolerance graph is at most one [25]. Furthermore, the minimum fill-in problem can be solved in polynomial time, when a multitolerance representation is given [25].

Formally, let $I = [l, r]$ be a closed interval on the real line and $l_t, r_t \in I$ be two numbers between l and r, called *tolerant points*. For every $\lambda \in [0,1]$, we define the interval $I_{l_t,r_t}(\lambda) = [l + (r_t - l)\lambda,\ l_t + (r - l_t)\lambda]$. Furthermore, we define the set $\mathcal{I}(I, l_t, r_t) = \{I_{l_t,r_t}(\lambda) \mid \lambda \in [0,1]\}$ of intervals. That is, $\mathcal{I}(I, l_t, r_t)$ is the set of all intervals that we obtain when we linearly transform $[l, l_t]$ into $[r_t, r]$. For an interval I, the *set of tolerance-intervals* τ of I is defined either as $\tau = \mathcal{I}(I, l_t, r_t)$ for some values $l_t, r_t \in I$ of tolerant points, or as $\tau = \{\mathbb{R}\}$. A graph $G = (V, E)$ is a *multitolerance* graph if there exists a collection $I = \{I_v \mid v \in V\}$ of closed intervals on the real line and a family $t = \{\tau_v \mid v \in V\}$ of sets of tolerance-intervals, such that: for any two vertices $u, v \in V$, $uv \in E$ if and only if there exists an element $Q_u \in \tau_u$ (resp. $Q_v \in \tau_v$) with $Q_u \subseteq I_v$ (resp. $Q_v \subseteq I_u$). Then, the pair $\langle I, t\rangle$ is called a *multitolerance representation* of G. If G has a multitolerance representation $\langle I, t\rangle$, in which $\tau \neq \{\mathbb{R}\}$ for every $v \in V$, then G is called a *bounded multitolerance* graph and $\langle I, t\rangle$ a *bounded multitolerance representation* of G.

A graph is *perfect* if the chromatic number of every induced subgraph equals the clique number of this subgraph. Perfect graphs include many important families of graphs, and serve to unify results relating colorings and cliques in those families. For instance, the minimum coloring, maximum clique, and maximum independent set problems can all be solved in polynomial time on perfect graphs using the Ellipsoid method [13]. Since multitolerance graphs are perfect [25], these problems can be solved in polynomial time on multitolerance graphs as well. However, these algorithms are not very efficient; thus, as it happens for most known subclasses of perfect graphs, it makes sense to design specific fast algorithms for these problems on multitolerance graphs.

A graph $G = (V, E)$ with n vertices 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 $\mu(u) \cap \mu(v) \neq \emptyset$. Then, F is called an *intersection model* of G. Note that every graph has a trivial intersection model based on adjacency relations [20]. Note that a multitolerance representation is not an intersection model, since two intervals may intersect without the corresponding vertices being necessarily adjacent. 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 obtain structural results, as well as to find efficient algorithms to solve optimization problems [20]. Therefore, it is of great importance to establish non-trivial intersection models for families of graphs. In particular, many important graph classes can be described as intersection graphs of set families that are derived from some kind of geometric configuration. For instance, a *permutation* (resp. *parallelogram* and *trapezoid*) graph is the intersection graph of line segments (resp. parallelograms and trapezoids) between two parallel lines L_1 and L_2 [7]. Such a representation with line segments (resp. parallelograms and trapezoids) is called a *permutation* (resp. *parallelogram* and *trapezoid*) *representation* of this graph.

A *comparability* graph is a graph which can be transitively oriented. A *cocomparability* graph is a graph whose complement is a comparability graph. Permutation graphs are a strict subset of parallelogram graphs [3]. Furthermore, parallelogram graphs are a strict subset of trapezoid graphs [26], and both are subsets of cocomparability graphs [7, 11]. On the other hand, not every tolerance graph is a cocomparability graph [7, 11]. Bounded tolerance graphs coincide with parallelogram graphs [1, 18]. Recently, a natural intersection model for general tolerance graphs has been presented in [22], given by parallelepipeds in the three-dimensional space. This parallelepiped representation generalizes the widely known parallelogram representation of bounded tolerance graphs, and has been used to improve the time complexity of three NP-hard problems on tolerance graphs [22].

Bounded multitolerance graphs (also known as *bounded bitolerance* graphs [2,11,16]) coincide with trapezoid graphs [25]. Thus, in particular, multitolerance graphs generalize also trapezoid graphs except tolerance graphs. Trapezoid graphs (or bounded bitolerance graphs) have received considerable attention in the literature, see e.g. [2,6,11,16,21]. However, the intersection model of trapezoids between two parallel lines can not cope with general multitolerance graphs, in which the set τ_v of tolerance-intervals for a vertex v can be $\tau_v = \{\mathbb{R}\}$. Therefore, the only way until now to deal with general multitolerance graphs was to use the inconvenient multitolerance representation, which uses an infinite number of tolerance-intervals. This is the main reason why only few algorithmic results concerning general multitolerance graphs are known [25].

Our contribution In this article we introduce the first non-trivial intersection model for general multitolerance graphs, given by objects in the three-dimensional space, called *trapezoepipeds*. This *trapezoepiped representation* unifies in a simple and intuitive way the widely known trapezoid representation for bounded multitolerance graphs and the parallelepiped representation for tolerance graphs [22]. The main idea is to exploit the third dimension to capture the information of the vertices with $\tau_v = \{\mathbb{R}\}$ as the set of tolerance-intervals. This intersection model can be constructed efficiently (in linear time), given a multitolerance representation.

Apart of being important on its own, the trapezoepiped representation can be also used to design efficient algorithms. Given a multitolerance graph with n vertices and m edges, we present three algorithms that compute a minimum coloring and a maximum clique in $O(n \log n)$ time (which is *optimal*), and a maximum weighted independent set in $O(m + n \log n)$ time (where $\Omega(n \log n)$ is a lower bound for the complexity of this problem). Note here that, although the parallelepiped representation of tolerance graphs is similar to the trapezoepiped representation of multitolerance graphs, the coloring and clique algorithms presented in [22] do not extend to the case of multitolerance graphs, and thus the here presented algorithms are new. On the contrary, the algorithm presented in [22] for the maximum weight independent set with complexity $O(n^2)$ on tolerance graphs can be extended with the same time complexity to the case of multitolerance graphs; nevertheless we present here a new algorithm for multitolerance graphs that achieves a better running time $O(m+n \log n)$, which also improves the best known running time of $O(n^2)$ on tolerance graphs [22]. To the best of our knowledge, no algorithm for these problems on multitolerance graphs has appeared until now, and thus the previously best known running times for these problems were those on perfect graphs [13].

Moreover, we prove several structural results on the class of multitolerance graphs, using our new intersection model and some known results from the hierarchy of perfect graphs given in [11]. In particular, we prove that multitolerance graphs *strictly* include tolerance and trapezoid graphs, as well as that they are *strictly* included in weakly chordal and in co-perfectly orderable graphs. Furthermore, we prove that multitolerance graphs are incomparable with alternately orientable and cocomparability graphs, i.e. none of these classes includes the other one. These results complement the hierarchy of perfect graphs given in [11], by placing multitolerance graphs in it. The resulting hierarchy of classes of perfect graphs is *complete*, i.e. all inclusions are strict.

Notation In this article we follow standard notation and terminology, see for instance [11]. We consider finite, simple, and undirected graphs. 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 *clique* if each two of its vertices are adjacent. 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.

A graph G is called *alternately orientable* if there exists an orientation F of G which is transitive on every chordless cycle of length at least 4, i.e. the directions of the oriented edges must alternate. A graph G is called *weakly chordal* (or *weakly triangulated*) if G has no induced subgraph isomorphic to the chordless cycle C_n with n vertices, or to its complement $\overline{C_n}$, for any $n \geq 5$. A vertex order $\prec$ of a graph G is called *perfect* if and only if G contains no induced path $abcd$ with $a \prec b$ and $d \prec c$. A graph G is called *co-perfectly orderable* if its complement $\overline{G}$ admits a perfect order. For more definitions we refer to [11].

Organization of the paper We present the new intersection model for multitolerance graphs in Section 2. In Section 3 we present a canonical representation of multitolerance graphs and an algorithm that computes it in $O(n \log n)$ time. Then, using this algorithm, we present in Section 4 optimal $O(n \log n)$ coloring and clique algorithms for multitolerance graphs. In Section 5 we present an $O(m + n \log n)$ algorithm that computes a maximum weight independent set. In Section 6 we classify multitolerance graphs in the hierarchy of perfect graphs of [11]. Finally, we discuss the presented results and further research in Section 7.

2 An intersection model for multitolerance graphs

In this section we present a three-dimensional intersection model for general multitolerance graphs, which unifies the intersection model of trapezoids in the plane for bounded multitolerance graphs [11] and that of parallelepipeds in the 3-dimensional space for tolerance graphs [22]. Given a multitolerance graph $G = (V, E)$ along with a multitolerance representation $\langle I, t \rangle$ of G, recall that vertex $v \in V$ corresponds to an interval $I_v = [l_v, r_v]$ on the real line and a set τ_v of tolerance-intervals, where either $\tau_v = \mathcal{I}(I_v, l_{t_v}, r_{t_v})$ for some values $l_{t_v}, r_{t_v} \in I_v$ of tolerant points, or $\tau_v = \{\mathbb{R}\}$.

Definition 1. *Given a multitolerance representation of a multitolerance graph $G = (V, E)$, vertex $v \in V$ is* bounded *if $\tau_v = \mathcal{I}(I_v, l_{t_v}, r_{t_v})$ for some values $l_{t_v}, r_{t_v} \in I_v$. Otherwise, v is* unbounded. *V_B and V_U are the sets of bounded and unbounded vertices in V, respectively. Clearly $V = V_B \cup V_U$.*

Definition 2. *For a vertex $v \in V_B$ (resp. $v \in V_U$) in a multitolerance representation of G, the values $t_{v,1} = l_{t_v} - l_v$ and $t_{v,2} = r_v - r_{t_v}$ (resp. $t_{v,1} = t_{v,2} = \infty$) are the* left tolerance *and the* right tolerance *of v, respectively. Moreover, if $v \in V_U$, then $t_v = \infty$ is the* tolerance *of v.*

It can be now easily seen by Definition 2 that if we set $t_{v,1} = t_{v,2}$ for every vertex $v \in V$, then we obtain a tolerance representation, in which $t_{v,1} = t_{v,2}$ is the (unique) tolerance of v. We may assume w.l.o.g. that no two bounded vertices share an endpoint or tolerant point, i.e. $\{l_u, r_u, l_{t_u}, r_{t_u}\} \cap \{l_v, r_v, l_{t_v}, r_{t_v}\}$ for all $u, v \in V_B$ with $u \neq v$ [25]. Furthermore, by possibly performing a small shift of the endpoints and the tolerant points, we may assume w.l.o.g. that

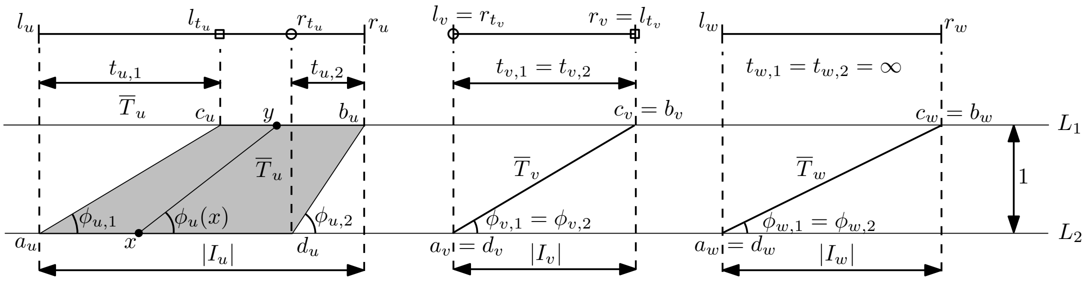


Fig. 1. Trapezoids $\overline{T}_u$ and $\overline{T}_v$ correspond to bounded vertices u and v, respectively, while $\overline{T}_w$ corresponds to an unbounded vertex w.

$t_{v,1}, t_{v,2} > 0$ for every $v \in V$ and that the left and right tolerances for every bounded vertex are distinct, i.e. $\{t_{u,1}, t_{u,2}\} \cap \{t_{v,1}, t_{v,2}\} = \emptyset$ for all $u, v \in V_B$ with $u \neq v$. Similarly, if $t_{v,1} \neq |I_v|$ (resp. $t_{v,2} \neq |I_v|$) for a bounded vertex $v \in V_B$, we may assume w.l.o.g. that also $t_{v,2} \neq |I_v|$ (resp. $t_{v,1} \neq |I_v|$). That is, for every $v \in V_B$, either $t_{v,1} = t_{v,2} = |I_v|$, or $t_{v,1} < |I_v|$ and $t_{v,2} < |I_v|$. For more details in the cases of tolerance and bounded multitolerance graphs we refer to [11,16].

Let now L_1 and L_2 be two parallel lines at unit distance in the Euclidean plane. In the following we define for every vertex $v \in V$ a trapezoid $\overline{T}_v$ in the plane between the lines L_1 and L_2. The values $\tan\phi$ and $\cot\phi = \frac{1}{\tan\phi}$ denote the tangent and the cotangent of a slope ϕ, respectively. Furthermore, $\phi = arc\cot x$ is the slope ϕ, for which $\cot\phi = x$.

Definition 3. *Given an interval $I_v = [l_v, r_v]$ and tolerances $t_{v,1}, t_{v,2}$, $\overline{T}_v$ is the trapezoid in $\mathbb{R}^2$ defined by the points c_v, b_v on L_1 and a_v, d_v on L_2, where $a_v = l_v$, $b_v = r_v$, $c_v = \min\{r_v, l_v + t_{v,1}\}$, and $d_v = \max\{l_v, r_v - t_{v,2}\}$. The values $\phi_{v,1} = arc\cot(c_v - a_v)$ and $\phi_{v,2} = arc\cot(b_v - d_v)$ are the* left slope *and the* right slope *of $\overline{T}_v$. Moreover, for every unbounded vertex $v \in V_U$, $\phi_v = \phi_{v,1} = \phi_{v,2}$ is the* slope *of $\overline{T}_v$.*

An example is depicted in Figure 1, where $\overline{T}_u$ and $\overline{T}_v$ correspond to bounded vertices u and v, and $\overline{T}_w$ corresponds to an unbounded vertex w. For each of these trapezoids, the corresponding interval (together with the associated tolerant points, if the vertex is bounded) is drawn above the trapezoid for better visibility. The left (resp. right) tolerant points are depicted by a square (resp. cycle). Observe that when a vertex v is bounded, the values c_v and d_v coincide with the tolerant points l_{t_v} and r_{t_v}, respectively, while $\phi_{v,1} = arc\cot t_{v,1}$ and $\phi_{v,2} = arc\cot t_{v,2}$. On the other hand, when a vertex v is unbounded, the values c_v and d_v coincide with the endpoints b_v and a_v of I_v, respectively, while $\phi_{v,1} = \phi_{v,2} = arc\cot|I_v|$. Observe also that in both cases where $t_{v,1} = t_{v,2} = |I_v|$ and $t_{v,1} = t_{v,2} = \infty$, the trapezoid $\overline{T}_v$ is reduced to a line segment (cf. $\overline{T}_v$ and $\overline{T}_w$ in Figure 1). Furthermore, similarly to the above, we can assume w.l.o.g. that all endpoints and slopes of the trapezoids are distinct, i.e. $\{a_u, b_u, c_u, d_u\} \cap \{a_v, b_v, c_v, d_v\} = \emptyset$ and $\{\phi_{u,1}, \phi_{u,2}\} \cap \{\phi_{v,1}, \phi_{v,2}\} = \emptyset$ for every $u, v \in V$ with $u \neq v$. Since $|I_v| > 0$ and $t_{v,1}, t_{v,2} > 0$ for every vertex v, it follows that $0 < \phi_{v,1} < \frac{\pi}{2}$ and $0 < \phi_{v,2} < \frac{\pi}{2}$ for all slopes $\phi_{v,1}, \phi_{v,2}$.

Definition 4. *Let $u \in V_B$ be a bounded vertex in a multitolerance representation and a_u, b_u, c_u, d_u be the endpoints of the trapezoid $\overline{T}_u$. Let $x \in [a_u, d_u]$ and $y \in [c_u, b_u]$ be two points on the lines L_2 and L_1, respectively, such that $x = \lambda a_u + (1-\lambda) d_u$ and $y = \lambda c_u + (1-\lambda) b_u$ for the same value $\lambda \in [0,1]$. Then $\phi_u(x)$ is the slope of the line segment with endpoints x and y on the lines L_2 and L_1, respectively.*

In the example of Figure 1, two points $x \in [a_u, d_u]$ and $y \in [c_u, b_u]$ are depicted on the lines L_2 and L_1, respectively, such that $x = \lambda a_u + (1-\lambda) d_u$ and $y = \lambda c_u + (1-\lambda) b_u$ for the same value $\lambda \in [0,1]$. Then, the interval $[x, y]$ on the real line, with values x and y as endpoints, coincides with the tolerance-interval $I_{l_{t_u},r_{t_u}}(1-\lambda) = [l_u + (r_{t_u} - l_u)(1-\lambda), l_{t_u} + (r_u - l_{t_u})(1-\lambda)]$ of $\mathcal{I}(I_u, l_{t_u}, r_{t_u})$ (cf. the definition of a multitolerance representation). Furthermore, for the slope $\phi_u(x)$, as defined in Definition 4 (cf. Figure 1), it follows that $\cot\phi_u(x) = y - x = \lambda(c_u - a_u) + (1-\lambda)(b_u - d_u)$. Therefore, since $\cot\phi_{u,1} = c_u - a_u$ and $\cot\phi_{u,2} = b_u - d_u$, the next observation follows.

Observation 1 *Let $x = \lambda a_u + (1-\lambda)d_u$ for a bounded vertex $u \in V_B$ and some value $\lambda \in [0,1]$. Then,* $\cot\phi_u(x) = \lambda\cot\phi_{u,1} + (1-\lambda)\cot\phi_{u,2}$.

Note that, in Definition 3, the endpoints a_v, b_v, c_v, d_v of any trapezoid $\overline{T}_v$ (on the lines L_1 and L_2) lie on the plane $z = 0$ in $\mathbb{R}^3$. Therefore, since we assumed that the distance between the lines L_1 and L_2 is one, these endpoints of $\overline{T}_v$ correspond to the points $(a_v, 0, 0)$, $(b_v, 1, 0)$, $(c_v, 1, 0)$, and $(d_v, 0, 0)$ in $\mathbb{R}^3$, respectively. For the sake of presentation, we may not distinguish in the following between these points in $\mathbb{R}^3$ and the corresponding real values a_v, b_v, c_v, d_v, whenever this slight abuse of notation does not cause any confusion.

For a set X of points in $\mathbb{R}^3$, denote by $H_{\text{convex}}(X)$ the *convex hull* defined by the points of X. That is, $\overline{T}_v = H_{\text{convex}}(a_v, b_v, c_v, d_v)$ for every vertex $v \in V$ by Definition 3, where a_v, b_v, c_v, d_v are points of the plane $z = 0$ in $\mathbb{R}^3$. We are now ready to give the main definition of this article.

Definition 5. *Let $G = (V, E)$ be a multitolerance graph with a multitolerance representation $\{I_v = [a_v, b_v], \tau_v \mid v \in V\}$ and $\varDelta = \max\{b_v \mid v \in V\} - \min\{a_v \mid v \in V\}$ be the greatest distance between two interval endpoints. For every vertex $v \in V$, the* trapezoepiped T_v *of v is the convex set of points in $\mathbb{R}^3$ defined as follows:*

(a) if $t_{v,1}, t_{v,2} \leq |I_v|$ (that is, v is bounded), then $T_v = H_{convex}(\overline{T}_v, a'_v, b'_v, c'_v, d'_v)$,
(b) if $t_v = t_{v,1} = t_{v,2} = \infty$ (that is, v is unbounded), then $T_v = H_{convex}(a'_v, c'_v)$,

where $a'_v = (a_v, 0, \varDelta - \cot\phi_{v,1})$, $b'_v = (b_v, 1, \varDelta - \cot\phi_{v,2})$, $c'_v = (c_v, 1, \varDelta - \cot\phi_{v,1})$, *and* $d'_v = (d_v, 0, \varDelta - \cot\phi_{v,2})$. *The set of trapezoepipeds $\{T_v \mid v \in V\}$ is a* trapezoepiped representation *of G.*

Note by the definition of $\varDelta$ that $\varDelta - \cot\phi_{v,1} \geq 0$ and $\varDelta - \cot\phi_{v,2} \geq 0$ for every $v \in V$. Furthermore, observe that for each interval I_v, the trapezoid $\overline{T}_v$ of Definition 3 (see also Figure 1) coincides with the projection of the trapezoepiped T_v on the plane $z = 0$. An example of this construction is given in Figure 2. A multitolerance graph G with seven vertices $\{v_1, v_2, \ldots, v_7\}$ is depicted in Figure 2(a), while the trapezoepiped representation of G is illustrated in Figure 2(b). The set of bounded and unbounded vertices in this representation are $V_B = \{v_3, v_4, v_6, v_7\}$ and $V_U = \{v_1, v_2, v_5\}$, respectively. We illustrate the endpoints $a_{v_i}, b_{v_i}, c_{v_i}, d_{v_i}$ and $a'_{v_i}, b'_{v_i}, c'_{v_i}, d'_{v_i}$ of T_{v_i}, as well as the relationship between the interval I_{v_i} and the corresponding trapezoepiped T_{v_i} for one unbounded and one bounded vertex, cf. v_1 and v_6, respectively. Note that $a_{v_1} = d_{v_1}$, $a'_{v_1} = d'_{v_1}$, $c_{v_1} = b_{v_1}$, and $c'_{v_1} = b'_{v_1}$, since v_1 is unbounded. In the case where $t_{v_i,1}, t_{v_i,2} < |I_{v_i}|$, the trapezoepiped T_{v_i} is three-dimensional, cf. T_{v_3}, T_{v_4}, and T_{v_6}, while in the border case where $t_{v_i,1} = t_{v_i,2} = |I_{v_i}|$ it degenerates to a two-dimensional rectangle, cf. T_{v_7}. In these two cases, each T_{v_i} corresponds to a bounded vertex v_i. In the remaining case where v_i is unbounded, i.e. $t_{v_i} = t_{v_i,1} = t_{v_i,2} = \infty$, the trapezoepiped T_{v_i} degenerates to an one-dimensional line segment above plane $z = 0$, cf. T_{v_1}, T_{v_2}, and T_{v_5}.

We now prove that the trapezoepiped representation forms a three-dimensional intersection model for the class of multitolerance graphs (namely, that every multitolerance graph G can be viewed as the intersection graph of the corresponding trapezoepipeds T_v). To this end, we first prove the next two lemmas.

Lemma 1. *Let $u \in V$ and $v \in V_U$ in a trapezoepiped representation of a multitolerance graph $G = (V, E)$. If $c_v < c_u$ on the line L_1, then $T_u \cap T_v = \emptyset$.*

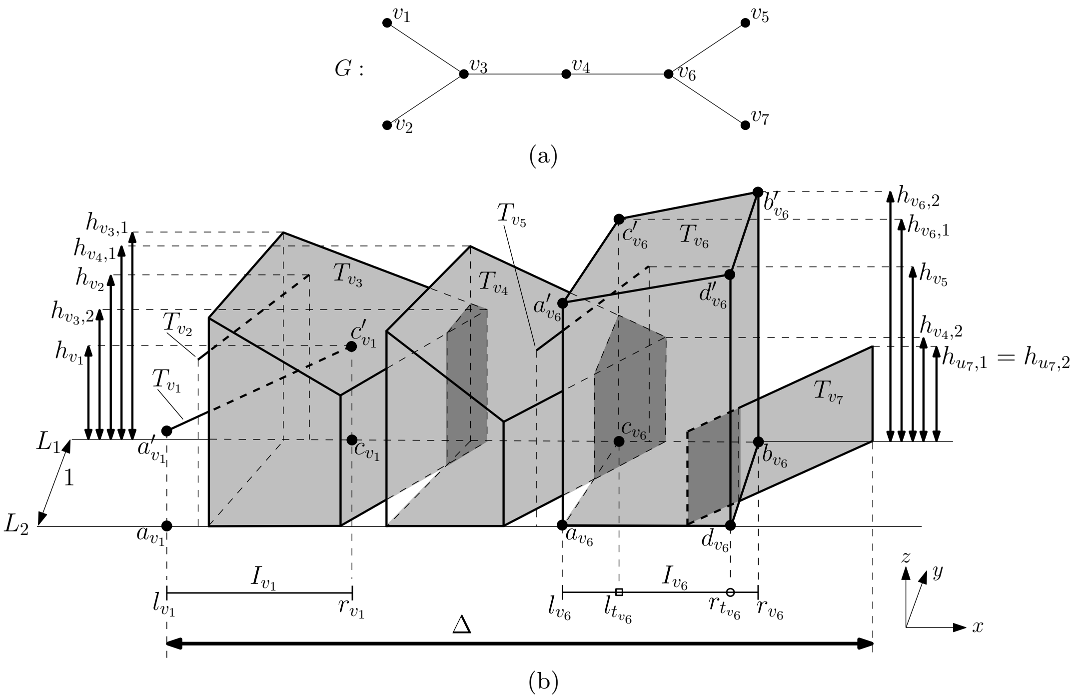


Fig. 2. (a) A multitolerance graph G and (b) a trapezoepiped representation R of G. Here, $h_{v_i,j} = \Delta - \cot\phi_{v_i,j}$ for every bounded vertex $v_i \in V_B$ and $j \in \{1,2\}$, while $h_{v_i} = \Delta - \cot\phi_{v_i}$ for every unbounded vertex $v_i \in V_U$.

Proof. Let first $u \in V_U$, i.e. u is unbounded. Then T_u and T_v are both line segments on the disjoint planes $z = \Delta - \cot\phi_u$ and $z = \Delta - \cot\phi_v$ of $\mathbb{R}^3$, and thus $T_u \cap T_v = \emptyset$. Let now $u \in V_B$, i.e. u is bounded. If $\overline{T}_u \cap \overline{T}_v = \emptyset$, then also $T_u \cap T_v = \emptyset$ by Definition 5. Suppose that $\overline{T}_u \cap \overline{T}_v \neq \emptyset$. Then, since we assumed that $c_v < c_u$, it follows that $a_u < a_v$. Therefore in particular $\phi_v > \phi_{u,1}$, and thus also $\tan\phi_v > \tan\phi_{u,1}$, i.e. $\Delta - \cot\phi_v > \Delta - \cot\phi_{v,1}$. Suppose first that $d_u < a_v$, i.e. $a_u < d_u < a_v$. Then also $\phi_v > \phi_{u,2}$, i.e. $\Delta - \cot\phi_v > \Delta - \cot\phi_{v,1}$ and $\Delta - \cot\phi_v > \Delta - \cot\phi_{v,2}$. Therefore the line T_v lies completely above the trapezoepiped T_u in $\mathbb{R}^3$, and thus $T_u \cap T_v = \emptyset$. Suppose now that $a_v < d_u$, i.e. $a_u < a_v < d_u$. Consider any point $x \in [a_u, a_v] \subseteq [a_u, d_u]$ on the line L_2. Then, $x = \lambda a_u + (1-\lambda)d_u$ for some value $\lambda \in [0,1]$. Consider also the point $y \in [c_u, b_u]$ on the line L_1, such that $y = \lambda c_u + (1-\lambda)b_u$ for the same value of λ. Then, the maximum height z of the trapezoepiped T_u above the point x is $\lambda(\Delta - \cot\phi_{u,1}) + (1-\lambda)(\Delta - \cot\phi_{u,2})$, which is equal to $\Delta - \cot\phi_u(x)$ by Observation 1. If $T_u \cap T_v \neq \emptyset$, then there exists such a point $x \in [a_v, d_v]$ on the line L_2 (and the corresponding point $y \in [c_v, b_u]$ on the line L_1), such that $\Delta - \cot\phi_u(x) > \Delta - \cot\phi_v$, i.e. $\phi_u(x) > \phi_v$. However, $x \leq a_v$ and $c_v < c_u \leq y$, and thus $\phi_u(x) < \phi_v$, which is a contradiction. Therefore $T_u \cap T_v = \emptyset$. $\square$

Lemma 2. *Let $u \in V_B$ and $v \in V_U$ in a trapezoepiped representation of a multitolerance graph $G = (V,E)$. Let a_u, d_u, and $a_v = d_v$ be the endpoints of $\overline{T}_u$ and $\overline{T}_v$, respectively, on the line L_2. If $a_v < a_u$, then $T_u \cap T_v \neq \emptyset$ if and only if $\overline{T}_u \cap \overline{T}_v \neq \emptyset$. If $d_u < a_v$, then $T_u \cap T_v = \emptyset$. Finally, if $a_u < a_v < d_u$, then $T_u \cap T_v \neq \emptyset$ if and only if $\phi_v \leq \phi_u(a_v)$.*

Proof. Recall first that $a_v = d_v$ and $c_v = b_v$, since v is unbounded. Let $a_v < a_u$. If $\overline{T}_u \cap \overline{T}_v = \emptyset$, then also $T_u \cap T_v = \emptyset$ by the definition of a trapezoepiped representation (cf. Definition 5). If $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, then $c_v > c_u$, since we assumed that $a_v < a_u$. Therefore, in particular $\phi_v < \phi_{u,1}$, and thus also $\tan\phi_v < \tan\phi_{u,1}$, i.e. $\Delta - \cot\phi_v < \Delta - \cot\phi_{u,1}$. Therefore the line T_v intersects the trapezoepiped T_u in R, i.e. $T_u \cap T_v \neq \emptyset$.

Let now $d_u < a_v$. If $\overline{T}_u \cap \overline{T}_v = \emptyset$, then also $T_u \cap T_v = \emptyset$ by Definition 5. Suppose that $\overline{T}_u \cap \overline{T}_v \neq \emptyset$. Then, since we assumed that $d_u < a_v$, it follows that $c_v < b_u$. Therefore in particular $\phi_v > \phi_{u,2}$, i.e. $\Delta - \cot\phi_v > \Delta - \cot\phi_{u,2}$. Suppose first that $c_v < c_u$, i.e. $c_v < c_u < b_u$. Then also $\phi_v > \phi_{u,1}$, i.e. $\Delta - \cot\phi_v > \Delta - \cot\phi_{u,1}$ and $\Delta - \cot\phi_v > \Delta - \cot\phi_{u,2}$. Therefore the whole trapezoepiped T_u lies completely below the plane $z = \Delta - \cot\phi_v$ of $\mathbb{R}^3$, on which the line T_v lies, and thus $T_u \cap T_v = \emptyset$. Suppose now that $c_u < c_v$, i.e. $c_u < c_v < b_u$. Consider any point $y \in [c_v, b_u] \subseteq [c_u, b_u]$ on the line L_1. Then, $y = \lambda c_u + (1-\lambda)b_u$ for some value $\lambda \in [0,1]$. Consider also the point $x \in [a_u, d_u]$ on the line L_2, such that $x = \lambda a_u + (1-\lambda)d_u$ for the same value of λ. Then, the maximum height z of the trapezoepiped T_u above the point y is $\lambda(\Delta - \cot\phi_{u,1}) + (1-\lambda)(\Delta - \cot\phi_{u,2})$, which is equal to $\Delta - \cot\phi_u(x)$ by Observation 1. If $T_u \cap T_v \neq \emptyset$, then there exists such a point $y \in [c_v, b_u]$ on the line L_1 (and the corresponding point $x \in [a_u, d_u]$ on the line L_2), such that $\Delta - \cot\phi_u(x) > \Delta - \cot\phi_v$, i.e. $\phi_u(x) > \phi_v$. However, $x \leq d_u < a_v$ and $c_v \leq y$, and thus $\phi_u(x) < \phi_v$, which is a contradiction. Therefore $T_u \cap T_v = \emptyset$.

Let finally $a_u < a_v < d_u$. Then, $a_v = \lambda a_u + (1-\lambda)d_u$ for some value $\lambda \in [0,1]$. Consider the point $y \in [c_u, b_u]$ on the line L_1, such that $y = \lambda c_u + (1-\lambda)b_u$ for the same value of λ. Since $v \in V_U$, the line T_v lies on the plane $z = \Delta - \cot\phi_v$ of $\mathbb{R}^3$. Suppose first that $\phi_v \leq \phi_u(a_v)$, and thus in particular $y \leq c_v$. Then the maximum height of the trapezoepiped T_u above the point a_v of L_2 is $\Delta - \cot\phi_u(a_v) \geq \Delta - \cot\phi_v$. Therefore $T_u \cap T_v \neq \emptyset$ if $\phi_v \leq \phi_u(a_v)$. Suppose now that $\phi_v > \phi_u(a_v)$, and thus in particular $c_v < y$. Then the maximum height of the trapezoepiped T_u above the point a_v of L_2 is $\Delta - \cot\phi_u(a_v) < \Delta - \cot\phi_v$. If $c_v < c_u$, then $T_u \cap T_v = \emptyset$ by Lemma 1. Suppose now that $c_u < c_v$, i.e. $c_u < c_v < y < b_u$. Then $c_v = \lambda' c_u + (1-\lambda')b_u$ for some value $\lambda' \in [0,1]$. Note that $\lambda' > \lambda$, since $c_v < y = \lambda c_u + (1-\lambda)b_u$. Consider the point $x \in [a_u, d_u]$ on the line L_2, such that $x = \lambda' a_u + (1-\lambda')d_u$ for the same value $\lambda' > \lambda$. Then the maximum height of the trapezoepiped T_u above the points c_v and x is $\lambda'(\Delta - \cot\phi_{u,1}) + (1-\lambda')(\Delta - \cot\phi_{u,2})$, which is equal to $\Delta - \cot\phi_u(x)$ by Observation 1. Furthermore, since $\lambda' > \lambda$, it follows that $x < a_v$. Thus, since also $c_v < y$, it follows that $\phi_u(x) < \phi_v$, i.e. $\Delta - \cot\phi_u(x) < \Delta - \cot\phi_v$. Summarizing, the maximum height of the trapezoepiped T_u above the point a_v of L_2 (resp. above the point c_v of L_1) is $\Delta - \cot\phi_u(a_v) < \Delta - \cot\phi_v$ (resp. $\Delta - \cot\phi_u(x) < \Delta - \cot\phi_v$). Therefore, since T_u is convex, it follows that the line T_v lies above the trapezoepiped T_u in $\mathbb{R}^3$. Therefore $T_u \cap T_v = \emptyset$ if $\phi_v > \phi_u(a_v)$. This completes the proof of the lemma. □

Theorem 1. *Let $G = (V, E)$ be a multitolerance graph with a multitolerance representation $\{I_v = [a_v, b_v], \tau_v \mid v \in V\}$. Then for every $u, v \in V$, $uv \in E$ if and only if $T_u \cap T_v \neq \emptyset$.*

Proof. Let a_u, b_u, c_u, d_u and a_v, b_v, c_v, d_v be the endpoints of the trapezoids $\overline{T}_u$ and $\overline{T}_v$ that correspond to vertices u and v, respectively (cf. Figure 1). Note that $I_u = [a_u, b_u]$ and $I_v = [a_v, b_v]$ for the corresponding intervals in the multitolerance representation. W.l.o.g. we may assume that $a_u < a_v$. Suppose first that both u and v are unbounded. Then T_u and T_v are both line segments on the disjoint planes $z = \Delta - \cot\phi_u$ and $z = \Delta - \cot\phi_v$ of $\mathbb{R}^3$, and thus $T_u \cap T_v = \emptyset$. On the other hand, $uv \notin E$, since no two unbounded vertices are adjacent in a multitolerance graph G. Suppose in the following of the proof that $u \in V_B$ or $v \in V_B$ (or both).

Suppose that $\overline{T}_u \cap \overline{T}_v = \emptyset$, i.e. $d_u < a_v$ and $b_u < c_v$. Then also $T_u \cap T_v = \emptyset$. On the other hand, let $Q_u = [p, q] \in \tau_u$ (resp. $Q_v = [p, q] \in \tau_v$) be a (non-infinite) tolerance-interval of vertex u, if $u \in V_B$ (resp. of vertex v, if $v \in V_B$). Then, $p \leq d_u < a_v$ (resp. $b_u < c_v \leq q$), and thus $Q_u \nsubseteq I_v = [a_v, b_v]$ (resp. $Q_v \nsubseteq I_u = [a_u, b_u]$). Therefore $uv \notin E$.

Suppose in the following that $\overline{T}_u \cap \overline{T}_v \neq \emptyset$. We distinguish now two cases according to the relative positions of the points d_u and a_v on the line L_2.

Case 1. $d_u < a_v$. Then, since we assumed that $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, it follows that $c_v < b_u$. Therefore in particular $\phi_{v,1} > \phi_{u,2}$, and thus also $\tan\phi_{v,1} > \tan\phi_{u,2}$, i.e. $\Delta - \cot\phi_{v,1} > \Delta - \cot\phi_{v,2}$.

Case 1a. Suppose first that $v \in V_B$. Then, necessarily $T_u \cap T_v \neq \emptyset$ by the definition of a trapezoepiped representation, since $\overline{T}_u \cap \overline{T}_v \neq \emptyset$ and $\Delta - \cot\phi_{v,1} > \Delta - \cot\phi_{v,2}$. On the other hand, consider the tolerance-interval $Q_v = [a_v, c_v] \in \tau_v$ (Q_u exists, since $v \in V_B$). Then $Q_v \subseteq [a_u, b_u] = I_u$, since $a_u \leq d_u < a_v < c_v < b_u$, and thus $uv \in E$. That is, $T_u \cap T_v \neq \emptyset$ and $uv \in E$.

Case 1b. Suppose now that $v \in V_U$, and thus $u \in V_B$. Then $\tau_v = \{\mathbb{R}\}$, and thus $Q_v \nsubseteq I_u$ for any $Q_v \in \tau_v$. Consider any tolerance-interval $Q_u = [p, q] \in \tau_u$. Then $p \leq d_u < a_v$, and thus $Q_u \nsubseteq I_v = [a_v, b_v]$. Therefore $uv \notin E$. On the other hand, $T_u \cap T_v = \emptyset$ by Lemma 2, since $d_u < a_v$. That is, $T_u \cap T_v \neq \emptyset$ and $uv \in E$.

Case 2. $a_v < d_u$. That is, $a_u < a_v < d_u$, and thus in particular $u \in V_B$ (since $a_u \neq d_u$).

Case 2a. Suppose first that $v \in V_B$. Then, necessarily $T_u \cap T_v \neq \emptyset$ by the definition of a trapezoepiped representation, since both $u, v \in V_B$ and $\overline{T}_u \cap \overline{T}_v \neq \emptyset$. We will now prove that also $uv \in E$. Let first $c_v < b_u$ on the line L_1 and consider the tolerance-interval $Q_v = [a_v, c_v] \in \tau_v$. Then $Q_v \subseteq I_u = [a_u, b_u]$, since $a_u < a_v < c_v < b_u$, and thus $uv \in E$. Let now $b_u < c_v$ on the line L_1 and consider the tolerance-interval $Q_u = [d_u, b_u] \in \tau_u$. Then $Q_u \subseteq I_v = [a_v, b_v]$, since $a_v < d_u < b_u < c_v$, and thus $uv \in E$. That is, $T_u \cap T_v \neq \emptyset$ and $uv \in E$.

Case 2b. Suppose now that $v \in V_U$. Then, $a_v = \lambda a_u + (1-\lambda) d_u$ for some value $\lambda \in [0, 1]$, since $a_u < a_v < d_u$. Consider the point $y \in [c_u, b_u]$ on the line L_1, such that $y = \lambda c_u + (1-\lambda) b_u$ for the same value of λ. Since $v \in V_U$, the line T_v lies on the plane $z = \Delta - \cot\phi_v$ of $\mathbb{R}^3$.

Let first $\phi_v \leq \phi_u(a_v)$, and thus in particular $y \leq c_v$. Then $T_u \cap T_v \neq \emptyset$ by Lemma 2. On the other hand, consider the tolerance-interval $Q_u = [a_v, y] \in \tau_u$. Then, $Q_u \subseteq I_v = [a_v, b_v]$, since $a_v < y \leq c_v = b_v$, and thus $uv \in E$. That is, $T_u \cap T_v \neq \emptyset$ and $uv \in E$, if $\phi_v \leq \phi_u(a_v)$.

Let now $\phi_v > \phi_u(a_v)$, and thus in particular $y > c_v$. Then $T_u \cap T_v = \emptyset$ by Lemma 2. On the other hand, consider any tolerance-interval $Q_u = [p, q] \in \tau_u$. Then, $p = \lambda' a_u + (1-\lambda') d_u$ and $q = \lambda' c_u + (1-\lambda') b_u$, for some value $\lambda' \in [0, 1]$. If $\lambda' > \lambda$, then $p < a_v$, and thus $Q_u = [p, q] \nsubseteq [a_v, b_v] = I_v = [a_v, b_v]$. If $\lambda' \leq \lambda$, then $q \geq y > c_v = b_v$, and thus again $Q_u = [p, q] \nsubseteq I_v = [a_v, b_v]$. Therefore $Q_u \nsubseteq I_v$ for every $Q_u \in \tau_u$, and thus $uv \notin E$. That is, $T_u \cap T_v = \emptyset$ and $uv \notin E$, if $\phi_v > \phi_u(a_v)$. □

Clearly, for each $v \in V$ the trapezoepiped T_v can be constructed in constant time; therefore the next lemma follows directly.

Lemma 3. *Given a multitolerance representation of a multitolerance graph G with n vertices, a trapezoepiped representation of G can be constructed in $O(n)$ time.*

3 A canonical representation of multitolerance graphs

In this section we introduce a canonical representation of multitolerance graphs, which is a special kind of a trapezoepiped representation. Moreover, we present an efficient algorithm that constructs in $O(n \log n)$ time a canonical representation of a multitolerance graph G with n vertices, given any trapezoepiped representation of G. This algorithm proves to be useful for designing efficient algorithms on multitolerance graphs for the minimum coloring and the maximum clique problems with optimal running time $O(n \log n)$, as we will present in Section 4. First, we state the following definition, similarly to the case of tolerance graphs [22] (see also [10, 11]).

Definition 6. *An unbounded vertex $v \in V_U$ of a multitolerance graph G is called* inevitable *(for a certain trapezoepiped representation), if replacing T_v by $H_{convex}(\overline{T}_v, a'_v, c'_v)$ creates a new edge uv in G; then u is a* hovering vertex *of v and the set $H(v)$ of all hovering vertices of v is the* hovering set *of v. Otherwise, v is called* evitable.

Recall that $a'_v = d'_v$ and $c'_v = b'_v$ for every unbounded vertex $v \in V_U$, and thus $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v) = H_{\text{convex}}(\overline{T}_v, a'_v, b'_v, c'_v, d'_v)$ in Definition 6. Therefore, replacing T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in the trapezoepiped representation of G is equivalent with replacing in the corresponding multitolerance representation of G the infinite tolerance $t_v = \infty$ by the finite tolerances $t_{v,1} = t_{v,2} = |I_v|$, i.e. with making v a bounded vertex. Note that the hovering set of an inevitable unbounded vertex v can have more than one elements, since the replacement of T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ may create more than one new edges in G. Furthermore, $uv \notin E$ for every hovering vertex u of v, while u can be both bounded or unbounded.

Lemma 4. *Let $G = (V, E)$ be a multitolerance graph, R be a trapezoepiped representation of G, and $v \in V_U$ be an inevitable unbounded vertex in R. Then, $N(v) \subseteq N(u)$ for every hovering vertex u of v.*

Proof. Note first that $a_v = d_v$ and $c_v = b_v$, since v is unbounded. Let u be a hovering vertex of v in R, i.e. $uv \notin E$ and replacing T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in R creates the new edge uv in G by Definition 6. Thus, in particular $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, while the line T_v lies above T_u in R. Consider a vertex $w \in N(v)$. Then w is bounded, since v is unbounded and no two unbounded vertices are adjacent. Furthermore $T_v \cap T_w \neq \emptyset$ and $\overline{T}_v \cap \overline{T}_w \neq \emptyset$, since $w \in N(v)$. We will prove that also $w \in N(u)$, in both cases where u is bounded and unbounded.

Case 1. u is bounded. If $d_w < a_v$, then $T_v \cap T_w = \emptyset$ by Lemma 2, which is a contradiction. Thus $a_v < d_w$. Suppose that $a_v < a_u$. Then, since $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, Lemma 2 implies that $T_u \cap T_v \neq \emptyset$, and thus $uv \in E$, which is a contradiction. Thus $a_u < a_v$, i.e. $a_u < a_v < d_w$.

Since both u and w are bounded, $T_u \cap T_w \neq \emptyset$ if and only if $\overline{T}_u \cap \overline{T}_w \neq \emptyset$ (cf. Definition 5). Therefore, in order to prove that $w \in N(u)$, it suffices to prove that $\overline{T}_u \cap \overline{T}_w \neq \emptyset$, i.e. that the corresponding trapezoids in the plane intersect. Suppose otherwise that $\overline{T}_u \cap \overline{T}_w = \emptyset$. Then, since $a_u < d_w$, it follows that trapezoid $\overline{T}_u$ lies completely to the left of the trapezoid $\overline{T}_w$. Therefore, since both $\overline{T}_v \cap \overline{T}_u \neq \emptyset$ and $\overline{T}_v \cap \overline{T}_w \neq \emptyset$, it follows that either $d_u < a_w < a_v$ and $c_v < b_u < c_w$, or $a_v < d_u < a_w$ and $b_u < c_w < c_v$.

Case 1a. $d_u < a_w < a_v$ and $c_v < b_u < c_w$. Recall now that $a_v < d_w$, i.e. $a_w < a_v < d_w$. Furthermore, note that $c_v < c_w \leq y$ for every point $y \in [c_w, b_w]$ on the line L_1. Therefore $\phi_v > \phi_w(a_v)$, and thus $T_v \cap T_w = \emptyset$ by Lemma 2, i.e. $w \notin N(v)$, which is a contradiction.

Case 1b. $a_v < d_u < a_w$ and $b_u < c_w < c_v$. Consider the tolerance-interval $Q_u = [d_u, b_u] \in \tau_u$. Then $Q_u \subseteq I_v = [a_v, b_v]$, since $a_v < d_u < b_u < c_v = b_v$. Therefore $uv \in E$, which is a contradiction.

Therefore $\overline{T}_u \cap \overline{T}_w \neq \emptyset$, and thus also $T_u \cap T_w \neq \emptyset$, i.e. $w \in N(u)$, since both u and w are bounded.

Case 2. u is unbounded. Then $a_u = d_u$ and $c_u = b_u$. Furthermore $\Delta - \cot \phi_u < \Delta - \cot \phi_v$, since the line T_v lies above T_u in R, and thus $\phi_u < \phi_v$. Therefore $a_u < a_v$ and $c_u > c_v$. If $d_w < a_v$, then $T_v \cap T_w = \emptyset$ by Lemma 2, which is a contradiction, since $w \in N(v)$. Suppose that $a_v < a_w$. Then $c_v > c_w$, since $\overline{T}_v \cap \overline{T}_w \neq \emptyset$. That is, $a_u < a_v < a_w$ and $c_u > c_v > c_w$, and thus in particular $\overline{T}_u \cap \overline{T}_w \neq \emptyset$. Then $T_u \cap T_w \neq \emptyset$ by Lemma 2, i.e. $w \in N(u)$. Suppose now that $a_w < a_v < d_w$, i.e. $a_v = \lambda a_w + (1 - \lambda) d_w$ for some value $\lambda \in [0, 1]$. Consider the point $y \in [c_w, b_w]$ on the line L_1, such that $y = \lambda c_w + (1 - \lambda) b_w$ for the same value of λ. Since $T_v \cap T_w \neq \emptyset$, Lemma 2 implies that $\phi_v \leq \phi_w(a_v)$, and thus $y < c_v$. That is, $a_u < a_v$ and $y < c_v < c_u$. Consider now the tolerance-interval $Q_w = [a_v, y] \in \tau_w$. Then $Q_w \subseteq I_u = [a_u, b_u]$, since $a_u < a_v < y < c_u = b_u$. Therefore $w \in N(u)$ in the case where u is unbounded. This completes the proof of the lemma. □

In the next definition we introduce the notion of a canonical representation of a multitolerance graph G.

Definition 7. *A trapezoepiped representation of a multitolerance graph G is called* canonical *if every unbounded vertex is inevitable.*

For example, in the multitolerance graph depicted in Figure 2, v_2 and v_5 are inevitable unbounded vertices, v_1 and v_4 are hovering vertices of v_2 and v_5, respectively, while v_1 is an evitable unbounded vertex. Therefore, this representation is not canonical for the graph G. However, if we replace T_{v_1} by $H_{\text{convex}}(\overline{T}_{v_1}, a'_{v_1}, c'_{v_1})$, we get a canonical representation for G. The next lemma follows now by Definitions 6 and 7 and by Lemma 4.

Lemma 5. *Let R be a canonical representation of a multitolerance graph G and $v \in V_U$ be an inevitable unbounded vertex of multitolerance graph G (in the representation R). Then, there exists a hovering vertex u of v, which is bounded.*

Proof. Since R is a canonical representation of G, all unbounded vertices of G in R are inevitable unbounded by Definition 7. Suppose that there exists an unbounded vertex v, such that every hovering vertex of v is unbounded, and let v be the vertex with the smallest slope ϕ_v among them. Let u be a hovering vertex of v in R, i.e. replacing T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in R creates the new edge uv in G by Definition 6. Thus, in particular $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, while the line T_v lies above T_u in R. Therefore, $\Delta - \cot\phi_v > \Delta - \cot\phi_u$, and thus $\phi_v > \phi_u$. Furthermore, since $\overline{T}_u \cap \overline{T}_v \neq \emptyset$ and both u and v are unbounded, it follows that $a_u < a_v$ and $c_v < c_u$. Moreover $N(v) \subseteq N(u)$ by Lemma 4.

Since R is a canonical representation and u is unbounded, it follows that u is an inevitable unbounded vertex of G in R. Then, since $\phi_v > \phi_u$, it follows by the choice of v that the unbounded vertex u has at least one hovering vertex u' in R, which is bounded. That is, $u'u \notin E$ and replacing T_u by $H_{\text{convex}}(\overline{T}_u, a'_u, c'_u)$ in R creates the new edge $u'u$ in G. Therefore $T_{u'} \cap T_u = \emptyset$ and $\overline{T}_{u'} \cap \overline{T}_u \neq \emptyset$. Furthermore, since $u' \notin N(u)$ and $N(v) \subseteq N(u)$, it follows that also $u' \notin N(v)$. Therefore $T_{u'} \cap T_v = \emptyset$.

We will now prove that $\overline{T}_{u'} \cap \overline{T}_v \neq \emptyset$. If $a_u < a_{u'}$, then $T_{u'} \cap T_u \neq \emptyset$ if and only if $\overline{T}_{u'} \cap \overline{T}_u \neq \emptyset$ by Lemma 2. This is a contradiction, since $T_{u'} \cap T_u = \emptyset$ and $\overline{T}_{u'} \cap \overline{T}_u \neq \emptyset$. Therefore $a_{u'} < a_u$. Suppose that $a_{u'} < a_u < d_{u'}$. Then, since $u'u \notin E$, i.e. $T_u \cap T_{u'} = \emptyset$, Lemma 2 implies that $\phi_u > \phi_{u'}(a_u)$. Therefore $c_u < y$ for some point $y \in [c_{u'}, b_{u'}]$ on the line L_1, and thus $c_u < b_{u'}$. That is, $a_{u'} < a_u < a_v$ and $c_v < c_u < b_{u'}$, and thus $\overline{T}_{u'} \cap \overline{T}_v \neq \emptyset$. Suppose now that $d_{u'} < a_u$, i.e. $d_{u'} < a_u < a_v$. Then, since $\overline{T}_{u'} \cap \overline{T}_u \neq \emptyset$, it follows that $c_u < b_{u'}$. That is, $d_{u'} < a_u < a_v$ and $c_v < c_u < b_{u'}$, and thus again $\overline{T}_{u'} \cap \overline{T}_v \neq \emptyset$.

Summarizing, $T_{u'} \cap T_v = \emptyset$ and $\overline{T}_{u'} \cap \overline{T}_v \neq \emptyset$ in every case. Therefore, the replacement of T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in R creates the new edge $u'v$ in G. Thus, u' is a hovering vertex of v. This is a contradiction to the assumption on v, since u' is bounded. Therefore, for every inevitable unbounded vertex $v \in V_U$, there exists a hovering vertex u of v, which is bounded. $\square$

3.1 The construction of a canonical representation

In this section we present an algorithm that constructs a canonical representation of a multitolerance graph G, given a trapezoepiped representation of G.

Definition 8. *Let L be a set of line segments in the plane. The* lower envelope *$Env(L)$ of L is the set of those points $p = (x, y)$ of the line segments of L, such that the point (x, y') does not belong to any line segment of L, for any $y' < y$.*

An example of a set L of non-vertical line segments in the Euclidean plane is illustrated in Figure 3. In this figure, the lower envelope $Env(L)$ of L is drawn gray for better visibility. Furthermore, the lower envelope $Env(L)$ of such a set L consists also of line segments (cf. Figure 3),

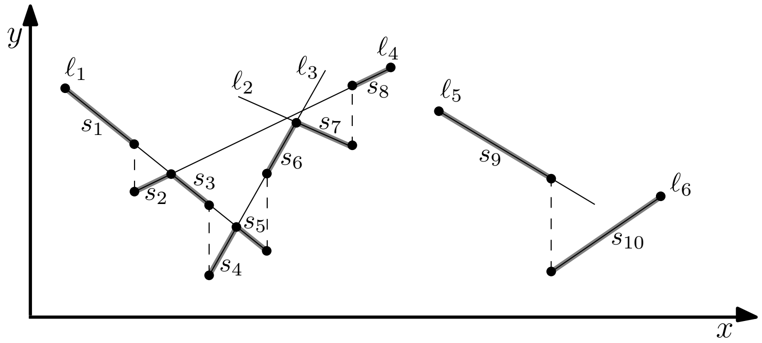


Fig. 3. A set $L = \{\ell_1, \ldots, \ell_6\}$ of line segments in the plane and the lower envelope $H_{\text{lower}}(L)$ of L, which consists of the line segments $\{s_1, \ldots, s_{10}\}$.

and thus $Env(L)$ can be also specified by the endpoints of its segments. Given a set of n line segments in the plane, the lower envelope of these segments can be computed in $O(n \log n)$ time using the algorithm presented in [15]. During the computation of $Env(L)$ by this algorithm, we can in the same time also store for every line segment s of $Env(L)$ the line segment ℓ of L, in which s belongs. We define now two subsets U_1 and U_2 of the set of inevitable unbounded vertices.

Definition 9. *Let $v \in V_U$ be an inevitable unbounded vertex. Then, $v \in U_1$ (resp. $v \in U_2$) if there exists at least one hovering vertex $u \in H(v)$ of v, such that u is unbounded (resp. u is bounded).*

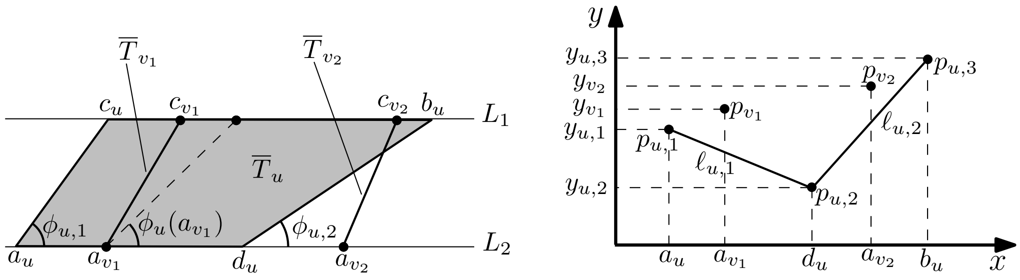
Note that, given a trapezoepiped representation of a multitolerance graph G, the sets U_1 and U_2 are not necessarily disjoint, since an unbounded vertex may have both unbounded and bounded hovering vertices. On the other hand, since every inevitable unbounded vertex has at least one hovering vertex (cf. Definition 6), $U_1 \cup U_2$ coincides with the set of inevitable unbounded vertices.

We associate now with every unbounded vertex $v \in V_U$ the point $p_v = (x_v, y_v)$ in the Euclidean plane, where $x_v = a_v$ and $y_v = \Delta - \cot \phi_v$. Moreover, we associate with every bounded vertex $u \in V_B$ three points $p_{u,1} = (x_{u,1}, y_{u,1}) = (a_u, \Delta - \cot \phi_{u,1})$, $p_{u,2} = (x_{u,2}, y_{u,2}) = (d_u, \Delta - \cot \phi_{u,2})$, and $p_{u,3} = (x_{u,3}, y_{u,3}) = (b_u, \Delta)$ in the plane. Furthermore, we associate with every bounded vertex $u \in V_B$ two line segments $\ell_{u,1} = (p_{u,1}, p_{u,2})$ and $\ell_{u,2} = (p_{u,2}, p_{u,3})$ in the plane, which have the points $p_{u,1}, p_{u,2}$ and $p_{u,2}, p_{u,3}$ as endpoints, respectively. An example of this construction is given in Figure 4, where the points p_{v_1} and p_{v_2} are associated with the unbounded vertices v_1 and v_2, respectively,while the points $p_{u,1}, p_{u,2}, p_{u,3}$ and the line segments $\ell_{u,1}, \ell_{u,2}$ are associated with the bounded vertex u.

In the following, let $L = \{\ell_{u,1}, \ell_{u,2} \mid u \in V_B\}$ be the $2|V_B|$ line segments that are associated with the bounded vertices $u \in V_B$. For an arbitrary point $p = (x, y)$ in the plane, we say that p *lies above* $Env(L)$ (resp. above the line segment $\ell_{u,1}$ or $\ell_{u,2}$ of L) if there exists a point $p' = (x, y')$ of $Env(L)$ (resp. of $\ell_{u,1}$ or $\ell_{u,2}$), such that $y > y'$. The next lemma, which is crucial for the analysis of Algorithm 1, characterizes the vertices of U_2 using the lower envelope $Env(L)$ of L.

Lemma 6. *Let be $v \in V_U$ be an unbounded vertex and $p_v = (x_v, y_v)$ be the associated point in the plane. Then, $v \in U_2$ if and only if p_v lies above $Env(L)$. Furthermore, if p_v lies above the segment $\ell_{u,1}$ or $\ell_{u,2}$ of L, then the bounded vertex u is a hovering vertex of v.*

Proof. Let $u \in V_B$ be a bounded vertex and consider an arbitrary point $x_0 \in [x_{u,1}, x_{u,2}] = [a_u, d_u]$. Then $x_0 = \lambda x_{u,1} + (1 - \lambda) x_{u,2} = \lambda a_u + (1 - \lambda) d_u$ for some value $\lambda \in [0, 1]$. Thus, the



(a) (b)

Fig. 4. Two inevitable unbounded vertices $v_1, v_2 \in U_2$, where the bounded vertex $u \in V_B$ is a hovering vertex of both v_1 and v_2: (a) the trapezoids $\overline{T}_u, \overline{T}_{v_1}, \overline{T}_{v_2}$ and (b) the line segments $\ell_{u,1}, \ell_{u,2}$ of vertex u and the points p_{v_1}, p_{v_2} of vertices v_1, v_2, respectively, where $y_{u,1} = \Delta - \cot(\phi_{u,1})$, $y_{u,2} = \Delta - \cot(\phi_{u,2})$, $y_{u,3} = \Delta$, and $y_{v,j} = \Delta - \cot(\phi_{u,j})$ for every $j = 1, 2$.

point of the line segment $\ell_{u,1}$ with abscissa x_0 is $p_0 = (x_0, \lambda y_{u,1} + (1-\lambda) y_{u,2})$. Recall that $y_{u,1} = \Delta - \cot\phi_{u,1}$ and $y_{u,2} = \Delta - \cot\phi_{u,2}$. Therefore $\lambda y_{u,1} + (1-\lambda) y_{u,2} = \Delta - (\lambda \cot\phi_{u,1} + (1-\lambda)\cot\phi_{u,2})$, and thus $\lambda y_{u,1} + (1-\lambda) y_{u,2} = \Delta - \cot\phi_u(x_0)$ by Observation 1. That is, the point of the line segment $\ell_{u,1}$ with abscissa $x_0 \in [a_u, d_u]$ is $p_0 = (x_0, \Delta - \cot\phi_u(x_0))$.

Consider now a point $x_0 \in [x_{u,2}, x_{u,3}] = [d_u, b_u]$. Then $x_0 = \lambda x_{u,2} + (1-\lambda) x_{u,3} = \lambda d_u + (1-\lambda) b_u$ for some value $\lambda \in [0,1]$. Thus, the point of the line segment $\ell_{u,2}$ with abscissa x_0 is $p_0 = (x_0, \lambda y_{u,2} + (1-\lambda) y_{u,3})$. Recall that $y_{u,3} = \Delta$ and that $y_{u,2} = \Delta - \cot\phi_{u,2}$. Note also that $\cot\phi_{u,2} = b_u - d_u$, and thus $y_{u,2} = \Delta - b_u + d_u$. Therefore $\lambda y_{u,2} + (1-\lambda) y_{u,3} = \Delta - \lambda b_u + \lambda d_u$, and thus $\lambda y_{u,2} + (1-\lambda) y_{u,3} = \Delta - b_u + x_0$, since $x_0 = \lambda d_u + (1-\lambda) b_u$. That is, the point of the line segment $\ell_{u,2}$ with abscissa $x_0 \in [d_u, b_u]$ is $p_0 = (x_0, \Delta - b_u + x_0)$.

Let $v \in V_U$ be an unbounded vertex, such that the point $p_v = (x_v, y_v)$ lies above $Env(L)$. That is, there exists a point $p' = (x_v, y') \in Env(L)$, such that $y_v > y'$. This point p' belongs to the line segment $\ell_{u,1}$ or to the line segment $\ell_{u,2}$, for some bounded vertex $u \in V_B$. In the example of Figure 4(b), the point p_{v_1} (resp. p_{v_2}) that is associated to the unbounded vertex v_1 (resp. v_2) lies above the line segment $\ell_{u,1}$ (resp. $\ell_{u,2}$) that is associated with the bounded vertex u (cf. Figure 4(b)).

Suppose that p' belongs to a line segment $\ell_{u,1}$, where $u \in V_B$. Then $x_v \in [x_{u,1}, x_{u,2}] = [a_u, d_u]$, and thus $p' = (x_v, y') = (x_v, \Delta - \cot\phi_u(x_v))$, as we proved above. Furthermore, since $y_v > y'$, it follows that $y_v = \Delta - \cot\phi_v > \Delta - \cot\phi_u(x_v)$, and thus $\phi_v > \phi_u(x_v)$. That is, there exists a bounded vertex $u \in V_B$, such that $a_u < x_v = a_v < d_u$ and $\phi_v > \phi_u(x_v)$, and thus $T_u \cap T_v = \emptyset$ by Lemma 2. Moreover $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, since $a_u < a_v < d_u$, and thus replacing T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in the trapezoepiped representation creates the new edge uv in G. Therefore v is an inevitable unbounded vertex and the bounded vertex u is a hovering vertex of v by Definition 6. In particular, $v \in U_2$.

Suppose now that p' belongs to a line segment $\ell_{u,2}$, where $u \in V_B$. Then $x_v \in [x_{u,2}, x_{u,3}] = [d_u, b_u]$, and thus $p' = (x_v, y') = (x_v, \Delta - b_u + x_v)$, as we proved above. Recall that $y_v = \Delta - \cot\phi_v$ and that $\cot\phi_v = c_v - a_v$, and thus $y_v = \Delta - c_v + a_v$. Therefore, since $y_v > y'$, it follows that $y_v = \Delta - c_v + a_v > \Delta - b_u + x_v$. Thus, since $x_v = a_v$, it follows that $c_v < b_u$. That is, there exists a bounded vertex $u \in V_B$, such that $d_u < x_v = a_v$ and $c_v < b_u$, and thus $\overline{T}_u \cap \overline{T}_v \neq \emptyset$. Moreover $T_u \cap T_v = \emptyset$ by Lemma 2, since $d_u < a_v$, and thus replacing T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in the trapezoepiped representation creates the new edge uv in G. Therefore v is an inevitable unbounded vertex and the bounded vertex u is a hovering vertex of v by Definition 6. In particular, $v \in U_2$.

Conversely, let $v \in U_2$. Then, replacing T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in the trapezoepiped representation creates a new edge uv in G, where u is a bounded hovering vertex of v. That is,

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

Input: A trapezoepiped representation R of a given multitolerance graph $G = (V, E)$
Output: A canonical representation R' of G and a hovering vertex u for every inevitable unbounded vertex v of G

$L \leftarrow \emptyset$; $R' \leftarrow R$; $R'' \leftarrow R \setminus \{T_v \mid v \in V_B\}$
for every vertex $v \in V$ **do**
  if $v \in V_U$ **then**
    $p_v \leftarrow (a_v, \Delta - \cot\phi_v)$
  else $\{v \in V_B\}$
    $p_{v,1} \leftarrow (a_v, \Delta - \cot\phi_{v,1})$; $p_{v,2} \leftarrow (d_v, \Delta - \cot\phi_{v,2})$; $p_{v,3} \leftarrow (b_v, \Delta)$
    $\ell_{v,1} = (p_{v,1}, p_{v,2})$; $\ell_{v,2} = (p_{v,2}, p_{v,3})$; $L \leftarrow L \cup \{\ell_{v,1}, \ell_{v,2}\}$
Compute the set U_1 of inevitable unbounded vertices in R'' and a hovering vertex $u \in V_U$ of v, for every $v \in U_1$, by the algorithm of [22]
Compute the lower envelope $Env(L)$ of L by the algorithm of [15]
{During the computation of $Env(L)$, store for every line segment s of $Env(L)$, the line segment $\ell_{u,1}$ or $\ell_{u,2}$ of L, in which s belongs}
for every vertex $v \in V_u \setminus U_1$ **do**
  if v lies above a segment s of $Env(L)$ **then** $\{v \in U_2 \setminus U_1\}$
    Let $\ell_{u,1}$ or $\ell_{u,2}$ be the line segment of L, in which s belongs
    $u \in V_B$ is a hovering vertex of v
  else $\{v$ is evitable unbounded$\}$
    Replace T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in R' $\{v$ is made bounded$\}$
return R'

$T_u \cap T_v = \emptyset$ and $\overline{T}_u \cap \overline{T}_v \neq \emptyset$. If $a_v < a_u$, then Lemma 2 implies that $T_u \cap T_v \neq \emptyset$ if and only if $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, which is a contradiction. Therefore $a_u < a_v$. Suppose first that $a_u < a_v < d_u$, or equivalently $x_{u,1} < x_v < x_{u,2}$. Then $\phi_v > \phi_u(a_v)$ by Lemma 2, since $T_u \cap T_v = \emptyset$. Therefore $\Delta - \cot\phi_v > \Delta - \cot\phi_u(a_v)$. Recall now that the point of the line segment $\ell_{u,1}$ with abscissa $x_v = a_v \in [a_u, d_u]$ is $p = (x_v, \Delta - \cot\phi_u(a_v))$. Therefore, since $y_v = \Delta - \cot\phi_v > \Delta - \cot\phi_u(a_v)$, it follows that the point $p_v = (x_v, y_v)$ lies above the line segment $\ell_{u,1}$, and thus p_v lies above $Env(L)$.

Suppose now that $d_u < a_v$. Then, since $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, it follows that $c_v < b_u$. Recall that $a_v < c_v$ (cf. Definition 3 and Figure 1), and thus $d_u < a_v < c_v < b_u$, i.e. $a_v \in [d_u, b_u]$. Note that $y_v = \Delta - \cot\phi_v = \Delta - (c_v - a_v)$. Furthermore, recall that the point on the line segment $\ell_{u,2}$ with abscissa $x_v = a_v \in [d_u, b_u]$ is $p = (a_v, \Delta - b_u + a_v)$. Therefore, since $c_v < b_u$, it follows that $y_v = \Delta - c_v + a_v > \Delta - b_u + a_v$, and thus the point $p_v = (x_v, y_v)$ lies above the line segment $\ell_{u,2}$, i.e. p_v lies above $Env(L)$. This completes the proof of the lemma. □

The next theorem shows that, given a trapezoepiped representation, we can construct by Algorithm 1 a canonical representation in $O(n \log n)$ time. This result is crucial for the time complexity analysis of the algorithms of Section 4.

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

Proof. We describe and analyze Algorithm 1 that generates a canonical representation R' of G. The main idea is to efficiently detect the evitable and the inevitable unbounded vertices in the given trapezoepiped representation R of G. Then, we replace T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ for every evitable unbounded vertex $v \in V_U$, and thus the resulting trapezoepiped representation is canonical.

First, we compute the point p_v in the plane for every unbounded vertex $v \in V_U$ and the three points $p_{v,1}, p_{v,2}, p_{v,3}$ in the plane for every bounded vertex $v \in V_B$. Moreover, we specify for every $v \in V_B$ the two line segments $\ell_{v,1}$ and $\ell_{v,2}$ in the plane with endpoints $p_{v,1}, p_{v,2}$ and $p_{v,2}, p_{v,3}$, respectively, and we compute the set $L = \{\ell_{v,1}, \ell_{v,2} \mid v \in V_B\}$ of $2|V_B|$ line segments.

Then, we consider the part R'' of the initial trapezoepiped representation R that consists only of the trivial trapezoepipeds (lines) T_v, for every unbounded vertex $v \in V_U$. Note that the graph induced by the vertices of V_U is an independent set with cardinality $|V_U|$, since no two unbounded vertices in a multitolerance graph are adjacent. Furthermore, R'' is a special case of a parallelogram representation* (see [22]). We use now Algorithm 1 of [22] that computes in particular all inevitable unbounded vertices v of a given parallelogram representation, as well as a hovering vertex u for each one of them. Observe now that the computed inevitable unbounded vertices v by this algorithm are exactly the vertices of U_1 (cf. Definition 9). Furthermore, for each of these vertices $v \in U_1$, the computed hovering vertex u is unbounded.

In the sequel, we use the algorithm presented in [15] to compute the lower envelope $Env(L)$ of the computed set L of $2|V_B|$ line segments. During the computation of $Env(L)$ by this algorithm, we also store in the same time for every line segment s of $Env(L)$ the line segment ℓ of L, in which s belongs.

After the lower envelope $Env(L)$ of L has been computed, we check for every unbounded vertex $v \in V_U \setminus U_1$ whether the point p_v lies above $Env(L)$. We distinguish the following cases:

Case 1. p_v lies above $Env(L)$. Let s be the line segment of $Env(L)$, above which the point p_v lies, and let $\ell_{u,1}$ or $\ell_{u,2}$ be the line segment of L, in which s belongs, for some $u \in V_B$. Then, Lemma 6 implies that v is an inevitable unbounded vertex ($v \in U_2$) and that u is a bounded hovering vertex of v.

Case 2. p_v does not lie above $Env(L)$. Then $v \notin U_2$ by Lemma 6. Furthermore, since also $v \notin U_1$, it follows that v is not an inevitable unbounded vertex, i.e. v is evitable. Then we replace the trapezoepiped T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in the current trapezoepiped representation and we consider from now on v as a bounded vertex. This replacement does not add any new edge to G, since v is evitable.

Regarding the time complexity, the initialization of the trapezoepiped representations R' and R'' in line 1, as well as the computation of the $O(n)$ points and $O(n)$ line segments in lines 2-7 can be done in linear $O(n)$ time. Furthermore, lines 8 and 9 of Algorithm 1 can be executed in $O(n \log n)$ time [15, 22]. Recall that, during the computation of $Env(L)$ by the algorithm presented in [15], we also store in the same time for every line segment s of $Env(L)$ the line segment ℓ of L, in which s belongs. Furthermore, note that the endpoints of the line segments of $Env(L)$ are returned sorted increasingly according to their x values [15].

After the lower envelope $Env(L)$ of L has been computed, we check for every unbounded vertex $v \in V_U$ in $O(\log n)$ time whether the point $p_v = (x_v, y_v)$ lies above $Env(L)$ (cf. line 11). This can be done as follows. Among all endpoints of the line segments of $Env(L)$, we compute in $O(\log n)$ time the endpoint $p_1 = (x_1, y_1)$ (resp. $p_2 = (x_2, y_2)$), such that x_1 (resp. x_2) is the greatest (resp. the smallest) value with $x_1 < x_v$ (resp. $x_v < x_2$). Then, we test in constant time whether the points p_1 and p_2 define a line segment s of $Env(L)$ and whether the point p_v lies above this segment s. If p_1 and p_2 do not define a segment s of $Env(L)$, or if p_1 and p_2 define such a segment s and p_v does not lie above s, then p_v does not lie above $Env(L)$. Otherwise,

*Note here that, in a parallelepiped representation of a tolerance graph, the height of the line P_v that corresponds to an unbounded vertex v, equals the slope ϕ_v of the line $\overline{P}_v$, which is the projection of P_v to the plane $z = 0$; for details, see [22]. On the other hand, in the trapezoepiped representation R'' (cf. line 1 of Algorithm 1), the height of the line T_v that corresponds to the unbounded vertex $v \in V_U$ equals $\Delta - \cot(\phi_v)$, where ϕ_v is the slope of the line $\overline{T}_v$, which is the projection of T_v to the plane $z = 0$. Note that for every two unbounded vertices $u, v \in V_U$ of the multitolerance graph G, $\Delta - \cot(\phi_v) > \Delta - \cot(\phi_u)$ if and only if $\phi_v > \phi_u$. Therefore, R'' can be considered as a special case of a parallelogram representation by changing the height of every line T_v, where $v \in V_U$, from $\Delta - \cot(\phi_v)$ to ϕ_v, since this change of the heights of the lines does not change the relative position of any two lines in R''.

Algorithm 2 Computation of a minimum coloring of a multitolerance graph G

Input: A trapezoepiped representation R of a given multitolerance graph $G = (V, E)$
Output: A minimum coloring of G

1: Construct a canonical representation R' of G by Algorithm 1, where a hovering vertex u_v is associated with every inevitable unbounded vertex v
2: Let V_B and V_U be the bounded and (inevitable) unbounded vertices of G in R', respectively
3: Color $G[V_B]$ by the algorithm of [6]
4: **for** every vertex $v \in V_U$ **do**
5: Create a pointer from the hovering vertex u_v of v to the vertex v
6: **for** every vertex $u \in V_B$ that has at least one pointer **do**
7: Assign the color of u to every vertex $v \in V_U$ that is reachable from u by a sequence of pointers

if p_v lies above the line segment s of $Env(L)$ defined by the points p_1 and p_2, then p_v lies above $Env(L)$.

Then, the execution of each of the lines 12, 13, and 15 can be simply done in constant time. Therefore, since the lines 11-15 are executed $|V_U \setminus U_1| = O(n)$ times, the complexity of the lines 10-15 is $O(n \log n)$. Summarizing, the total time complexity of Algorithm 1 in $O(n \log n)$. This completes the proof of the theorem. □

4 Coloring and clique Algorithms in $O(n \log n)$ time

In this section we present optimal $O(n \log n)$ algorithms for the minimum coloring and the maximum clique problems on a multitolerance graph G with n vertices, given any trapezoepiped representation of G. To the best of our knowledge, no algorithm for these problems on multitolerance graphs has appeared until now, and thus the previously best known running times for these problems were those of perfect graphs [13].

4.1 Minimum coloring

In the next theorem we present an optimal $O(n \log n)$ algorithm for computing a minimum coloring of a multitolerance graph G with n vertices, given any trapezoepiped representation of G. This algorithm uses Algorithm 1 to compute efficiently a canonical representation of G, as well as the algorithm of [6] that computes a coloring of a given trapezoid graph with n vertices in optimal $O(n \log n)$ time.

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

Proof. We present Algorithm 2 that computes a minimum coloring of G. First, we compute from the initial trapezoepiped representation R of G a canonical representation R' of G by Algorithm 1. Denote by V_B and V_U the sets of bounded and unbounded vertices of G in the canonical representation R', respectively. Then, the induced subgraph $G[V_B]$ of G on the vertices of V_B is a bounded multitolerance (also called bounded bitolerance) graph, i.e. a trapezoid graph [11, 25]. Thus, we compute a minimum coloring of $G[V_B]$ using the algorithm of [6].

Note that every unbounded vertex $v \in V_U$ is inevitable, since R' is canonical. Furthermore, exactly one hovering vertex u_v is assigned to every $v \in V_U$ by Algorithm 1. Now, for every $v \in V_U$, we create a pointer from u_v to v. Since every $v \in V_U$ has exactly one hovering vertex u_v assigned to it, it follows that after the execution of lines 4-5 every $v \in V_U$ has exactly one incoming pointer from some vertex u_v.

Consider now an arbitrary inevitable unbounded vertex $v \in V_U$ and its hovering vertex u_v. If $u_v \in V_B$, then a color has been assigned to u_v by the coloring of the graph $G[V_B]$. Suppose that $u_v \in V_U$. Then, u_v is unbounded in both the canonical representation R' and in the initial representation R of G. Furthermore, if we replace T_v by $H_{\text{convex}}(\overline{T}_v, a'_v, c'_v)$ in R, we create the new edge $u_v v$ in G. Thus, since both u_v and v are unbounded in R, it follows in particular that $\Delta - \cot(\phi_v) > \Delta - \cot(\phi_{u_v})$, i.e. $\phi_v > \phi_{u_v}$. That is, if a vertex $u_v \in V_U$ has a pointer to a vertex $v \in V_U$, then $\phi_v > \phi_{u_v}$. Therefore, for every vertex $v = v_0 \in V_U$, there exists a maximal chain $(v_0, v_1, \ldots, v_k)$ of vertices in V_U, such that v_i has a pointer to v_{i-1} and $\phi_{v_{i-1}} > \phi_{v_i}$, for every $i = 1, 2, \ldots, k$. Then, since v_i is a hovering vertex of v_{i-1} for every $i = 1, 2, \ldots, k$, it follows by Lemma 4 that $N(v_0) \subseteq N(v_1) \subseteq \ldots \subseteq N(v_k)$. Furthermore, since R' is canonical, the unbounded vertex v_k is inevitable unbounded, and thus v_k has an incoming pointer from a hovering vertex u_{v_k} of v_k. In particular, u_{v_k} is bounded in R', i.e. $u_{v_k} \in V_B$, due to the maximality of the chain $(v_0, v_1, \ldots, v_k)$ in V_U. Moreover $N(v_k) \subseteq N(u_{v_k})$ by Lemma 4, since u_{v_k} is a hovering vertex of v_k, and thus also $N(v_0) \subseteq N(v_k)$. That is, for every vertex $v \in V_U$, there exists exactly one vertex $u \in V_B$, such that v can be reached by a sequence of pointers from u and $N(v) \subseteq N(u)$.

Now, starting from every bounded vertex $u \in V_B$, we assign the color of u (in the coloring of $G[V_B]$) to every unbounded vertex v that is reachable from u by a sequence of pointers. It remains to prove that this coloring is a proper coloring of G. Suppose otherwise that there exists a vertex $v \in V_U$ and a vertex $w \in N(v)$, such that u and w have the same color. Then w is bounded in both representations R and R', since two unbounded vertices are never adjacent and $w \in N(v)$. Let $u \in V_B$ be the unique vertex in G, such that v can be reached from u by a sequence of pointers. Then $N(v) \subseteq N(u)$ by the previous paragraph, and thus also $w \in N(u)$. Furthermore, all u, v, and w have the same color. This is a contradiction, since $u, w \in V_B$ and the coloring of $G[V_B]$ is proper. Therefore, for every vertex $v \in V_U$, all vertices $w \in N(v)$ have different color than v, and thus the constructed coloring of G is proper. Furthermore, this coloring is minimum, since the constructed coloring of $G[V_B]$ in line 3 is also minimum [6].

Regarding the time complexity, the computation of the canonical representation R' of G can be done in $O(n \log n)$ time by Theorem 2. Furthermore, a minimum coloring of $G[V_B]$ can be computed by the algorithm of [6] in $O(n \log n)$ time. The execution of the lines 4-5 can be done in linear $O(n)$ time, since we visit every vertex $v \in V_U$ once. Finally, the execution of the lines 6-7 can be also done in linear $O(n)$ time, by traversing the graph G following the pointers that we created in lines 4-5. Summarizing, Algorithm 2 computes a minimum coloring of G in $O(n \log n)$ time. Moreover, since $\Omega(n \log n)$ is a lower bound for the time complexity of the minimum coloring problem on tolerance graphs [22] and on trapezoid graphs [6], it follows that Algorithm 2 has also optimal running time. □

4.2 Maximum clique

In the next theorem we present an optimal $O(n \log n)$ algorithm for computing a maximum clique of a multitolerance graph G with n vertices, given any trapezoepiped representation of G. This algorithm uses the Algorithm 1 for the efficient construction of a canonical representation of G, as well as the algorithm of [6] that computes a maximum clique of a given trapezoid graph with n vertices in optimal $O(n \log n)$ time.

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

Proof. First we compute a canonical representation of G in $O(n \log n)$ time by Algorithm 1. By the correctness of Algorithm 2, cf. the proof of Theorem 3, it follows that $\chi(G) = \chi(G[V_B])$, where $\chi(H)$ denotes the chromatic number of a given graph H. Since multitolerance graphs are perfect graphs [25], $\omega(G) = \chi(G)$ and $\omega(G[V_B]) = \chi(G[V_B])$, where $\omega(H)$ denotes the clique number of a given graph H. Therefore $\omega(G) = \omega(G[V_B])$. We compute now a maximum clique Q of the bounded multitolerance (i.e. trapezoid) graph $G[V_B]$ in $O(n \log n)$ time by the algorithm presented in [6] for trapezoid graphs. Then, since $\omega(G) = \omega(G[V_B])$, Q is a maximum clique of G as well. Finally, since $\Omega(n \log n)$ is a lower bound for the time complexity of the maximum clique problem on tolerance graphs [22] and on trapezoid graphs [6], it follows that the clique algorithm for multitolerance graphs has also optimal running time. $\square$

5 Weighted independent set algorithm in $O(m + n \log n)$ time

In this section we present an algorithm that computes a maximum weight independent set in a multitolerance graph $G = (V, E)$ with n vertices and m edges in $O(m + n \log n)$ time, given a trapezoepiped representation of G and a weight $w(v) > 0$ for every vertex $v \in V$. Although the algorithm presented in [22] for the maximum weight independent set on tolerance graphs with complexity $O(n^2)$ can be extended with the same time complexity to the case of multitolerance graphs with a given trapezoepiped representation, we present here a new algorithm for multitolerance graphs that achieves a better running time $O(m + n \log n)$. Thus this algorithm improves also the best known running time of $O(n^2)$ for the maximum weight independent set on tolerance graphs [22]. Note here that $\Omega(n \log n)$ is a lower bound for the time complexity of this problem on trapezoid graphs [6], and thus also on multitolerance graphs.

First, given a trapezoepiped representation of a multitolerance graph $G = (V, E)$, we sort on the line L_2 the points $\{a_v, d_v \mid v \in V\}$ of the trapezoids $\overline{T}_v$, $v \in V$, and we visit these points sequentially from right to left. Note that $a_v = d_v$ for every unbounded vertex $v \in V_U$. A vertex v is called *processed* only after we visit the endpoint a_v. During the execution of the algorithm we maintain two finite sets M and H of $O(n)$ weighted markers each on the line L_1, which are placed on some points c_v, where $v \in V$. We maintain the sets M and H in such a way that values can be inserted to and deleted from these sets, as well as the predecessor or successor of a given query value can be found. Using binary search trees, for instance AVL-trees, all these operations can be executed in $O(\log n)$ time [12].

The markers of the set M are placed at points c_v on the line L_1, for some bounded vertices $v \in V_B$. After an iteration of the algorithm, where the vertices of the set $U \subseteq V$ have been processed, the weight $W(m)$ of a marker m placed at the point c_v on the line L_1 equals the maximum weight of an independent set, which includes only vertices $u \in U$ such that $c_v \leq c_u$. Moreover, a marker m is placed at c_v only if such a maximum weight independent set includes the (bounded) vertex v.

The markers of the set H are placed at points c_v on the line L_1, where $v \in V_U$. After an iteration of the algorithm, where the vertices of the set $U \subseteq V$ have been processed, there is a weighted marker $h \in H$ placed at the point c_v on L_1, for every unbounded vertex $v \in V_u \cap U$, while the weight $w(h)$ of h equals the weight $w(v)$ of vertex v. Furthermore, in the AVL-tree of the set H, we store at every internal vertex x also a label with the total weight of the tree that consists of x and its right subtree. Note that after an insertion of a new marker h to the AVL-tree that stores H, we can update in $O(\log n)$ time these labels of the internal vertices, as follows. First, we need to update a constant number of labels during the "trinode restructure" operation (for more details, see [12]). Then, following the path from the interval vertex that stores the new marker h to the root, we add the weight $w(h)$ of h to the label of every internal vertex that has h in its right subtree. For every marker $m \in M$ (resp. $h \in H$), we denote by p_m (resp. p_h) the point of L_1, at which the marker m (resp. h) is placed.

Algorithm 3 Maximum weight independent set of a multitolerance graph G

Input: A trapezoepiped representation of a given multitolerance graph $G = (V, E)$
Output: The value of a maximum weight independent set of G

Place a marker m_0 at the point $p_{m_0} = \max\{b_v \mid v \in V\} + 1$ of the line L_1
$W(m_0) \leftarrow 0$; $M \leftarrow \{m_0\}$

for every $v \in V_B$ **do** {initialization}
    $W(v) \leftarrow 0$
    Compute the value $\widetilde{w}_v = \sum\{w(u) \mid u \in V_U, c_u \in (c_v, b_v)\}$
    for every $u \in N(v)$ **do**
        if $u \in V_U$ and $c_u \in (c_v, b_v)$ **then** {v is not a hovering vertex of u}
            $\widetilde{w}_v \leftarrow \widetilde{w}_v - w(u)$

for each point $p \in \{a_v, d_v \mid v \in V\}$ from right to left **do** {p lies on the line L_2}
    if $p = a_v$ for some $v \in V_U$ **then** {the unbounded vertex v is being processed}
        Insert a new marker $h \in H$ at the point $p_h = c_v$
        $w(h) \leftarrow w(v)$
        $m \leftarrow$ the leftmost marker of M to the right of c_v on L_1
        Remove all markers $m' \in M$ to the left of m, for which $W(m') \leq W(m) + w(H[p_{m'}, p_m))$
    if $p = d_v$ for some $v \in V_B$ **then**
        $m \leftarrow$ the leftmost marker of M to the right of b_v on L_1
        $W(v) \leftarrow (w(v) + \widetilde{w}_v) + W(m) + w(H[b_v, p_m))$ {do not modify the markers of M}
    if $p = a_v$ for some $v \in V_B$ **then** {the bounded vertex v is being processed}
        $m \leftarrow$ the leftmost marker of M to the right of c_v on L_1
        if $W(v) > W(m) + w(H[c_v, p_m))$ **then**
            Insert a new marker $m' \in M$ at the point $p_{m'} = c_v$
            $W(m') \leftarrow W(v)$
            Remove all markers $m'' \in M$ to the left m', for which $W(m'') \leq W(m') + w(H[p_{m''}, p_{m'}))$
return $W(m) + w(H[c, p_m))$, where $c = \min\{c_v \mid v \in V\} - 1$ and m is the leftmost marker of M

For every two points q and q' on L_1, where $q < q'$, denote for simplicity by $H[q, q')$ (resp. $H[q, +\infty)$) the set of the markers h in the current set H that have been placed in the semi-closed interval $[q, q')$ (resp. in the subline $[q, +\infty)$) of L_1. Denote also by $w(H[q, q'))$ (resp. $w(H[q, +\infty))$) the sum of the weights of the markers $h \in H[q, q')$ (resp. $h \in H[q, +\infty)$). For simplicity, in the case where $q' = q$, we set $w(H[q, q)) = 0$. Furthermore, note that if $q \leq q' \leq q''$, then $w(H[q, q'')) = w(H[q, q')) + w(H[q', q''))$. For every point q on L_1, we can compute in $O(\log n)$ time the value $w(H[q, +\infty))$, as follows. First, we locate in $O(\log n)$ time the leftmost marker $h \in H$ that has been placed at a point q', such that $q \leq q'$. Then, we follow in the AVL-tree of H the path from the root to the internal vertex x that stores h and sum up the label stored at x and the labels stored at the internal vertices of this path, at which we follow the left child. Furthermore, since $w(H[q, q')) = w(H[q, +\infty)) - w(H[q', +\infty))$ for every two points q, q' on the line L_1 such that $q \leq q'$, we can compute the value $w(H[q, q'))$ in $O(\log n)$ time as well.

In the following we present our Algorithm 3 that computes the value of a maximum weight independent set of a multitolerance graph G, given a trapezoepiped representation of G. A slight modification of this algorithm computes in the same time also a maximum weight independent set of G, instead of its value. For every marker $m \in M$ (resp. $h \in H$), we denote by p_m (resp. p_h) the point of L_1, at which the marker m (resp. h) is placed.

We now prove in the next lemma that Algorithm 3 maintains a monotonicity property of the weights of the markers in the set M, which is crucial for the proof of correctness of the algorithm.

Lemma 7. *Let m_1 and m_2 be two markers of M after an iteration of Algorithm 3, such that $p_{m_2} < p_{m_1}$. Then $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$.*

Proof. The proof is done by induction on the iterations of the algorithm. The condition of the lemma clearly holds before the first iteration, since initially the set M has only one marker m_0. This proves the induction basis. Suppose that the condition of the lemma holds after an iteration of the algorithm. For the induction step, consider the next iteration, during which the algorithm visits a point $p \in \{a_v, d_v \mid v \in V\}$ on the line L_2. We distinguish in the following the cases regarding the point p.

Case 1. $p = a_v$ for some unbounded vertex $v \in V_U$, i.e. v is being processed at this iteration. Let m be the leftmost marker of M to the right of c_v on L_1 (cf. line 13 of Algorithm 3). The algorithm adds a marker h to the set H at the point c_v of the line L_1 with weight $w(h) = w(v)$. Furthermore, the algorithm removes all markers $m' \in M$, such that $p_{m'} < p_m$ and $W(m') \leq W(m)+w(H[p_{m'}, p_m))$ (cf. line 14 of Algorithm 3). Consider two markers $m_1, m_2 \in M$, such that $p_{m_2} < p_{m_1}$, i.e. m_2 is lies to the left of m_1 on the line L_1 after the iteration in which v is processed. Note that $c_v \neq p_{m_1}$ and $c_v \neq p_{m_2}$, since v is an unbounded vertex and the points p_{m_1} and p_{m_2} correspond to bounded vertices. Suppose first that $c_v < p_{m_2} < p_{m_1}$ or $p_{m_2} < p_{m_1} < c_v$. Then the values $W(m_1)$, $W(m_2)$, and $w(H[p_{m_2}, p_{m_1}))$ are the same before and after v is processed, and thus $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$ by the induction hypothesis. Suppose now that $p_{m_2} < c_v < p_{m_1}$. If $m_1 = m$, then $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$, since otherwise the marker m_2 would be removed from M after the process of v, which is a contradiction. If $m_1 \neq m$, then $p_{m_2} < c_v < p_m < p_{m_1}$ by definition of m. Note that $W(m_2) > W(m) + w(H[p_{m_2}, p_m))$, since otherwise m_2 would be removed from M, which is again a contradiction. Furthermore $W(m) > W(m_1) + w(H[p_m, p_{m_1}))$, as we proved above, since $c_v < p_m < p_{m_1}$. Summing up the last two inequalities, it follows that $W(m_2) > W(m_1) + w(H[p_{m_2}, p_m)) + w(H[p_m, p_{m_1}))$, i.e. $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$.

Case 2. $p = d_v$ for some bounded vertex $v \in V_B$. Then, v is not being processed at this iteration and the algorithm does not modify the sets M and H. Thus the condition of the lemma holds by the induction hypothesis.

Case 3. $p = a_v$ for some bounded vertex $v \in V_B$, i.e. v is being processed at this iteration. Note that the value $W(v)$ has been computed previously (at a previous iteration, if $a_v < d_v$, or at the current iteration, if $a_v = d_v$). Let m be the leftmost marker of M to the right of c_v on L_1 (cf. line 19 of Algorithm 3). If $W(v) \leq W(m) + w(H[c_v, p_m))$, then the sets M and H are not modified (cf. line 20 of Algorithm 3), and thus the condition of the lemma holds by the induction hypothesis. Suppose that $W(v) > W(m) + w(H[c_v, p_m))$. Then, the algorithm adds a new marker m' to the set M at the point c_v with weight $W(m') = W(v)$. Furthermore, it removes from M all markers m'', such that $p_{m''} < p_{m'}$ and $W(m'') \leq W(m') + w(H[p_{m''}, p_{m'}))$ (cf. line 23 of Algorithm 3).

Suppose first that $c_v < p_{m_2} < p_{m_1}$ or $p_{m_2} < p_{m_1} < c_v$. Then the values $W(m_1)$, $W(m_2)$, and $w(H[p_{m_2}, p_{m_1}))$ are the same before and after v is processed, and thus $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$ by the induction hypothesis. Suppose that $c_v = p_{m_1}$, i.e. $m' = m_1$ and $p_{m_2} < c_v = p_{m_1} = p_{m'}$. Then $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$, since otherwise the marker m_2 would be removed from M after the process of v, which is a contradiction. Suppose that $c_v = p_{m_2}$, i.e. $m' = m_2$ and $c_v = p_{m_2} < p_m \leq p_{m_1}$ by definition of m. Then $W(m') = W(v) > W(m) + w(H[c_v, m))$ by definition of $W(m')$, i.e. $W(m_2) > W(m) + w(H[p_{m_2}, p_m))$. If $m_1 \neq m$, then $W(m) > W(m_1) + w(H[p_m, p_{m_1}))$, as we proved above, since in this case $c_v < p_m < p_{m_1}$. Therefore, summing up the last two inequalities, it follows that $W(m_2) > W(m_1) + w(H[p_{m_2}, p_m)) + w(H[p_m, p_{m_1}))$, i.e. $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$. If $m_1 = m$, then it follows by substitution that $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$, since $W(m_2) > W(m) + w(H[p_{m_2}, p_m))$.

Suppose now that $p_{m_2} < c_v < p_{m_1}$, i.e. $p_{m_2} < c_v = p_{m'} < p_m \leq p_{m_1}$ by definition of m and m'. Recall that $W(v) > W(m) + w(H[c_v, p_m))$. Thus, since $W(m') = W(v)$ and $c_v = p_{m'}$, it follows that $W(m') > W(m) + w(H[p_{m'}, p_m))$. Furthermore, note that $W(m_2) > W(m') + w(H[p_{m_2}, p_{m'}))$, since otherwise the marker m_2 would be removed from M after the process of v, which is a contradiction. Therefore, summing up the last two inequalities, it follows that $W(m_2) > W(m) + w(H[p_{m_2}, p_{m'})) + w(H[p_{m'}, p_m))$, i.e. $W(m_2) > W(m) + w(H[p_{m_2}, p_m))$. If $m_1 = m$, then it follows by substitution that $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$. If $m_1 \neq m$, then $W(m) > W(m_1) + w(H[p_m, p_{m_1}))$, as we proved above, since in this case $c_v < p_m < p_{m_1}$. Recall now that $W(m_2) > W(m) + w(H[p_{m_2}, p_m))$. Therefore, summing up the last two inequalities, it follows that $W(m_2) > W(m_1) + w(H[p_{m_2}, p_m)) + w(H[p_m, p_{m_1}))$, i.e. $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$.

Summarizing Cases 1, 2, and 3, $W(m_2) > W(m_1) + w(H[p_{m_2}, p_{m_1}))$ for every two markers $m_1, m_2 \in M$ such that $p_{m_2} < p_{m_1}$ after the iteration, during which the algorithm visits the point $p \in \{a_v, d_v \mid v \in V\}$. This completes the induction step, and thus the lemma follows. □

Recall that $w(H[q, q')) = w(H[q, +\infty)) - w(H[q', +\infty))$ for every two points q, q' on the line L_1 such that $q \leq q'$. Therefore, the next corollary follows directly by Lemma 7.

Corollary 1. *Let m_1 and m_2 be two markers of M after an iteration of Algorithm 3. Then $p_{m_2} < p_{m_1}$ if and only if $W(m_2) - w(H[p_{m_2}, +\infty)) > W(m_1) - w(H[p_{m_1}, +\infty))$.*

Definition 10. *Let $U \subseteq V$ be a set of vertices and y be a point of the line L_1. Then we define $Opt(U, y)$ to be a maximum weight independent set, which includes only vertices $u \in U$ such that $y \leq c_u$. The* weight *$w(Opt(U, y))$ is the total weight of the vertices of $Opt(U, y)$.*

Suppose now that Algorithm 3 visits the point $p \in \{a_v, d_v \mid v \in V\}$ of the line L_2 at some iteration. Then, $U = \{u \in V \mid p \leq a_v\}$ is the set of vertices that have been *processed* after this iteration by the algorithm. For every point $y \leq p_{m_0}$ of the line L_1, the next lemma determines the value $w(Opt(U, y))$ using only the markers of the sets M and H after this iteration. For the sake of presentation, we may not distinguish in the following between the set $H[q, q')$ of markers (for some points q, q' on the line L_1) and the set of the corresponding unbounded vertices, whenever this slight abuse of notation does not cause any confusion.

Lemma 8. *Let $U \subseteq V$ be the set of vertices that have been processed after an iteration of Algorithm 3. Let $y \leq p_{m_0}$ be a point on the line L_1 and m_y be the leftmost marker of M after this iteration, for which $y \leq p_{m_y}$. Then $w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$.*

Proof. The proof is done by induction on the iterations of the algorithm. Before the first iteration of the algorithm, $U = \emptyset$. Furthermore, in this case $m_y = m_0$, while $W(m_0) = 0$ and $w(H[y, p_{m_y})) = 0$. Thus, before the first iteration, $W(m_y) + w(H[y, p_{m_y})) = 0$ equals the weight $w(Opt(U, y)) = 0$ of a maximum weight independent set $Opt(U, y)$, which includes only vertices $u \in U = \emptyset$ such that $y \leq c_u$. This proves the induction basis.

Suppose that the condition of the lemma holds after an iteration of the algorithm, where the vertices of the set $U \subseteq V$ have been processed. For the induction step, consider the next iteration, during which the algorithm visits a point $p \in \{a_v, d_v \mid v \in V\}$ on the line L_2. Let m_y (resp. m'_y) be the leftmost marker of M, for which $y \leq p_{m_y}$ (resp. $y \leq p_{m'_y}$) after (resp. before) the iteration of the algorithm that the point p is visited. We distinguish in the following the cases regarding the currently visited point p of L_2.

Case 1. $p = a_v$ for some unbounded vertex $v \in V_U$. Then, v is being processed at this iteration, i.e. after this iteration the vertices of the set $U \cup \{v\}$ have been processed. Let m^* be the leftmost marker of M to the right of c_v on L_1 (cf. line 13 of Algorithm 3). Note that the marker m^* belongs to M before, as well as after the process of v. The algorithm adds a marker h to the set H at the point $p_h = c_v$ of the line L_1, while the weight of h is $w(h) = w(v)$. Moreover, the algorithm removes all markers $m' \in M$, such that $p_{m'} < p_{m^*}$ and $W(m') \leq W(m^*) + w(H[p_{m'}, p_{m^*}))$. Note that, during the process of the unbounded vertex v, no new marker is inserted to M, while some markers of M may be removed. Therefore, in particular marker m_y exists in the set M also before the process of v, i.e. $p_{m'_y} \leq p_{m_y}$ by definition of m'_y. We distinguish in the following the cases regarding the position of the point $p_h = c_v$ on the line L_1. Note that $p_h \neq p_{m_y}$ and $p_h \neq p_{m'_y}$, since the point p_h corresponds to an unbounded vertex and the points p_{m_y} and $p_{m'_y}$ correspond to bounded vertices.

Suppose first that $p_h < y$. Then for every vertex $u \in U \cup \{v\}$ such that $y \leq c_u$, it follows that $u \neq v$, i.e. $u \in U$. Therefore $Opt(U \cup \{v\}, y) = Opt(U, y)$. Furthermore $m'_y = m_y$, since during the process of v only markers to the left of $p_h = c_v$ on L_1 may be removed from M. Thus, since $w(Opt(U, y)) = W(m'_y) + w(H[y, p_{m'_y}))$ by the induction hypothesis, it follows that also $w(Opt(U \cup \{v\}, y)) = Opt(U, y) = W(m_y) + w(H[y, p_{m_y}))$. Since $p_h < y$, note here that the values $w(H[y, p_{m'_y}))$ and $w(H[y, p_{m_y}))$ remain the same before and after the addition of the marker h to H.

Suppose now that $y \leq p_h$. Recall by the induction hypothesis that $w(Opt(U, y)) = W(m'_y) + w(H[y, p_{m'_y}))$, where the value $w(H[y, p_{m'_y}))$ is computed before the process of v, i.e. before the addition of the marker h to H.

Let S_v be a maximum weight independent set, which includes the unbounded vertex v and vertices $u \in U$ such that $y \leq c_u$. We will prove that the total weight of the vertices of S_v is $w(S_v) = W(m^*) + w(H[y, p_{m^*}))$, where the value $w(H[y, p_{m^*}))$ is computed after the insertion of h to H. To this end, let $u \in U$ be a bounded vertex of S_v. Then $a_v < a_u$, since v is processed by the algorithm after $u \in U$. If $c_u < c_v = p_h$, then $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, and thus also $T_u \cap T_v \neq \emptyset$ by Lemma 2. That is, $uv \in E$, which is a contradiction, since S_v is an independent set. Therefore $p_h = c_v < c_u$ for every bounded vertex of S_v. Consider now the point $p_h = c_v$ of L_1 before the process of vertex v. The induction hypothesis implies that $w(Opt(U, p_h)) = W(m^*) + w(H[p_h, p_{m^*}))$, where here the value $w(H[p_h, p_{m^*}))$ is computed before the process of v, i.e. without the weight $w(h) = w(v)$. Note that for every vertex u of the independent set $Opt(U, p_h)$ we have $p_h < c_u$. Therefore, for every vertex $u \in Opt(U, p_h)$ and every unbounded vertex $u' \in \{v\} \cup H[y, p_h)$, Lemma 1 implies that $T_u \cap T_{u'} = \emptyset$, i.e. $uu' \notin E$, since $c_{u'} \leq p_h < c_u$ for all such vertices u, u'. Thus, since $Opt(U, p_h)$ is optimal (before the process of v), it follows that the set $Opt(U, p_h) \cup \{v\} \cup H[y, p_h)$ is a maximum weight independent set that includes vertex v, as well as unbounded vertices $u \in U$ with $y \leq c_u$ and bounded vertices $u' \in U$ with $p_h < c_{u'}$. Therefore, since $p_h = c_v < c_u$ for every bounded vertex u of S_v, it follows that the independent sets S_v and $Opt(U, p_h) \cup \{v\} \cup H[y, p_h)$ have the same total weight. Moreover, note that the weight of $Opt(U, p_h) \cup \{v\} \cup H[y, p_h)$ is equal to $W(m^*) + w(H[y, p_{m^*}))$ (after the insertion of h to H). Therefore $w(S_v) = W(m^*) + w(H[y, p_{m^*}))$, where the value $w(H[y, p_{m^*}))$ is computed after the insertion of h to H.

For the sequel of the analysis for Case 1, we will compare the total weight $w(S_v)$ of S_v with the weight $w(Opt(U, y))$ of the independent set $Opt(U, y)$ that does not include vertex v. It is easy to see that if $w(S_v) \geq w(Opt(U, y))$, then $w(Opt(U \cup \{v\}, y)) = w(S_v)$. Furthermore, if $w(S_v) \leq w(Opt(U, y))$, then $w(Opt(U \cup \{v\}, y)) = w(Opt(U, y))$. Note that the induction hypothesis implies that $Opt(U, y) = W(m'_y) + w(H[y, p_{m'_y}))$, where here the value $w(H[y, p_{m'_y}))$ is computed before the insertion of h to H. Recall that $y \leq p_h$. Therefore, either $y \leq p_h < p_{m'_y}$ or $y \leq p_{m'_y} < p_h$, as we distinguish in the following cases.

Case 1a. $y \leq p_h < p_{m'_y}$. Then m'_y belongs to M also after the process of v, since during the process of v only markers to the left of $p_h = c_v$ on L_1 may be removed from M. Therefore $m'_y = m_y$. Furthermore $m'_y = m^*$ by the definition of m^*. Thus, it follows by the induction hypothesis that $w(Opt(U, y)) = W(m^*) + w(H[y, p_{m^*}))$, where the value $w(H[y, p_{m^*}))$ is computed before the insertion of h to H. Recall that $w(S_v) = W(m^*) + w(H[y, p_{m^*}))$, where here the value $w(H[y, p_{m^*}))$ is computed after the insertion of h to H. Therefore $w(S_v) > w(Opt(U, y))$, since $w(h) = w(v) > 0$, and thus $w(Opt(U \cup \{v\}, y)) = w(S_v) = W(m^*) + w(H[y, p_{m^*}))$ after the insertion of h to H. Therefore, since $m^* = m'_y = m_y$, it follows that $w(Opt(U \cup \{v\}, y)) = W(m_y) + w(H[y, p_{m_y}))$ after the insertion of h to H.

Case 1b. $y \leq p_{m'_y} < p_h$. Recall that m^* is the leftmost marker of M to the right of $p_h = c_v$ on L_1 (cf. line 13 of Algorithm 3). Then $y \leq p_{m'_y} < p_h < p_{m^*}$. Recall also by the induction hypothesis that $w(Opt(U, y)) = W(m'_y) + w(H[y, p_{m'_y}))$. Note here that the value $w(H[y, p_{m'_y}))$ remains the same before and after the insertion of h to H, since $p_{m'_y} < p_h$.

Suppose first that m'_y is removed from M during the process of v, i.e. $W(m'_y) \leq W(m^*) + w(H[p_{m'_y}, p_{m^*}))$ after the insertion of h to H (cf. line 14 of Algorithm 3). Therefore, after the insertion of h to H, $W(m^*) + w(H[y, p_{m^*})) = W(m^*) + w(H[y, p_{m'_y})) + w(H[p_{m'_y}, p_{m^*})) \geq W(m'_y) + w(H[y, p_{m'_y}))$, i.e. $W(m^*) + w(H[y, p_{m^*})) \geq w(Opt(U, y))$, where the value $w(H[y, p_{m^*}))$ is computed after insertion of h to H. We will now prove that $m^* = m_y$. Consider any marker m of M with $p_{m'_y} < p_m < p_{m^*}$ before the process of v, i.e. $p_{m'_y} < p_m < p_h < p_{m^*}$. Then Lemma 7 implies that $W(m) + w(H[p_{m'_y}, p_m)) < W(m'_y)$ before the process of v. Therefore, since $W(m'_y) \leq W(m^*) + w(H[p_{m'_y}, p_{m^*}))$ after the insertion of h to H, it follows that $W(m) + w(H[p_{m'_y}, p_m)) < W(m^*) + w(H[p_{m'_y}, p_{m^*}))$ after the insertion of h to H, and thus $W(m) \leq W(m^*) + w(H[p_m, p_{m^*}))$ after the insertion of h to H, since $w(H[p_{m'_y}, p_{m^*})) = w(H[p_{m'_y}, p_m)) + w(H[p_m, p_{m^*}))$. That is, every marker m, for which $p_{m'_y} \leq p_m < p_{m^*}$, is removed from M during the process of v in line 14 of Algorithm 3. Therefore m^* is the leftmost marker of M to the right of y on the line L_1 after the process of v, i.e. $m^* = m_y$. Recall that $w(S_v) = W(m^*) + w(H[y, p_{m^*})) \geq w(Opt(U, y))$ after the insertion of h to H. Therefore, since $m^* = m_y$, it follows that $w(Opt(U \cup \{v\}, y)) = w(S_v) = W(m_y) + w(H[y, p_{m_y}))$ after the insertion of h to H.

Suppose now that m'_y is not removed from M during the process of v, i.e. $m'_y = m_y$. Then the induction hypothesis implies that $w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$, since $m'_y = m_y$. Note here that the value $w(H[y, p_{m_y}))$ remains the same before and after the insertion of h to H, since $p_{m_y} = p_{m'_y} < p_h$. Recall now that $y \leq p_{m_y} = p_{m'_y} < p_h < p_{m^*}$. Therefore Lemma 7 implies that $W(m_y) > W(m^*) + w(H[p_{m_y}, p_{m^*}))$, i.e. $W(m_y) + w(H[y, p_{m_y})) > W(m^*) + w(H[y, p_{m^*}))$ after the addition of h to H. This is equivalent to $w(Opt(U, y)) > w(S_v)$, and thus $w(Opt(U \cup \{v\}, y)) = w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$ after the insertion of h to H.

Case 2. $p = d_v$ for some bounded vertex $v \in V_B$. Then, no new vertex is being processed at this iteration, i.e. the set U is not modified. Furthermore, the algorithm does not modify the sets M and H, and thus the induction step follows in this case by the induction hypothesis. At this iteration, the algorithm computes and stores a value $W(v)$. This value will be used at the iteration where the algorithm visits the point a_v of the bounded vertex v, i.e. when vertex v is processed. If a new marker m' is inserted to M during that iteration (at the point c_v of the line L_1, cf. line 21 of Algorithm 3), then the value $W(v)$ is assigned as the weight $W(m')$ of m' (cf. line 22).

We will now prove that the computed value $W(v)$ equals the maximum weight of an independent set, which includes the bounded vertex v and vertices of the set $U' = \{u \in V \mid a_v < a_u, c_v < c_u\}$. This fact will be used for the proof of the induction step in Case 3. First observe that the value $\widetilde{w}_v$ that is computed in lines 3-8 of Algorithm 3 equals the total weight of the unbounded vertices u, such that v is a hovering vertex of u and $c_v < c_u < b_v$. Note that $a_v < a_u$ for every such unbounded vertex u. Indeed, otherwise $a_u < a_v$ and $c_v < c_u$, i.e. $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, and thus also $T_u \cap T_v \neq \emptyset$ by Lemma 2, which is a contradiction, since $uv \notin E$.

Consider a maximum weight independent set S that includes the bounded vertex v and vertices of U'. Observe that for every vertex $u \in S \setminus \{v\}$, either v is a hovering vertex of the unbounded vertex u, or $b_v < c_u$ and $d_v < a_u$. Let v be a hovering vertex of the unbounded vertex $u \in S \setminus \{v\}$, and suppose that $b_v < c_u$. Then, since v is a hovering vertex of u, it follows that $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, and thus $a_u < d_v$. That is, $a_u < d_v$ and $b_v < c_u$, and thus $\phi_u < \phi_{v,2}$. Therefore, the line T_u intersects the trapezoepiped T_v in the trapezoepiped representation of G, and thus $uv \in E$, which is a contradiction. Thus $c_u < b_v$, if v is a hovering vertex of the unbounded vertex $u \in S \setminus \{v\}$, and thus $c_v < c_u < b_v$, since $u \in U'$ by assumption. Let now m be the leftmost marker of M on the line L_1, such that $b_v < p_m$ (cf. line 16 of Algorithm 3). The induction hypothesis implies that $w(Opt(U, b_v)) = W(m) + w(H[b_v, p_m))$. Note that $Opt(U, b_v)$ is the maximum weight independent set among all vertices u, such that $b_v < c_u$ and $d_v < a_u$ (since the algorithm visits the point $p = d_v$ after all vertices $u \in U$ have been processed). Therefore, since the value $\widetilde{w}_v$ equals the total weight of the unbounded vertices u, such that v is a hovering vertex of u and $c_v < c_u < b_v$, it follows that the total weight of S equals $(w(u) + \widetilde{w}_v) + w(Opt(U, b_v)) = (w(u) + \widetilde{w}_v) + W(m) + w(H[b_v, p_m))$, which equals the value $W(v)$ computed in line 17 of Algorithm 3. That is, $W(v)$ equals the maximum weight of an independent set, which includes the bounded vertex v and vertices of the set $U' = \{u \in V \mid a_v < a_u, c_v < c_u\}$.

Case 3. $p = a_v$ for some bounded vertex $v \in V_B$. Then, v is being processed at this iteration, i.e. after this iteration the vertices of the set $U \cup \{v\}$ have been processed. Let m^* be the leftmost marker of M to the right of c_v on L_1 (cf. line 19 of Algorithm 3). Note that the induction hypothesis implies that $Opt(U, y) = W(m'_y) + w(H[y, p_{m'_y}))$.

Suppose first that $c_v < y$. Then for every vertex $u \in U \cup \{v\}$ such that $y \leq c_u$, it follows that $u \neq v$, i.e. $u \in U$. Therefore $Opt(U \cup \{v\}, y) = Opt(U, y)$. Furthermore $m'_y = m_y$, since the markers of M to the right of c_v (and thus also to the right of y) remain the same before and after the process of v. Therefore, the induction hypothesis implies that $Opt(U \cup \{v\}, y) = Opt(U, y) = W(m_y) + w(H[y, p_{m_y}))$, since $m'_y = m_y$.

Suppose now that $y \leq c_v$. Let S_v be a maximum weight independent set, which includes the bounded vertex v and vertices $u \in U$ such that $y \leq c_u$. We will prove that the total weight of the vertices of S_v is $w(S_v) = W(v) + w(H[y, c_v))$. First note that, since all vertices of U have been processed by the algorithm before v, it follows in particular that $a_v < a_u$ for every $u \in S_v \setminus \{v\}$. Suppose that $c_u < c_v$ for a bounded vertex $u \in S_v \setminus \{v\}$. Then $\overline{T}_u \cap \overline{T}_v \neq \emptyset$, and thus also $T_u \cap T_v \neq \emptyset$, since both u and v are bounded. That is, $uv \in E$, which is a contradiction, since S_v is an independent set. Therefore $c_v < c_u$ for every bounded vertex $u \in S_v \setminus \{v\}$. Thus, every vertex $u \in S_v \setminus \{v\}$ with $c_u < c_v$ is an unbounded vertex that corresponds to a marker of the set $H[y, c_v)$. Recall by the analysis of Case 2 that $W(v)$ equals the maximum weight of an independent set, which includes the bounded vertex v and vertices of the set $U' = \{u \in V \mid a_v < a_u, c_v < c_u\}$. Furthermore, Lemma 1 implies that $T_u \cap T_{u'} = \emptyset$, i.e. $uu' \notin E$, for every unbounded vertex $u \in H[y, c_v)$ and every vertex u' with $c_v \leq c_{u'}$. Therefore, since we assumed that $v \in S_v$, it follows that $w(S_v) = W(v) + w(H[y, c_v))$.

For the sequel of the analysis for Case 3, we will compare the total weight $w(S_v)$ of S_v with the weight $w(Opt(U, y))$ of the independent set $Opt(U, y)$ that does not include vertex v. It is easy to see that if $w(S_v) \geq w(Opt(U, y))$, then $w(Opt(U \cup \{v\}, y)) = w(S_v)$. Furthermore, if $w(S_v) \leq w(Opt(U, y))$, then $w(Opt(U \cup \{v\}, y)) = w(Opt(U, y))$.

Consider the case where $W(v) \leq W(m^*) + w(H[c_v, p_{m^*}))$. Then the algorithm does not modify the sets M and H (cf. line 20 of Algorithm 3), and thus in particular $m_y = m'_y$. Therefore, the induction hypothesis implies that $w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$, since $m_y = m'_y$. Recall that $y \leq c_v$ and $c_v < p_{m^*}$, and thus $y < p_{m^*}$. Therefore, $y \leq m_y \leq p_{m^*}$ by definition of m_y, and thus Lemma 7 implies that

$W(m_y) \geq W(m^*) + w(H[p_{m_y}, p_{m^*}))$ (note that here the equality holds only if $m_y = m^*$). Therefore, since $w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$, it follows that $w(Opt(U, y)) \geq W(m^*) + w(H[y, p_{m^*}))$, i.e. $w(Opt(U, y)) \geq W(m^*) + w(H[y, c_v)) + w(H[c_v, p_{m^*}))$. Furthermore, since $W(m^*) + w(H[c_v, p_{m^*})) \geq W(v)$ by our assumption on $W(v)$, it follows that $w(Opt(U, y)) \geq W(v) + w(H[y, c_v)) = w(S_v)$. That is, $w(Opt(U, y)) \geq w(S_v)$, and thus $w(Opt(U \cup \{v\}, y)) = w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$.

Consider now the case where $W(v) > W(m^*) + w(H[c_v, p_{m^*}))$. Then, the algorithm adds a marker m' to the set M at the point $p_{m'} = c_v$ of the line L_1, while the weight of m' is $W(m') = W(v)$ (cf. lines 21 and 22 of Algorithm 3). Moreover, the algorithm removes all markers $m'' \in M$, such that $p_{m''} < p_{m'}$ and $W(m'') \leq W(m') + w(H[p_{m''}, p_{m'}))$ (cf. line 23 of Algorithm 3). Recall that $y \leq p_{m'} = c_v$, and thus $y \leq p_{m_y} \leq p_{m'}$ by definition of m_y. Then, either $y \leq p_{m_y} < p_{m'}$ or $y \leq p_{m_y} = p_{m'}$, as we distinguish in the following cases.

Case 3a. $y \leq p_{m_y} < p_{m'}$. Then $W(m_y) > W(m') + w(H[p_{m_y}, p_{m'}))$, since otherwise the marker m_y would be removed from M after the addition of m' to M (cf. line 23 of Algorithm 3). We will now prove that $m'_y = m_y$, i.e. that m_y was the leftmost marker of M to the right of y, also before the process of v. Suppose otherwise that $m'_y \neq m_y$, i.e. $p_{m'_y} < p_{m_y}$ and the marker m'_y has been removed from M during the process of v (cf. line 23 of Algorithm 3). Then, since $p_{m'_y} < p_{m_y}$, Lemma 7 implies that $W(m'_y) > W(m_y) + w(H[p_{m'_y}, p_{m_y}))$ before the process of vertex v. Therefore, since $W(m_y) > W(m') + w(H[p_{m_y}, p_{m'}))$, it follows that $W(m'_y) > W(m') + w(H[p_{m'_y}, p_{m'}))$, and thus m'_y has not been removed from M during the process of v in line 23, which is a contradiction. Thus $m'_y = m_y$. Therefore the induction hypothesis implies that $w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$, since $m'_y = m_y$. Furthermore, since $W(m_y) > W(m') + w(H[p_{m_y}, p_{m'}))$, it follows that $w(Opt(U, y)) > W(m') + w(H[y, p_{m'}))$. Therefore, since $W(m') = W(v)$ and $p_{m'} = c_v$, it follows that $w(Opt(U, y)) > W(v) + w(H[y, c_v)) = w(S_v)$, and thus $w(Opt(U \cup \{v\}, y)) = w(Opt(U, y)) = W(m_y) + w(H[y, p_{m_y}))$.

Case 3b. $y \leq p_{m_y} = p_{m'}$. Then, since the marker $m_y = m'$ was not in the set M before the process of v, it follows that either $y \leq p_{m'_y} < p_{m_y} = p_{m'}$ or $y \leq p_{m_y} = p_{m'} < p_{m'_y}$.

Suppose first that $y \leq p_{m'_y} < p_{m_y} = p_{m'}$. Then, the marker m'_y is removed from M during the process of v, since m_y is after the process of v the leftmost marker of M to the right of y on the line L_1. Thus $W(m'_y) \leq W(m') + w(H[p_{m'_y}, p_{m'}))$ (cf. line 23 of Algorithm 3). Recall that $w(Opt(U, y)) = W(m'_y) + w(H[y, p_{m'_y}))$ by the induction hypothesis, and thus $w(Opt(U, y)) \leq W(m') + w(H[y, p_{m'}))$. Therefore, since $W(m_y) = W(m') = W(v)$ and $p_{m_y} = p_{m'} = c_v$, it follows that $w(Opt(U, y)) \leq W(v) + w(H[y, c_v)) = w(S_v)$, and thus $w(Opt(U \cup \{v\}, y)) = w(S_v) = W(m_y) + w(H[y, p_{m_y}))$.

Suppose finally that $y \leq p_{m_y} = p_{m'} < p_{m'_y}$. Recall that m^* is the leftmost marker of M to the right of c_v on L_1 (cf. line 19 of Algorithm 3). Then $m^* = m'_y$ by the definition of the markers m^* and m'_y, since the marker $m' = m_y$ was not in the set M before the process of v. Recall that $W(m') = W(v) > W(m^*) + w(H[c_v, p_{m^*}))$, since otherwise the marker m' would not be inserted to M, which is a contradiction. Therefore, since $m^* = m'_y$ and $c_v = p_{m'}$, it follows that $W(m') > W(m'_y) + w(H[p_{m'}, p_{m'_y}))$. Recall that $w(Opt(U, y)) = W(m'_y) + w(H[y, p_{m'_y}))$ by the induction hypothesis, and thus $w(Opt(U, y)) = W(m'_y) + w(H[y, p_{m'})) + w(H[p_{m'}, p_{m'_y}))$, i.e. $w(Opt(U, y)) < W(m') + w(H[y, p_{m'}))$. Therefore, since $W(m_y) = W(m') = W(v)$ and $p_{m_y} = p_{m'} = c_v$, it follows that $w(Opt(U, y)) < W(v) + w(H[y, c_v)) = w(S_v)$, and thus $w(Opt(U \cup \{v\}, y)) = w(S_v) = W(m_y) + w(H[y, p_{m_y}))$.

Summarizing Cases 1, 2, and 3, it follows that the condition of the lemma holds also after the iteration of the algorithm, during which the algorithm visits the point $p \in \{a_v, d_v \mid v \in V\}$ on the line L_2. This completes the induction step and the lemma follows. ☐

We are now ready to present the main theorem of this section.

Theorem 5. *A maximum weight independent set of a multitolerance graph $G = (V, E)$ with n vertices and m edges can be computed using Algorithm 3 in $O(m + n \log n)$ time.*

Proof. Let $c = \min\{c_v \mid v \in V\} - 1$ and m be the leftmost marker of M on the line L_1 after the last iteration of the algorithm, i.e. after all vertices of G have been processed. Recall by Definition 10 that $Opt(V, c)$ denotes a maximum weight independent set, which includes vertices $u \in V$ such that $c \leq c_u$, and thus $Opt(V, c)$ is a maximum weight independent set of G. Then, Lemma 8 implies that the returned value $W(m) + w(H[c, p_m))$ equals the weight $w(Opt(V, c))$ of $Opt(V, c)$. This implies the correctness of Algorithm 3.

Algorithm 3 can be implemented to run in $O(|E| + n \log n)$ time, as follows. First, for the initialization phase of lines 3-8 of the algorithm, we sort in $O(n \log n)$ time all unbounded vertices $u \in V_U$ increasingly according to their endpoints c_u on the line L_1. As we described in the preamble of Algorithm 3, we can store the unbounded vertices in an AVL-tree, such that for any two points q, q' on the line L_1 we can compute in $O(\log n)$ time the total weight of the unbounded vertices $u \in V_U$ with $c_u \in [q, q')$ (cf. the weight $w(H[q, q'))$ in the set H of markers). Thus, the initialization of the value $\widetilde{w}_v$ in line 5 of the algorithm can be done in $O(\log n)$ time for every $v \in V_B$. Then, for every $v \in V_B$ we need $O(|N(v)|)$ time to execute lines 6-8 of the algorithm. Note that $\sum_{v \in V_B} |N(v)| = O(|E|)$. Therefore, the initialization phase of lines 3-8 needs $O(|E| + n \log n)$ time.

We now analyze the time complexity of lines 10-23 in the main part of the algorithm. As we described in the preamble of the algorithm, the insertion and the location of a marker in the sets M and H can be executed in $O(\log n)$ time if we implement the sets M and H with AVL-trees. Therefore, each of the lines 11, 13, 16, 19, and 21 of the algorithm can be executed in $O(\log n)$ time. Furthermore, as we described in the preamble of the algorithm, we can compute the value $w(H[q, q'))$ for two given points q, q' of the line L_1 in $O(\log n)$ time. Therefore, lines 17 and 20 of the algorithm can be also executed in $O(\log n)$ time each.

Consider now the lines 14 and 23 of Algorithm 3. Recall by Corollary 1 that for any two markers $m_1, m_2 \in M$ after an iteration of Algorithm 3, $p_{m_2} < p_{m_1}$ if and only if $W(m_2) - w(H[p_{m_2}, +\infty)) > W(m_1) - w(H[p_{m_1}, +\infty))$, i.e. the values $W(m) - w(H[p_m, +\infty))$ for the markers $m \in M$ are increasing from right to left on the line L_1. Note that this monotonicity property on the set M is restored in the line 14 (resp. in the line 23) of Algorithm 3 when a new marker is added to the set H (resp. when a new marker is added to the set M).

Consider first the restoration of the monotonicity property in line 14, when a new marker h is added to the set H at the position $p_h = c_v$, for some unbounded vertex $v \in V_U$ (cf. line 11). Let m^* be the leftmost marker of M to the right of $p_h = c_v$ (cf. line 13). Then, for every marker m' to the left of m^*, the value $W(m') - w(H[p_{m'}, +\infty))$ decreases by $w(h)$ after the addition of h to H. In line 14 we want to find and remove from M all markers m' to the left of m^*, such that $W(m') - w(H[p_{m'}, +\infty)) \leq W(m^*) - w(H[p_{m^*}, +\infty))$ after the addition of h to H, or equivalently $W(m') - w(H[p_{m'}, +\infty)) \leq w(h) + W(m^*) - w(H[p_{m^*}, +\infty))$ before the addition of h to H.

Consider now the restoration of the monotonicity property in line 23, when a new marker m' is added to the set M at the position $p_{m'} = c_v$, for some bounded vertex $v \in V_B$ (cf. line 21). Let m^* be the leftmost marker of M to the right of c_v (cf. line 19). Then, since $W(m') = W(v) > W(m^*) + w(H[c_v, p_{m^*}))$ (cf. lines 20 and 22), the monotonicity property of Corollary 1 holds also after the addition of m' to M (cf. line 21) for the markers to the right of m' on the line L_1. In line 23 we want to find and remove from M all markers m'' to the left of m', such that $W(m'') - w(H[p_{m''}, +\infty)) \leq W(m') - w(H[p_{m'}, +\infty))$. Equivalently, before the addition of m' to M, we want to find and remove from M all markers all markers m'' to the left of m^*, such that $W(m'') - w(H[p_{m''}, +\infty)) \leq W(v) - w(H[c_v, +\infty))$.

Therefore, in both lines 14 and 23 of Algorithm 3, we want to find and remove from M all markers all markers m to the left of m^*, such that the value $W(m) - w(H[p_m, +\infty))$ is smaller than or equal to a given query value Q. This can be implemented as follows. Except for the two AVL-trees that store the values of M and of H, respectively, we also maintain in a balanced binary search tree, for instance an AVL-tree, the markers of the union $M \cup H$ on the line L_1. Denote by $x.left$ and $x.right$ the left and the right child of an arbitrary internal vertex of this tree, respectively. Furthermore, at every internal vertex x of the AVL-tree of $M \cup H$ that stores a marker $m \in M$ (resp. a marker $h \in H$), we store except the weight $W(m)$ (resp. $w(h)$) also a label $\ell(x)$, which equals the total weight of the markers of H that are stored in the subtree rooted at x. Observe here that the value $w(H[p_m, +\infty))$ for any marker $m \in M$ can be computed as follows: in the path of the AVL-tree from the root to the internal vertex x that stores m, we sum up the values $(\ell(x) - \ell(x.left))$ and $(\ell(y) - \ell(y.left))$, for every internal vertex of this path, at which we follow the left child.

We first find in $O(\log n)$ time in the AVL-tree of M the first marker m^{**} of M to the left of m^* on the line L_1. Then, we locate in $O(\log n)$ time the marker m^{**} in the AVL-tree of $M \cup H$. Furthermore, following in this tree the path from the root to the internal vertex that stores m^{**}, we compute in $O(\log n)$ time the value $w(H[p_{m^{**}}, +\infty))$ by the labels $\ell(x)$ of the internal vertices x. If $W(m^{**}) + w(H[p_{m^{**}}, +\infty)) \leq Q$, where Q is the given query value, then we remove the marker m^{**} from M (in both AVL-trees for M and for $M \cup H$, respectively). We iterate until there is no marker of M to the left of m^* on the line L_1, or until $W(m^{**}) + w(H[p_{m^{**}}, +\infty)) > Q$ for the first marker m^{**} of M to the left of m^*. In the latter case there is no other marker of M to remove (cf. lines 14 and 23 of Algorithm 3), since the values $W(m) - w(H[p_m, +\infty))$ for the markers $m \in M$ are increasing from right to left on the line L_1.

Furthermore, we can update in $O(\log n)$ time the labels $\ell(x)$ of the internal vertices x of the AVL-tree of $M \cup H$, whenever the sets H or M are modified, as follows. If a marker m is inserted to or removed from M, cf. the lines 14, 21, and 23 of Algorithm 3, then we need to update a constant number of labels $\ell(x)$ during each one of at most $O(\log n)$ "trinode restructure" operations (see [12] for more details). Suppose now that a marker h is inserted to H, cf. line 11 of Algorithm 3 (that is, h is inserted in the AVL-tree that stores the markers of H in line 11). Note that we add h to the AVL-tree that stores the markers of $M \cup H$ after the update of the set M in line 14. Then, we update similarly a constant number of labels $\ell(x)$ during each one of at most $O(\log n)$ "trinode restructure" operations (see [12]). Moreover, following the path from the internal vertex x that stores h to the root, we add the weight $w(h)$ to the label $\ell(y)$ for every internal vertex y of this path that includes vertex x in the subtree rooted at y.

Summarizing, we can implement in $O(\log n)$ time the removal of one marker from M in the lines 14 and 23 of Algorithm 3. Thus, since during the execution of Algorithm 3 we need to remove at most $O(n)$ markers from the set M, the total time needed to execute lines 14 and 23 for all iterations of the algorithm is $O(n \log n)$. Recall now that the execution of each of the lines 11, 13, 16, 17, 19, 20, and 21 of the algorithm can be done in $O(\log n)$ time. Therefore, since the lines 10-23 of the algorithm are executed $O(n)$ times, the execution of the lines 9-23 can be done in $O(n \log n)$ time. Furthermore, the line 24 can be executed in $O(\log n)$ time by just locating the leftmost marker $m \in M$ and computing the value $w(H[c, p_m))$. Therefore, since the initialization phase of lines 3-8 needs $O(|E| + n \log n)$ time, it follows that Algorithm 3 computes the value of a maximum weight independent set of G in $O(|E| + n \log n)$ time.

After computing the value $w(Opt(V, c))$ of a maximum weight independent set $Opt(V, c)$ in G, we can easily compute in $O(n \log n)$ time the set $Opt(V, c)$ itself, instead of its value, as follows. Initially compute the set of unbounded vertices $\{u \in V_U \mid c_u \in (c, p_m)\}$ and set $Opt(V, c)$ to be equal to this set. Note that this set of unbounded vertices has total weight $w(H[c, p_m))$ (cf. line 24 of Algorithm 3). Then, visit sequentially all markers of M from left to right. For

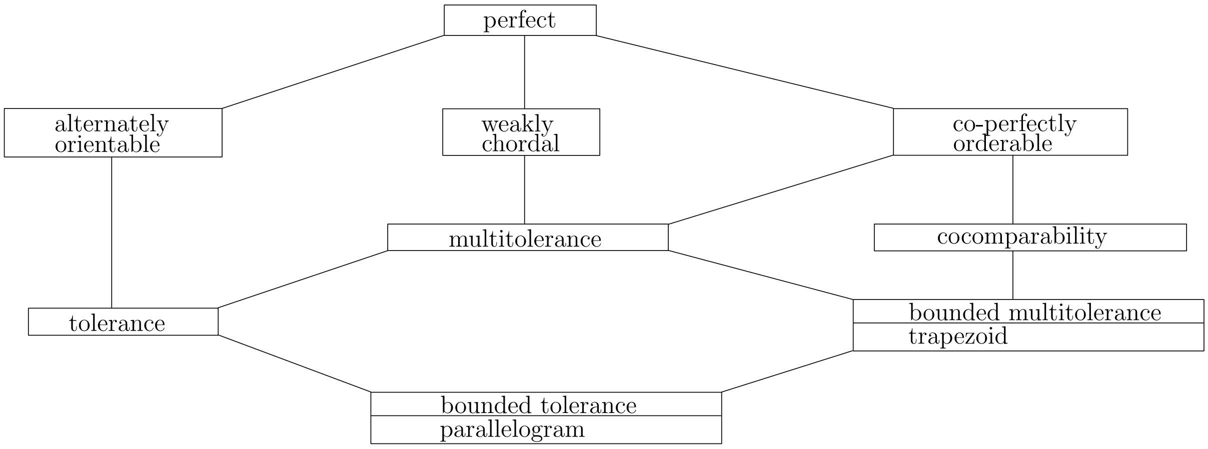


Fig. 5. The classification of multitolerance graphs in the hierarchy of perfect graphs in [11]. This hierarchy is complete, i.e. every inclusion is strict.

every $m \in M$, which is placed at the point $p_m = c_v$ on the line L_1 for some bounded vertex $v \in V_B$, we augment the current set $Opt(V,c)$ by $\{v\}$. Moreover, compute the set of unbounded vertices $\{u \in V_U \mid c_u \in (c_v, b_v), u \notin N(v)\}$ and augment $Opt(V,c)$ by this set. Note that this set of unbounded vertices has total weight $\widetilde{w}_v$ (cf. the lines 5-8 of Algorithm 3). Furthermore, for every two consecutive markers m_1, m_2 in M, where $p_{m_1} = c_{v_1} < p_{m_2}$ for a bounded vertex $v_1 \in V_B$, compute the set of unbounded vertices $\{u \in V_u \mid c_u \in (b_{v_1}, p_{m_2}), u \notin N(v_1)\}$ and augment $Opt(V,c)$ by this set. Note that the latter set of unbounded vertices has total weight $w(H[b_{v_1}, p_{m_2}))$ (cf. the line 17 of Algorithm 3). After these computations, it follows easily that the computed set $Opt(V,c)$ is a maximum weight independent set of G.

Regarding the complexity of these computations, we need linear time $O(n)$ to traverse M from left to right. For the computation of the above $O(n)$ sets of unbounded vertices, we partition the unbounded vertices $u \in V_U$ (i.e. the markers of the set H) into $|M|+1 = O(n)$ subsets, according to the position of the endpoints c_u on the line L_1. We can determine each of these subsets in $O(\log n)$ time using binary search in the AVL-tree that stores the set H. After a partition has been determined, we need constant time for every vertex $u \in V_U$ in this partition. Therefore, after the termination of Algorithm 3, we can compute in $O(n \log n)$ time a maximum weight independent set $Opt(V,c)$ of G, instead of its value. This completes the proof of the theorem. $\square$

6 Classification of multitolerance graphs

In this section we classify the class of multitolerance graphs inside the hierarchy of perfect graphs given in Figure 2.8 of [11][†]. The resulting hierarchy of classes of perfect graphs is *complete*, i.e. all inclusions are strict. This hierarchy is schematically presented in Figure 5. We prove these results by using the trapezoepiped representation of multitolerance graphs presented in Section 2, as well as some known results on the hierarchy of perfect graphs given in [11].

Theorem 6. *Multitolerance graphs* strictly *include tolerance and trapezoid graphs, while they are* strictly *included in weakly chordal graphs.*

[†]It was claimed in [25] (in Theorem 3.1(b)) that the class of tolerance graphs is *strictly* included in the class of multitolerance graphs; however, in the proof of that theorem only inclusion has been shown, and not strict inclusion. We prove strict inclusion in Theorem 6. Moreover, it has been correctly shown in [25] that a multitolerance graph does not contain any chordless cycle C_n, where $n \geq 5$. We prove in Theorem 6 that actually the same holds also for the complements $\overline{C}_n$ of C_n, where $n \geq 5$, and thus every multitolerance graph is weakly chordal.

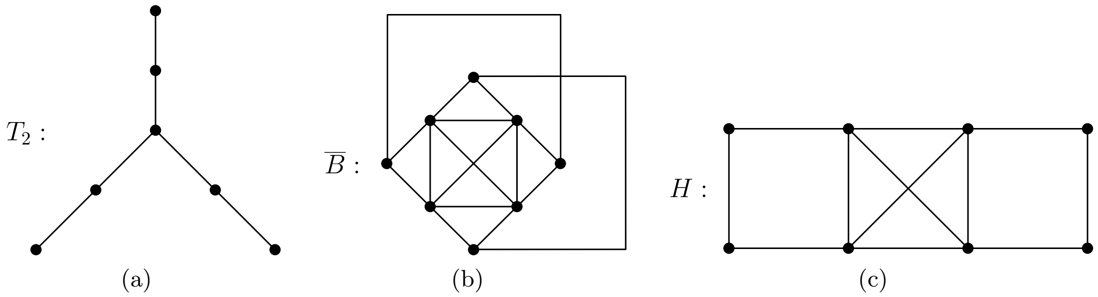


Fig. 6. The graphs (a) T_2, (b) $\overline{B}$ (the Berlin graph), and (c) H.

Proof. First recall that any tolerance graph $G = (V, E)$ is also a multitolerance graph, in which the left and right tolerances of every vertex coincide, i.e. $t_{v,1} = t_{v,2}$ for every $v \in V$ (cf. Definition 2). Furthermore, a graph G is a trapezoid graph if and only if G is a bounded multitolerance graph [25] (also known as bounded bitolerance graph [11]). Therefore, any trapezoid graph G is also a multitolerance graph. The graph T_2, which is illustrated in Figure 6(a), is a tolerance but not a trapezoid graph [11]. Furthermore, the Berlin graph $\overline{B}$, which is illustrated in Figure 6(b), is a trapezoid but not a tolerance graph [11]. Therefore, the graph T_2 is a multitolerance but not a trapezoid graph, while the graph $\overline{B}$ is a multitolerance but not a tolerance graph, and thus multitolerance graphs strictly include both tolerance and trapezoid graphs.

We will now prove that every multitolerance graph is weakly chordal. To this end, we have to prove by the definition of weakly chordal graphs that every multitolerance graph G has no induced subgraph isomorphic to the chordless cycle C_n, with n vertices, or to its complement $\overline{C_n}$, for any $n \geq 5$. Since multitolerance graphs are hereditary, it suffices to prove that neither C_n nor its complement $\overline{C_n}$ are multitolerance graphs, for any $n \geq 5$. Suppose otherwise that C_n (resp. $\overline{C_n}$) is a multitolerance graph, and let R be a trapezoepiped representation of C_n (resp. of $\overline{C_n}$). Since bounded multitolerance graphs, i.e. trapezoid graphs, are known to be weakly chordal [11], it follows that C_n (resp. $\overline{C_n}$) is not bounded multitolerance. Therefore, there exists at least one inevitable unbounded vertex v in the trapezoepiped representation R of C_n (resp. of $\overline{C_n}$). Let u be a hovering vertex of v. Then $uv \notin E$ by Definition 6 and $N(v) \subseteq N(u)$ by Lemma 4. However, it is easy to verify that for any $n \geq 5$, the graph C_n (resp. $\overline{C_n}$) has no pair of non-adjacent vertices u and v, such that $N(v) \subseteq N(u)$. This is a contradiction, and thus neither C_n nor its complement $\overline{C_n}$ are multitolerance graphs, for any $n \geq 5$. Therefore every multitolerance graph is weakly chordal.

Consider finally the complement $\overline{H}$ of the graph H that is illustrated in Figure 6(c). The graph $\overline{H}$ is a weakly chordal but not a trapezoid graph [11]. Suppose that $\overline{H}$ is a multitolerance graph and let R be a trapezoepiped representation of $\overline{H}$. Since $\overline{H}$ is not a trapezoid (i.e. bounded multitolerance) graph, it follows that R has at least one inevitable unbounded vertex v in the trapezoepiped representation R of $\overline{H}$. Let u be a hovering vertex of v. Then $uv \notin E$ by Definition 6 and $N(v) \subseteq N(u)$ by Lemma 4. It is now easy to verify that the graph $\overline{H}$ has no pair of non-adjacent vertices u and v, such that $N(v) \subseteq N(u)$. This is a contradiction, and thus $\overline{H}$ is not a multitolerance graph. That is, $\overline{H}$ is a weakly chordal but not a multitolerance graph, and thus multitolerance graphs are strictly included in weakly chordal graphs. This completes the proof of the theorem. □

Theorem 7. *Multitolerance graphs are* strictly *included in co-perfectly orderable graphs.*

Proof. First we prove that the complement $\overline{G}$ of every multitolerance graph G is perfectly orderable, using the trapezoepiped representation of multitolerance graphs. Let $G = (V, E)$ be a multitolerance graph and R be a trapezoepiped representation of G. Recall that for every vertex $v \in V$, T_v denotes the trapezoepiped of v in the trapezoepiped representation R, while the trapezoid $\overline{T}_v$ denotes the projection of T_v on the plane $z = 0$, cf. Figures 1 and 2. We define the linear order $\prec$ of a the vertex set V as follows: for every $u, v \in V$, $u \prec v$ if and only if $a_v < a_u$ in the representation R. We will prove that $\prec$ is a perfect order of the complement $\overline{G}$ of G. Suppose otherwise that there exists an induced path $v_1v_2v_3v_4$ of $\overline{G}$, such that $v_1 \prec v_2$ and $v_4 \prec v_3$. Note that, since $v_1v_2v_3v_4$ is an induced path of $\overline{G}$, it follows that $v_2v_4v_1v_3$ is an induced path of G.

Since $v_1v_2 \notin E$, it follows that $T_{v_1} \cap T_{v_2} = \emptyset$. Suppose that $\overline{T}_{v_1} \cap \overline{T}_{v_2} \neq \emptyset$. Then, at least one of the vertices v_1 and v_2 is unbounded, since otherwise $T_{v_1} \cap T_{v_2} \neq \emptyset$, which is a contradiction. Furthermore, either replacing T_{v_1} by $H_{\text{convex}}(\overline{T}_{v_1}, a'_{v_1}, c'_{v_1})$ or replacing T_{v_2} by $H_{\text{convex}}(\overline{T}_{v_2}, a'_{v_2}, c'_{v_2})$ in R creates the new edge v_1v_2 in G. That is, either v_2 is a hovering vertex of the unbounded vertex v_1 or v_1 is a hovering vertex of the unbounded vertex v_2. If v_2 is a hovering vertex of the unbounded vertex v_1, then $N(v_1) \subseteq N(v_2)$ by Lemma 4. This is a contradiction, since $v_3 \in N(v_1) \setminus N(v_2)$. Therefore v_1 is a hovering vertex of the unbounded vertex v_2. Furthermore $a_{v_2} < a_{v_1}$ by the definition of the vertex order $\prec$, since $v_1 \prec v_2$. If $v_1 \in V_B$, then Lemma 2 implies that $T_{v_1} \cap T_{v_2} \neq \emptyset$ if and only if $\overline{T}_{v_1} \cap \overline{T}_{v_2} \neq \emptyset$. This is a contradiction, since we $T_{v_1} \cap T_{v_2} = \emptyset$ and $\overline{T}_{v_1} \cap \overline{T}_{v_2} \neq \emptyset$ (as we assumed). Suppose that $v_1 \in V_U$, i.e. both v_1 and v_2 are unbounded. Then, since $a_{v_2} < a_{v_1}$ and $\overline{T}_{v_1} \cap \overline{T}_{v_2} \neq \emptyset$, it follows that $c_{v_1} < c_{v_2}$, and thus in particular $\phi_{v_1} > \phi_{v_2}$, i.e. $\Delta - \cot \phi_{v_1} > \Delta - \cot \phi_{v_2}$. Therefore, the line T_{v_1} lies above the line T_{v_2} in R, and thus replacing T_{v_2} by $H_{\text{convex}}(\overline{T}_{v_2}, a'_{v_2}, c'_{v_2})$ in R does not create the new edge v_1v_2 in G. That is, v_1 is not a hovering vertex of the unbounded vertex v_2, which is a contradiction. Therefore $\overline{T}_{v_1} \cap \overline{T}_{v_2} = \emptyset$, i.e. either $\overline{T}_{v_2}$ lies completely to the left or completely to the right of $\overline{T}_{v_1}$. Thus, since $a_{v_2} < a_{v_1}$, it follows that $\overline{T}_{v_2}$ lies completely to the left of $\overline{T}_{v_1}$.

Similarly to the case of v_1 and v_2, it follows for the symmetric case of v_4 and v_3 that $\overline{T}_{v_3} \cap \overline{T}_{v_4} = \emptyset$, i.e. either $\overline{T}_{v_3}$ lies completely to the left or completely to the right of $\overline{T}_{v_4}$. Furthermore $a_{v_3} < a_{v_4}$ by the definition of the vertex order $\prec$, since $v_4 \prec v_3$. Therefore, $\overline{T}_{v_3}$ lies completely to the left of $\overline{T}_{v_4}$.

Consider now the vertices v_2 and v_3. Since $v_2v_3 \notin E$, it follows that $T_{v_2} \cap T_{v_3} = \emptyset$. Suppose that $\overline{T}_{v_2} \cap \overline{T}_{v_3} \neq \emptyset$. Then, at least one of the vertices v_2 and v_3 is unbounded, since otherwise $T_{v_2} \cap T_{v_3} \neq \emptyset$, which is a contradiction. Furthermore, either replacing T_{v_2} by $H_{\text{convex}}(\overline{T}_{v_2}, a'_{v_2}, c'_{v_2})$ or replacing T_{v_3} by $H_{\text{convex}}(\overline{T}_{v_3}, a'_{v_3}, c'_{v_3})$ in R creates the new edge v_2v_3 in G. That is, either v_3 is a hovering vertex of the unbounded vertex v_2 or v_2 is a hovering vertex of the unbounded vertex v_3. If v_3 is a hovering vertex of the unbounded vertex v_2, then $N(v_2) \subseteq N(v_3)$ by Lemma 4. This is a contradiction, since $v_4 \in N(v_2) \setminus N(v_3)$. On the other hand, if v_2 is a hovering vertex of the unbounded vertex v_3, then $N(v_3) \subseteq N(v_2)$ by Lemma 4. This is a contradiction, since $v_1 \in N(v_3) \setminus N(v_2)$. Therefore $\overline{T}_{v_2} \cap \overline{T}_{v_3} = \emptyset$, i.e. either $\overline{T}_{v_2}$ lies completely to the left or completely to the right of $\overline{T}_{v_3}$.

Suppose that $\overline{T}_{v_2}$ lies completely to the left of $\overline{T}_{v_3}$. Then, since $\overline{T}_{v_3}$ lies completely to the left of $\overline{T}_{v_4}$, as we proved above, it follows that $\overline{T}_{v_2}$ lies completely to the left of $\overline{T}_{v_4}$, and thus $\overline{T}_{v_2} \cap \overline{T}_{v_4} = \emptyset$ and $T_{v_2} \cap T_{v_4} = \emptyset$. This is a contradiction, since $v_2v_4 \in E$. On the other hand, suppose that $\overline{T}_{v_2}$ lies completely to the right of $\overline{T}_{v_3}$, i.e. $\overline{T}_{v_3}$ lies completely to the right of $\overline{T}_{v_2}$. Then, since $\overline{T}_{v_2}$ lies completely to the left of $\overline{T}_{v_1}$, as we proved above, it follows that $\overline{T}_{v_3}$ lies completely to the left of $\overline{T}_{v_1}$, and thus $\overline{T}_{v_3} \cap \overline{T}_{v_1} = \emptyset$ and $T_{v_3} \cap T_{v_1} = \emptyset$. This is a contradiction, since $v_3v_1 \in E$.

Summarizing, there exists no induced path $v_1v_2v_3v_4$ of $\overline{G}$, such that $v_1 \prec v_2$ and $v_4 \prec v_3$. Therefore the vertex order $\prec$ is perfect, i.e. the complement $\overline{G}$ of G is perfectly orderable. Finally, the complement $\overline{C}_6$ of the chordless cycle C_6 with six vertices is a co-perfectly orderable but not a tolerance graph [11]. Moreover, $\overline{C_6}$ is not a multitolerance graph, since every multitolerance graph is weakly chordal by Theorem 6. Therefore, $\overline{C_6}$ is a co-perfectly orderable but not a multitolerance graph, and thus multitolerance graphs are strictly included in co-perfectly orderable graphs. □

Theorem 8. *Multitolerance graphs are* incomparable *with alternately orientable and with cocomparability graphs.*

Proof. The Berlin graph $\overline{B}$, which is illustrated in Figure 6(b), is a trapezoid but not an alternately orientable graph [11]. Furthermore the graph T_2, which is illustrated in Figure 6(a), is a tolerance but not a cocomparability graph [11]. Therefore, since trapezoid (resp. tolerance) graphs are also multitolerance graphs (cf. Theorem 6), it follows that $\overline{B}$ (resp. T_2) is a multitolerance but not an alternately orientable (resp. cocomparability) graph.

On the other hand, the chordless cycle C_6 with six vertices is an alternately orientable graph, while its complement $\overline{C_6}$ is a cocomparability graph [11]. However, C_6 and $\overline{C_6}$ are not weakly chordal by the definition of weakly chordal graphs, and thus C_6 and $\overline{C_6}$ are also not multitolerance graphs by Theorem 6. Therefore, C_6 (resp. $\overline{C_6}$) is an alternately orientable (resp. cocomparability) but not a multitolerance graph, and thus multitolerance graphs are incomparable with alternately orientable and with cocomparability graphs. □

7 Conclusions and further research

In this article we proposed the first non-trivial intersection model for general multitolerance graphs, given by objects in the three-dimensional space, called trapezoepipeds. This trapezoepiped representation unifies in a simple and intuitive way the well known trapezoid representation for bounded multitolerance graphs and the recently introduced parallelepiped representation for tolerance graphs in [22]. Using this representation, we presented three new efficient algorithms that compute a minimum coloring, a maximum clique, and a maximum weight independent set on a multitolerance graph, respectively. The running times of the first two algorithms are optimal, while the third algorithm also improves the best known running time for the maximum weight independent set on tolerance graphs [22]. To the best of our knowledge, no algorithm for these problems on multitolerance graphs has appeared until now, and thus the previously best known running times for these problems were those of perfect graphs. Furthermore, we proved several structural results on the class of multitolerance graphs, which complement the hierarchy of perfect graphs given in [11]. It can be expected that the proposed intersection model will prove useful for deriving new algorithmic and structural results on multitolerance graphs. It is important to note here that the recognition problem for general multitolerance graphs, remains an interesting open problem. On the contrary, it is known that trapezoid (i.e. bounded multitolerance) graphs can be recognized efficiently [19, 21], while it is NP-complete to recognize tolerance and bounded tolerance graphs [23].

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. K. P. Bogart and G. Isaak. Proper and unit bitolerance orders and graphs. *Discrete Mathematics*, 181(1-3):37–51, 1998.
3. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey.* Society for Industrial and Applied Mathematics (SIAM), 1999.
4. A. H. Busch. A characterization of triangle-free tolerance graphs. *Discrete Applied Mathematics*, 154(3):471–477, 2006.

5. S. Felsner. Tolerance graphs and orders. *Journal of Graph Theory*, 28(3):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. M. C. Golumbic. *Algorithmic graph theory and perfect graphs (Annals of Discrete Mathematics, Vol. 57)*. North-Holland Publishing Co., 2004.
8. M. C. Golumbic and C. L. 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.
9. M. C. Golumbic, C. L. Monma, and W. T. Trotter. Tolerance graphs. *Discrete Applied Mathematics*, 9(2):157–170, 1984.
10. M. C. Golumbic and A. Siani. Coloring algorithms for tolerance graphs: reasoning and scheduling with interval constraints. In *Proceedings of the Joint International Conferences on Artificial Intelligence, Automated Reasoning, and Symbolic Computation (AISC/Calculemus)*, pages 196–207, 2002.
11. M. C. Golumbic and A. N. Trenk. *Tolerance Graphs*. Cambridge studies in advanced mathematics, 2004.
12. M. T. Goodrich and R. Tamassia. *Algorithm Design: Foundations, Analysis, and Internet Examples*. John Wiley & Sons, Inc., 2002.
13. M. Grötshcel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
14. R. B. Hayward and R. Shamir. A note on tolerance graph recognition. *Discrete Applied Mathematics*, 143(1-3):307–311, 2004.
15. J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.
16. G. Isaak, K. L. Nyman, and A. N. Trenk. A hierarchy of classes of bounded bitolerance orders. *Ars Combinatoria*, 69, 2003.
17. 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.
18. L. Langley. *Interval tolerance orders and dimension*. PhD thesis, Dartmouth College, 1993.
19. T.-H. Ma and J. P. Spinrad. On the 2-chain subgraph cover and related problems. *Journal of Algorithms*, 17(2):251–268, 1994.
20. T. A. McKee and F. R. McMorris. *Topics in intersection graph theory*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999.
21. G. B. Mertzios and D. G. Corneil. Vertex splitting and the recognition of trapezoid graphs. Technical Report AIB-2009-16, Department of Computer Science, RWTH Aachen University, September 2009.
22. G. B. Mertzios, I. Sau, and S. Zaks. A new intersection model and improved algorithms for tolerance graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1800–1813, 2009. A preliminary version appeared in the *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 285–295, 2009.
23. G. B. Mertzios, I. Sau, and S. Zaks. The recognition of tolerance and bounded tolerance graphs. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 585–596, 2010.
24. G. Narasimhan and R. Manber. Stability and chromatic number of tolerance graphs. *Discrete Applied Mathematics*, 36:47–56, 1992.
25. A. Parra. Triangulating multitolerance graphs. *Discrete Applied Mathematics*, 84(1-3):183–197, 1998.
26. S. P. Ryan. Trapezoid order classification. *Order*, 15:341–354, 1998.

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`

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

2008-19 Dirk Wilking: Empirical Studies for the Application of Agile Methods to Embedded Systems

2009-02 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications

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

2009-04 Daniel Klünder: Entwurf eingebetteter Software mit abstrakten Zustandsmaschinen und Business Object Notation

2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs

2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete

2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I

- 2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs
- 2009-09 Michael Nett: Implementation of an Automated Proof for an Algorithm Solving the Maximum Independent Set Problem
- 2009-10 Felix Reidl, Fernando Sánchez Villaamil: Automatic Verification of the Correctness of the Upper Bound of a Maximum Independent Set Algorithm
- 2009-11 Kyriaki Ioannidou, George B. Mertzios, Stavros D. Nikolopoulos: The Longest Path Problem is Polynomial on Interval Graphs
- 2009-12 Martin Neuhäußer, Lijun Zhang: Time-Bounded Reachability in Continuous-Time Markov Decision Processes
- 2009-13 Martin Zimmermann: Time-optimal Winning Strategies for Poset Games
- 2009-14 Ralf Huuck, Gerwin Klein, Bastian Schlich (eds.): Doctoral Symposium on Systems Software Verification (DS SSV'09)
- 2009-15 Joost-Pieter Katoen, Daniel Klink, Martin Neuhäußer: Compositional Abstraction for Stochastic Systems
- 2009-16 George B. Mertzios, Derek G. Corneil: Vertex Splitting and the Recognition of Trapezoid Graphs
- 2009-17 Carsten Kern: Learning Communicating and Nondeterministic Automata
- 2009-18 Paul Hänsch, Michaela Slaats, Wolfgang Thomas: Parametrized Regular Infinite Games and Higher-Order Pushdown Strategies
- 2010-02 Daniel Neider, Christof Löding: Learning Visibly One-Counter Automata in Polynomial Time

* These reports are only available as a printed version.

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