Using ILP/SAT to determine pathwidth, visibility representations, and other grid-based graph drawings*

Therese Biedl[1], Thomas Bläsius[2], Benjamin Niedermann[2], Martin Nöllenburg[2], Roman Prutkin[2], and Ignaz Rutter[2]

[1] David R. Cheriton School of Computer Science, University of Waterloo, Canada
[2] Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Germany

Abstract. We present a simple and versatile formulation of grid-based graph representation problems as an integer linear program (ILP) and a corresponding SAT instance. In a grid-based representation vertices and edges correspond to axis-parallel boxes on an underlying integer grid; boxes can be further constrained in their shapes and interactions by additional problem-specific constraints. We describe a general d-dimensional model for grid representation problems. This model can be used to solve a variety of NP-hard graph problems, including pathwidth, bandwidth, optimum st-orientation, area-minimal (bar-k) visibility representation, boxicity-k graphs and others. We implemented SAT-models for all of the above problems and evaluated them on the Rome graphs collection. The experiments show that our model successfully solves NP-hard problems within few minutes on small to medium-size Rome graphs.

1 Introduction

Integer linear programming (ILP) and Boolean satisfiability testing (SAT) are indispensable and widely used tools in solving many hard combinatorial optimization and decision problems in practical applications [3,9]. In graph drawing, especially for planar graphs, these methods are not frequently applied. A few notable exceptions are crossing minimization [8, 10, 19, 22], orthogonal graph drawing with vertex and edge labels [4] and metro-map layout [26]. Recent work by Chimani et al. [11] uses SAT formulations for testing upward planarity. All these approaches have in common that they exploit problem-specific properties to derive small and efficiently solvable models, but they do not generalize to larger classes of problems.

In this paper we propose a generic ILP model that is flexible enough to capture a large variety of different grid-based graph layout problems, both polynomially-solvable and NP-complete. We demonstrate this broad applicability by adapting the base model to six different NP-complete example problems: pathwidth, bandwidth, optimum st-orientation, minimum area bar- and bar k-visibility representation, and boxicity-k testing. For minimum-area visibility representations and boxicity this is, to the best of our knowledge, the first implementation of an exact solution method. Of course this flexibility comes at the cost of losing some of the efficiency of more specific approaches.

* T. Biedl is supported by NSERC, M. Nöllenburg is supported by the Concept for the Future of KIT under grant YIG 10-209.

Our goal, however, is not to achieve maximal performance for a specific problem, but to provide an easy-to-adapt solution method for a larger class of problems, which allows quick and simple prototyping for instances that are not too large. Our ILP models can be translated into equivalent SAT formulations, which exhibit better performance in the implementation than the ILP models themselves. We illustrate the usefulness of our approach by an experimental evaluation that applies our generic model to the above six NP-complete problems using the well-known Rome graphs [1] as a benchmark set. Our evaluation shows that, depending on the problem, our model can solve small to medium-size instances (sizes varying from about 25 vertices and edges for bar-1 visibility testing up to more than 250 vertices and edges, i.e., all Rome graphs, for optimum st-orientation) to optimality within a few minutes. In Section 2, we introduce generic grid-based graph representations and formulate an ILP model for d-dimensional integer grids. We show how this model can be adapted to six concrete one-, two- and d-dimensional grid-based layout problems in Sections 3 and 4. In Section 5 we evaluate our implementations and report experimental results. The implementation is available from `i11www.iti.kit.edu/gdsat`. Omitted proofs are in the full version [2].

2 Generic Model for Grid-Based Graph Representations

In this section we explain how to express d-dimensional boxes in a d-dimensional integer grid as constraints of an ILP or a SAT instance. In the subsequent sections we use these boxes as basic elements for representing vertices and edges in problem-specific ILP and SAT models. Observe that we can restrict ourselves to boxes in integer grids.

Lemma 1. *Any set I of n boxes in $\mathbb{R}^d$ can be transformed into another set I' of n closed boxes on the integer grid $\{1,\dots,n\}^d$ such that two boxes intersect in I if and only if they intersect in I'.*

2.1 Integer Linear Programming Model

We will describe our model in the general case for a d-dimensional integer grid, where $d \geq 1$. Let $\mathcal{R}^d = [1, U_1] \times \ldots \times [1, U_d]$ be a bounded d-dimensional integer grid, where $[A, B]$ denotes the set of integers $\{A, A+1, \ldots, B-1, B\}$. In a *grid-based graph representation* vertices and/or edges are represented as d-dimensional boxes in $\mathcal{R}^d$. A *grid box* R in $\mathcal{R}^d$ is a subset $[s_1, t_1] \times \ldots \times [s_d, t_d]$ of $\mathcal{R}^d$, where $1 \leq s_k \leq t_k \leq U_k$ for all $1 \leq k \leq d$. In the following we describe a set of ILP constraints that together create a non-empty box for some object v. We denote this ILP as $\mathcal{B}(d)$.

We first extend $\mathcal{R}^d$ by a margin of dummy points to $\bar{\mathcal{R}}^d = [0, U_1 + 1] \times \ldots \times [0, U_d + 1]$. We use three sets of binary variables:

$$x_{\mathbf{i}}(v) \in \{0,1\} \qquad \forall \mathbf{i} \in \bar{\mathcal{R}}^d \tag{1}$$

$$b_i^k(v) \in \{0,1\} \qquad \forall 1 \leq k \leq d \text{ and } 1 \leq i \leq U_k \tag{2}$$

$$e_i^k(v) \in \{0,1\} \qquad \forall 1 \leq k \leq d \text{ and } 1 \leq i \leq U_k \tag{3}$$

Variables $x_{\mathbf{i}}(v)$ indicate whether grid point $\mathbf{i}$ belongs to the box representing v ($x_{\mathbf{i}}(v) = 1$) or not ($x_{\mathbf{i}}(v) = 0$). Variables $b_i^k(v)$ and $e_i^k(v)$ indicate whether the box of v may start

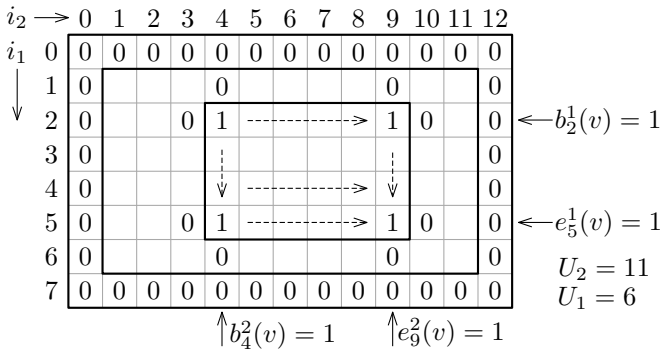


Fig. 1. Example of a 2-dimensional 8×13 grid $\bar{\mathcal{R}}^d$ with a 4×6 grid box and the corresponding variable assignments.

or end at position i in dimension k. We use $\mathbf{i}[k]$ to denote the k-th coordinate of grid point $\mathbf{i} \in \mathcal{R}^d$ and $\mathbf{1_k} = (0, \ldots, 0, 1, 0, \ldots, 0)$ to denote the k-th d-dimensional unit vector. If $d = 1$ we will drop the dimension index of the variables to simplify the notation. The following constraints model a box in $\mathcal{R}^d$ (see Fig. 1 for an example):

$$x_{\mathbf{i}}(v) = 0 \qquad \forall \mathbf{i} \in \bar{\mathcal{R}}^d \setminus \mathcal{R}^d \tag{4}$$

$$\sum_{\mathbf{i} \in \bar{\mathcal{R}}^d} x_{\mathbf{i}}(v) \geq 1 \tag{5}$$

$$\sum_{i \in [1, U_k]} b_i^k(v) = 1 \qquad \forall 1 \leq k \leq d \tag{6}$$

$$\sum_{i \in [1, U_k]} e_i^k(v) = 1 \qquad \forall 1 \leq k \leq d \tag{7}$$

$$x_{\mathbf{i}-\mathbf{1_k}}(v) + b_{\mathbf{i}[k]}^k(v) \geq x_{\mathbf{i}}(v) \qquad \forall \mathbf{i} \in \mathcal{R}^d \text{ and } 1 \leq k \leq d \tag{8}$$

$$x_{\mathbf{i}}(v) \leq x_{\mathbf{i}+\mathbf{1_k}}(v) + e_{\mathbf{i}[k]}^k(v) \qquad \forall \mathbf{i} \in \mathcal{R}^d \text{ and } 1 \leq k \leq d \tag{9}$$

Constraint (4) creates a margin of zeroes around $\mathcal{R}^d$. Constraint (5) ensures that the shape representing v is non-empty, and constraints (6) and (7) provide exactly one start and end position in each dimension. Finally, due to constraints (8) and (9) each grid point inside the specified bounds belongs to v and all other points don't.

Lemma 2. *The ILP $\mathcal{B}(d)$ defined by constraints (1)–(9) correctly models all non-empty grid boxes in $\mathcal{R}^d$.*

Our example ILP models in Sections 3 and 4 extend ILP $\mathcal{B}(d)$ by constraints controlling additional properties of vertex and edge boxes. For instance, boxes can be easily constrained to be single points, to be horizontal or vertical line segments, to intersect if and only if they are incident or adjacent in G, to meet in endpoints etc. The definition of an objective function for the ILP depends on the specific problem at hand and will be discussed in the problem sections. In the full version [2] we explain how to translate the ILP $\mathcal{B}(d)$ into an equivalent SAT formulation with better practical performance.

3 One-dimensional Problems

In the following, let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. One-dimensional grid-based graph representations can be used to model vertices as intersecting intervals (one-dimensional boxes) or as disjoint points that induce a certain vertex order. We present ILP models for three such problems.

3.1 Pathwidth

The pathwidth of a graph G is a well-known graph parameter with many equivalent definitions. We use the definition via the smallest clique size of an interval supergraph. More precisely, a graph is an *interval graph* if it can be represented as intersection graph of 1-dimensional intervals. A graph G has *pathwidth* $pw(G) \leq p$ if there exists an interval graph H that contains G as a subgraph and for which all cliques have size at most $p+1$. It is NP-hard to compute the pathwidth of an arbitrary graph and even hard to approximate it [5]. There are fixed-parameter algorithms for computing the pathwidth, e.g. [6], however, we are not aware of any implementations of these algorithms. The only available implementations are exponential-time algorithms, e.g., in sage[3].

Problem 1 (Pathwidth). Given a graph $G = (V, E)$, determine the pathwidth of G, i.e., the smallest integer p so that $pw(G) \leq p$.

There is an interesting connection between pathwidth and planar graph drawings of small height. Any planar graph that has a planar drawing of height h has pathwidth at most h [18]. Also, pathwidth is a crucial ingredient in testing in polynomial time whether a graph has a planar drawing of height h [15].

We create a one-dimensional grid representation of G, in which every vertex is an interval and every edge forces the two vertex intervals to intersect. The objective is to minimize the maximum number of intervals that intersect in any given point. We use the ILP $\mathcal{B}(1)$ for a grid $\mathcal{R} = [1, n]$, which already assigns a non-empty interval to each vertex $v \in V$. We add binary variables for the edges of G, a variable $p \in \mathbb{N}$ representing the pathwidth of G, and a set of additional constraints as follows.

$$x_i(e) \in \{0, 1\} \qquad \forall i \in \mathcal{R} \text{ and } e \in E \tag{10}$$

$$\sum_{i \in \mathcal{R}} x_i(e) \geq 1 \qquad \forall e \in E \tag{11}$$

$$x_i(uv) \leq x_i(u) \qquad x_i(uv) \leq x_i(v) \qquad \forall uv \in E \tag{12}$$

$$\sum_{v \in V} x_i(v) \leq p + 1 \qquad \forall i \in \mathcal{R} \tag{13}$$

Our objective function is to minimize the value of p subject to the above constraints.

It is easy to see that every edge must be represented by some grid point (constraint (11)), and can only use those grid points, where the two end vertices intersect (constraint (12)). Hence the intervals of vertices define some interval graph H that is

[3] www.sagemath.org

a supergraph of G. Constraint (13) enforces that at most $p+1$ intervals meet in any point, which by Helly's property means that H has clique-size at most $p+1$. So G has pathwidth at most p. By minimizing p we obtain the desired result. In our implementation we translate the ILP into a SAT instance. We test satisfiability for fixed values of p, starting with $p=1$ and increasing it incrementally until a solution is found.

Theorem 1. *There exists an ILP/SAT formulation with $O(n(n+m))$ variables and $O(n(n+m))$ constraints / $O(n^3 + n\binom{n}{p+2})$ clauses of maximum size n that has a solution of value $\leq p$ if and only if G has pathwidth $\leq p$.*

With some easy modifications, the above ILP can be used for testing whether a graph is a (proper) interval graph. Section 4.2 shows that boxicity-d graphs, the d-dimensional generalization of interval graphs, can also be recognized by our ILP.

3.2 Bandwidth

The bandwidth of a graph G with n vertices is another classic graph parameter, which is NP-hard to compute [12]; due to the practical importance of the problem there are also a few approaches to find exact solutions to the bandwidth minimization problem. For example, [14] and [25] use the branch-and-bound technique combined with various heuristics. We present a solution that can be easily described using our general framework. However, regarding the running time, it cannot be expected to compete with techniques specially tuned for solving the bandwidth minimization problem.

Let $f : V \to \{1, \ldots, n\}$ be a bijection that defines a linear vertex order. The *bandwidth* of G is defined as $bw(G) = \min_f \max\{f(v) - f(u) \mid uv \in E \text{ and } f(u) < f(v)\}$, i.e., the minimum length of the longest edge in G over all possible vertex orders.

In the full version [2] we describe an ILP that assigns the vertices of G to disjoint grid points and requires for an integer k that any pair of adjacent vertices is at most k grid points apart, i.e., we can test if $bw(G) \leq k$.

Theorem 2. *There exists an ILP/SAT formulation with $O(n^2)$ variables and $O(n \cdot m)$ constraints / $O(n^3)$ clauses of maximum size n that has a solution if and only if G has bandwidth $\leq k$.*

3.3 Optimum st-orientation

Let G be an undirected graph and let s and t be two vertices of G with $st \in E$. An *st-orientation* of G is an orientation of the edges such that s is the unique source and t is the unique sink [17]. Such an orientation can exist only if $G \cup (s,t)$ is biconnected. Computing an st-orientation can be done in linear time [7, 17], but it is NP-complete to find an st-orientation that minimizes the length of the longest path from s to t, even for planar graphs [28]. It has many applications in graph drawing [27] and beyond.

Problem 2 (Optimum st-orientation). Given a graph $G = (V, E)$ and two vertices $s, t \in V$ with $st \in E$, find an orientation of E such that s is the only source, t is the only sink, and the length of the longest directed path from s to t is minimum.

In the full version [2] we formulate an ILP using points for vertices and non-degenerate intervals for edges that computes a *height-k st-orientation* of G, i.e., an st-orientation such that the longest path has length at most k (if one exists).

Theorem 3. *There exists an ILP with $O(n(n+m))$ variables and constraints that computes an optimum st-orientation. Alternatively, there exists an ILP/SAT formulation with $O(k(n+m))$ variables and $O(k(n+m))$ constraints / $O(k^2(n+m))$ clauses of maximum size n that has a solution if and only if a height-k st-orientation of G exists.*

4 Higher-Dimensional Problems

In this section we give examples of two-dimensional visibility graph representations and a d-dimensional grid-based graph representation problem. Let again $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$.

4.1 Visibility representations

A visibility representation (also: *bar visibility representation* or *weak visibility representation*) of a graph $G = (V, E)$ maps all vertices to disjoint horizontal line segments, called *bars*, and all edges to disjoint vertical bars, such that for each edge $uv \in E$ the bar of uv has its endpoints on the bars for u and v and does not intersect any other vertex bar. Visibility representations are an important visualization concept in graph drawing, e.g., it is well known that a graph is planar if and only if it has a visibility representation [29, 30]. An interesting recent extension are bar k-visibility representations [13], which additionally allow edges to intersect at most k non-incident vertex bars. We use our ILP to compute compact visibility and bar k-visibility representations. Minimizing the area of a visibility representation is NP-hard [24] and we are not aware of any implemented exact algorithms to solve the problem for any $k \geq 0$. By Lemma 1 we know that all bars can be described with integer coordinates of size $O(m+n)$.

Problem 3 (Bar k-Visibility Representation). Given a graph G and an integer $k \geq 0$, find a bar k-visibility representation on an integer grid of size $H \times W$ (if one exists).

Bar visibility representations. Our goal is to test whether G has a visibility representation in a grid with H columns and W rows (and thus minimize H or W). We set $\mathcal{R}^2 = [1, H] \times [1, W]$ and use ILP $\mathcal{B}(2)$ to create grid boxes for all edges and vertices in G. We add one more set of binary variables for vertex-edge incidences and the following constraints.

$$x_{\mathbf{i}}(e, v) \in \{0, 1\} \qquad \forall \mathbf{i} \in \mathcal{R}^2 \, \forall e \in E \, \forall v \in e \tag{14}$$

$$b_i^1(v) = e_i^1(v) \qquad \forall i \in [1, U_1] \, \forall v \in V \tag{15}$$

$$b_i^2(e) = e_i^2(e) \qquad \forall i \in [1, U_2] \, \forall e \in E \tag{16}$$

$$\sum_{v \in V} x_{\mathbf{i}}(v) \leq 1 \qquad \forall \mathbf{i} \in \mathcal{R}^2 \tag{17}$$

$$\sum_{v \in V \setminus e} x_{\mathbf{i}}(v) \leq (1 - x_{\mathbf{i}}(e)) \qquad \forall \mathbf{i} \in \mathcal{R}^2 \, \forall e \in E \tag{18}$$

$$x_{\mathbf{i}}(e,v) \le x_{\mathbf{i}}(e) \qquad x_{\mathbf{i}}(e,v) \le x_{\mathbf{i}}(v) \quad \forall \mathbf{i} \in \mathcal{R}^2\, \forall e \in E\, \forall v \in e \tag{19}$$

$$\sum_{\mathbf{i} \in \mathcal{R}^2} x_{\mathbf{i}}(e,v) \ge 1 \qquad \forall e \in E\, \forall v \in e \tag{20}$$

$$x_{\mathbf{i}}(e,v) \le b^1_{\mathbf{i}[1]}(e) + e^1_{\mathbf{i}[1]}(e) \qquad \forall \mathbf{i} \in \mathcal{R}^2\, \forall e \in E\, \forall v \in e \tag{21}$$

Constraints (15) and (16) ensure that all vertex boxes are horizontal bars of height 1 and all edge boxes are vertical bars of width 1. Constraint (17) forces the vertex boxes to be disjoint; edge boxes will be implicitly disjoint (for a simple graph) due to the remaining constraints. No edge is allowed to intersect a non-incident vertex due to constraint (18). Finally, we need to set the new incidence variables $x_{\mathbf{i}}(e,v)$ for an edge e and an incident vertex v so that $x_{\mathbf{i}}(e,v) = 1$ if and only if e and v share the grid point $\mathbf{i}$. Constraints (19) and (20) ensure that each incidence in G is realized in at least one grid point, but it must be one that is used by the boxes of e and v. Finally, constraint (21) requires edge e to start and end at its two intersection points with the incident vertex boxes. This constraint is optional, but yields a tighter formulation.

Since every graph with a visibility representation is planar (and vice versa) we have $m \in O(n)$. Moreover, our ILP and SAT models can also be used to test planarity of a given graph by setting $H = n$ and $W = 2n - 4$, which is sufficient due to Tamassia and Tollis [29]. This might not look interesting at first sight since planarity testing can be done in linear time [21]. However, we think that this is still useful as one can add other constraints to the ILP model, e.g., to create simultaneous planar embeddings, and use it as a subroutine for ILP formulations of applied graph drawing problems such as metro maps [26] and cartograms.

Theorem 4. *There is an ILP/SAT formulation with $O(HWn)$ variables and $O(HWn)$ constraints / $O(HWn^2)$ clauses of maximum size HW that solves Problem 3 for $k = 0$.*

Bar k-visibility representations. It is easy to extend our previous model for $k = 0$ to test bar k-visibility representations for $k \ge 1$. See [2] for a detailed description.

Theorem 5. *There exists an ILP/SAT formulation with $O(HW(n+m))$ variables and $O(HW(m^2+n))$ constraints / $O(\binom{HW}{k+1}m + HWm^2)$ clauses of maximum size HW that solves Problem 3 for $k \ge 1$.*

4.2 Boxicity-d graphs

A graph is said to have *boxicity* d if it can be represented as intersection graph of d-dimensional axis-aligned boxes. Testing whether a graph has boxicity d is NP-hard, even for $d = 2$ [23]. We are not aware of any implemented algorithms to determine the boxicity of a graph. By Lemma 1 we can restrict ourselves to a grid of side length n. In the full version [2], we give an ILP model for testing whether a graph has boxicity d.

Theorem 6. *There exists an ILP with $O(n^d(n+m))$ variables and $O(n^{d+2})$ constraints as well as a SAT instance with $O(n^d(n+m))$ variables and $O(n^{d+2})$ clauses of maximum size $O(n^d)$ to test whether a graph G has boxicity d.*

5 Experiments

We implemented and tested our formulation for minimizing pathwidth, bandwidth, length of longest path in an st-orientation, and width of bar-visibility and bar 1-visibility representations, as well as deciding whether a graph has boxicity 2.

We performed the experiments on a single core of an AMD Opteron 6172 processor running Linux 3.4.11. The machine is clocked at 2.1 Ghz, and has 256 GiB RAM. Our implementation (available from `http://i11www.iti.kit.edu/gdsat`) is written in C++ and was compiled with GCC 4.7.1 using optimization `-O3`. As test sample we used the *Rome graphs* dataset [1] which consists of 11533 graphs with vertex number between 10 and 100. 18% of the Rome graphs are planar. The size distribution of the Rome graphs can be found in the full version [2].

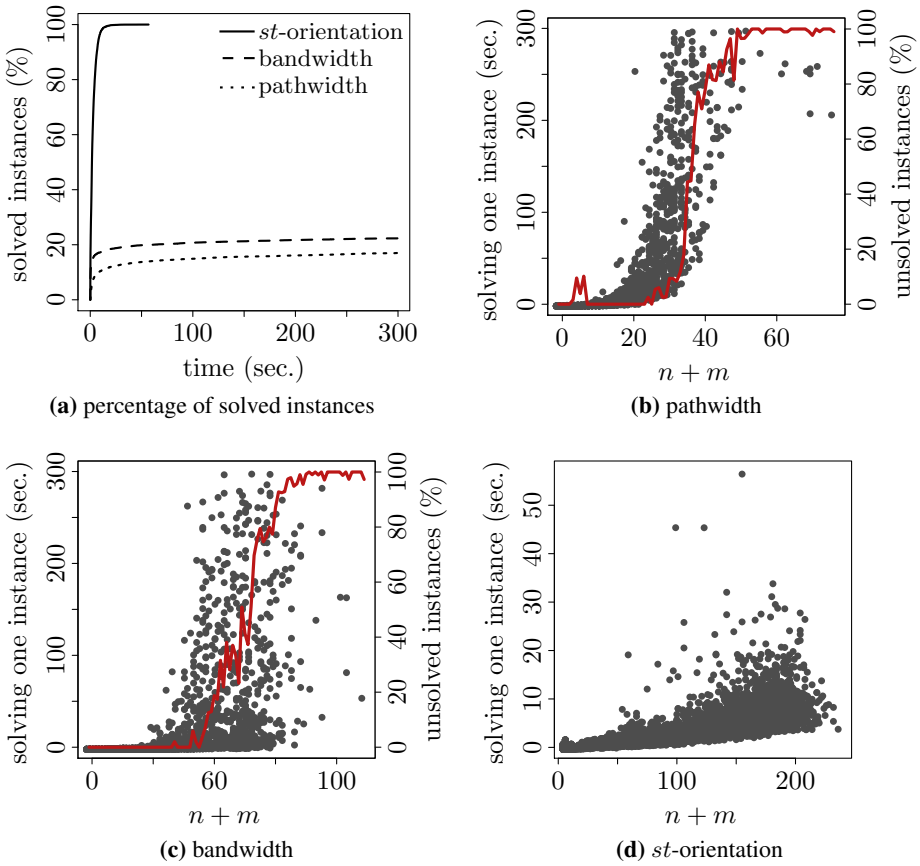
We initially used the *Gurobi* solver [20] to test the implementation of the ILP formulations, however it turned out that even for very small graphs ($n < 10$) solving a single instance can take minutes. We therefore focused on the equivalent SAT formulations gaining a significant speed-up. As SAT solver we used *MiniSat* [16] in version 2.2.0. For each of the five minimization problems we determined obvious lower and upper bounds in $O(n)$ for the respective graph parameter. Starting with the lower bound we iteratively increased the parameter to the next integer until a solution was found (or a predefined timeout was exceeded). Each iteration consists of constructing the SAT formulation and executing the SAT solver. We measured the total time spent in all iterations. For boxicity 2 we decided to consider square grids and minimize their side lengths. Thus the same iterative procedure applies to boxicity 2.

Note that for all considered problems a binary search-like procedure for the parameter value did not prove to be efficient, since the solver usually takes more time with increasing parameter value, which is mainly due to the increasing number of variables and clauses. For the one-dimensional problems we used a timeout of 300 seconds, for the two-dimensional problems of 600 seconds.

We ran the instances sorted by size $n + m$ starting with the smallest graphs. If more than 400 consecutive graphs in this order produced timeouts, we ended the experiment prematurely and evaluated only the so far obtained results. Figures 2 and 3 summarize our experimental results and show the percentage of Rome graphs solved within the given time limit, as well as scatter plots with each solved instance represented as a point depending on its graph size and the required computation time.

Pathwidth. As Fig. 2a shows, we were able to compute the pathwidth for 17.0% of all Rome graphs, from which 82% were solved within the first minute and only 3% within the last. Therefore, we expect that a significant increase of the timeout value would be necessary for a noticeable increase of the percentage of solved instances. We note that almost all small graphs ($n + m < 45$) could be solved within the given timeout, however, for larger graphs, the percentage of solved instances rapidly drops, as the red curve in Fig. 2b shows. Almost no graphs with $n + m > 70$ were solved.

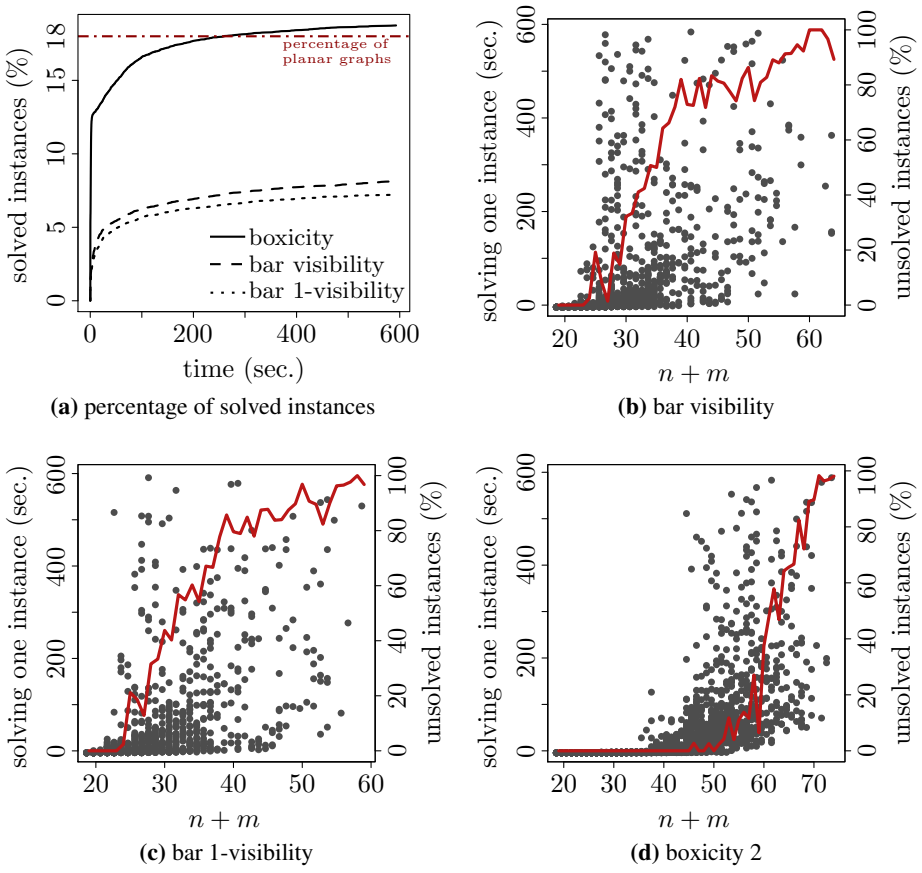
Bandwidth. We were able to compute the bandwidth for 22.3% of all Rome graphs (see Fig. 2a), from which 90% were solved within the first minute and only 1.3% within the last. Similarly to the previous case, the procedure terminated successfully within 300 seconds for almost all small graphs ($n + m < 55$ in this case), while almost none of the larger graphs ($n + m > 80$) were solved; see the red curve in Fig. 2c.



(a) percentage of solved instances **(b)** pathwidth **(c)** bandwidth **(d)** st-orientation

Fig. 2. Experimental results for the one-dimensional problems. (a) Percentage of solved instances. (b)–(d): Time in seconds for solving an instance (dots) and percentage of instances not solved within 300 seconds (red curves), both in relation to $n + m$

Optimum st-orientation. Note that very few of the Rome graphs are biconnected. Therefore, to test our SAT implementation for computing the minimum number of levels in an st-orientation, we subdivided each graph into biconnected blocks and removed those with $n \leq 2$, which produced 13606 blocks in total ($3 \leq n + m \leq 230$). Then, for each such block, we randomly selected one pair of vertices s, t, $s \neq t$, connected them by an edge if it did not already exist and ran the iterative procedure. In this way, for the respective choice of s, t we were able to compute the minimum number of levels in an st-orientation for all biconnected blocks; see Fig. 2a. Moreover, no graph took longer than 57 seconds, for 97% of the graphs it took less than 10 seconds and for 68% less than 3 seconds. Even for the biggest blocks with $m + n > 200$, the procedure successfully terminated within 15 seconds in 93% of the cases; see Fig. 2d.



(a) percentage of solved instances

(b) bar visibility

(c) bar 1-visibility

(d) boxicity 2

Fig. 3. Experimental results for the two-dimensional problems. (a) Percentage of solved instances. The red horizontal line shows the percentage of planar graphs over all Rome graphs. (b)–(d): Time in seconds for solving an instance (dots) and percentage of instances not solved within 600 seconds (red curves), both in relation to $n + m$.

Bar visibility. To compute bar-visibility representations of minimum width, we iteratively tested for each graph all widths W between 1 and $2n - 4$. We used the trivial upper bound $H = n$ for the height. We were able to compute solutions for 28.5% of all 3281 planar Rome graphs (see Fig. 3a), 69% of which were solved within the first minute and less than 0.1% within the last. We were able to solve all small instances with $n + m \leq 23$ and almost none for $n + m > 55$; see the red curve in Fig. 3b.

Bar 1-visibility. We also ran width minimization for bar 1-visibility representations on all Rome graphs. The procedure terminated successfully within the given time for 833 graphs (7.2% of all Rome graphs), which is close to the corresponding number for bar-visibility; see Fig. 3a. For bar 1-visibility, eight graphs were solved which were not solved for bar-visibility. Interestingly, they were all planar. All but 113 graphs suc-

cessfully processed in the previous experiment were also successfully processed in this one. A possible explanation for those 113 graphs is that the SAT formulation for bar 1-visibility requires more clauses. All small graphs with $n + m \leq 23$ were processed successfully. Interestingly, for none of the processed graphs the minimum width actually decreased in comparison to their minimum-width bar-visibility representation.

Boxicity-2. For testing boxicity 2, we started with a 3×3 grid for each graph and then increased height and width simultaneously after each iteration. Within the specified timeout of 600 seconds, we were able to decide whether a graph has boxicity 2 for 18.7% of all Rome graphs (see Fig. 3b), 82% of which were processed within the first minute and 0.3% within the last. All of the successfully processed graphs actually had boxicity 2. Small graphs with $n + m \leq 50$ were processed almost completely, while almost none of the graphs with $n + m > 70$ finished; see Fig. 3d.

6 Conclusion

We presented a versatile ILP formulation for determining placement of grid boxes according to problem-specific constraints. We gave six examples of how to extend this formulation for solving numerous NP-hard graph drawing and representation problems. Our experimental evaluation showed that while solving the original ILP is rather slow, the derived SAT formulations perform well for smaller graphs. While our approach is not suitable to replace faster specialized exact or heuristic algorithms, it does provide a simple-to-use tool for solving problems that can be modeled by grid-based graph representations with little implementation effort. This can be useful, e.g., for verifying counterexamples, NP-hardness gadgets, or for solving certain instances in practice.

Many other problems can easily be formulated as ILPs by assigning grid-boxes to vertices or edges. Among those are, e.g., testing whether a planar graph has a straight-line drawing of height h, whether a planar graph has a rectangular dual with integer coordinates and prescribed integral areas, whether a graph is a t-interval graph, or whether a bipartite graph can be represented as a planar bus graph. Important open problems are to reduce the complexity of our formulations and whether approximation algorithms for graph drawing can be derived from our model via fractional relaxation.

References

1. Rome graphs. `www.graphdrawing.org/download/rome-graphml.tgz`
2. Biedl, T., Bläsius, T., Niedermann, B., Nöllenburg, M., Prutkin, R., Rutter, I.: A versatile ILP/SAT formulation for pathwidth, optimum st-orientation, visibility representation, and other grid-based graph drawing problems. CoRR abs/1308.6778 (2013)
3. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. IOS Press (2009)
4. Binucci, C., Didimo, W., Liotta, G., Nonato, M.: Orthogonal drawings of graphs with vertex and edge labels. Comput. Geom. Theory Appl. 32(2), 71–114 (2005)
5. Bodlaender, H.L., Gilbert, J., Hafsteinsson, H., Kloks, T.: Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. J. Algorithms 18(2), 238–255 (1995)
6. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithms 21(2), 358–402 (1996)

7. Brandes, U.: Eager st-ordering. In: Möhring, R., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 247–256. Springer (2002)
8. Buchheim, C., Chimani, M., Ebner, D., Gutwenger, C., Jünger, M., Klau, G.W., Mutzel, P., Weiskircher, R.: A branch-and-cut approach to the crossing number problem. Discrete Optimization 5(2), 373–388 (2008)
9. Chen, D.S., Batson, R.G., Dang, Y.: Applied Integer Programming. Wiley (2010)
10. Chimani, M., Mutzel, P., Bomze, I.: A new approach to exact crossing minimization. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 284–296. Springer (2008)
11. Chimani, M., Zeranski, R.: Upward planarity testing via SAT. In: Didimo, W., Patrignani, M. (eds.) Graph Drawing 2012. LNCS, vol. 7704, pp. 248–259. Springer (2013)
12. Chinn, P.Z., Chvátalova, J., Dewdney, A.K., Gibbs, N.E.: The bandwidth problem for graphs and matrices—a survey. J. Graph Theory 6(3), 223–254 (1982)
13. Dean, A.M., Evans, W., Gethner, E., Laison, J.D., Safari, M.A., Trotter, W.T.: Bar k-visibility graphs. J. Graph Algorithms Appl. 11(1), 45–59 (2007)
14. Del Corso, G.M., Manzini, G.: Finding exact solutions to the bandwidth minimization problem. Computing 62(3), 189–203 (1999)
15. Dujmovic, V., Fellows, M.R., Kitching, M., Liotta, G., McCartin, C., Nishimura, N., Ragde, P., Rosamond, F.A., Whitesides, S., Wood, D.R.: On the parameterized complexity of layered graph drawing. Algorithmica 52(2), 267–292 (2008)
16. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) Proc. Theory and Appl. of Satisfiability Testing (SAT'03), LNCS, vol. 2919, pp. 502–518. Springer (2004)
17. Even, S., Tarjan, R.E.: Computing an st-numbering. Theoret. Comput. Sci. 2(3), 339–344 (1976)
18. Felsner, S., Liotta, G., Wismath, S.: Straight-line drawings on restricted integer grids in two and three dimensions. J. Graph Algorithms Appl. 7(4), 363–398 (2003)
19. Gange, G., Stuckey, P.J., Marriott, K.: Optimal k-level planarization and crossing minimization. In: Brandes, U., Cornelsen, S. (eds.) Graph Drawing 2010. LNCS, vol. 6502, pp. 238–249. Springer (2011)
20. Gurobi Optimization, Inc.: Gurobi optimizer reference manual (2013)
21. Hopcroft, J., Tarjan, R.: Efficient planarity testing. J. ACM 21(4), 549–568 (Oct 1974)
22. Jünger, M., Mutzel, P.: 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. J. Graph Algorithms Appl. 1(1), 1–25 (1997)
23. Kratochvíl, J.: A special planar satisfiability problem and a consequence of its NP-completeness. Discrete Appl. Math. 52(3), 233–252 (Aug 1994)
24. Lin, X., Eades, P.: Towards area requirements for drawing hierarchically planar graphs. Theoret. Comput. Sci. 292(3), 679–695 (2003)
25. Martí, R., Campos, V., Piñana, E.: A branch and bound algorithm for the matrix bandwidth minimization. Europ. J. of Operational Research 186, 513–528 (2008)
26. Nöllenburg, M., Wolff, A.: Drawing and labeling high-quality metro maps by mixed-integer programming. IEEE TVCG 17(5), 626–641 (2011)
27. Papamanthou, C., Tollis, I.G.: Applications of parameterized st-orientations. J. Graph Algorithms Appl. 14(2), 337–365 (2010)
28. Sadasivam, S., Zhang, H.: NP-completeness of st-orientations for plane graphs. Theoret. Comput. Sci. 411(7-9), 995–1003 (Feb 2010)
29. Tamassia, R., Tollis, I.: A unified approach to visibility representations of planar graphs. Discrete Comput. Geom. 1(1), 321–341 (1986)
30. Wismath, S.K.: Characterizing bar line-of-sight graphs. In: Proc. first Ann. Symp. Comput. Geom. pp. 147–152. SCG '85, ACM, New York, NY, USA (1985)