An SDP Approach to Multi-level Crossing Minimization

Markus Chimani[*] Philipp Hungerländer[†] Michael Jünger[‡] Petra Mutzel[§]

Abstract

We present an approach based on semidefinite programs (SDP) to tackle the multi-level crossing minimization problem. Thereby, we are given a layered graph (i.e., the graph's vertices are assigned to multiple parallel levels) and ask for an ordering of the nodes on their levels such that, when drawing the graph with straight lines, the resulting number of crossings is minimized. Solving this step is crucial in the probably most widely used graph drawing scheme, the so-called Sugiyama framework.

The problem has received a lot of attention both in the field of heuristics and exact methods. For a long time, integer linear programming (ILP) approaches were the only exact algorithms applicable at least to small graphs. Recently, SDP formulations for the special case of two levels were proposed and dominated the ILP for dense instances.

In this paper, we present a new SDP formulation for the general multi-level version that, for two-levels, is even stronger than the aforementioned specialized SDP. As a side-product, we also obtain an SDP-based heuristic which in practice always gives (near-)optimal solutions.

We conduct a large set of experiments, both on randomized and on real-world instances, and compare our approach to a state-of-the-art ILP-based branch-and-cut implementation. The SDP clearly dominates for denser graphs, while the ILP approach is usually faster for sparse instances. However, even for such sparse graphs, the SDP solves more instances to optimality than the ILP. In fact, there is no single instance the ILP solved, which the SDP did not. Overall, our experiments reveal that for sparse graphs, one should usually try to find an optimal solution with the ILP first. If this approach does not solve the instance to optimality within reasonable time, the SDP still has a good chance to do so.

Being able to solve larger real-world instances than reported before, we are also able to evaluate heuristics for this problem. In this paper we do so for the traditional barycenter-heuristic (showing that it leaves a large gap to the true optimum) and the state-of-the-art upward-planarization method (showing that it is usually close to the optimum).

1 Introduction

Multi-level crossing minimization (MLCM) is an important task in automatic graph drawing. Hierarchical graphs (e.g., directed acyclic graphs) are mostly drawn with the framework suggested by Sugiyama et al. [30]. Here, the vertices are assigned to levels (corresponding to horizontal layers), essentially fixing the y-coordinates of the vertices. Then, the graph is transformed into a *proper* multi-level graph in which edges are subdivided such that each edge connects two vertices on adjacent layers. The aim of the multi-level crossing minimization step is to reorder the vertices within the levels so that the number of crossings is minimized when the edges are drawn as straight lines. Finally, the position of the vertices are assigned while keeping the leveling and the ordering of the vertices. An alternative paradigm to Sugiyama's approach is based on upward planarization [3]. Also in this setting, finding optimal solutions of MLCM is of interest (see Section 4).

In practice, MLCM is often reduced to a sequence of 2-level crossing minimization problems in which one level is fixed. Many heuristics [30, 24] as well as FPT algorithms [7] have been suggested for this restricted problem, but so far variants of the simple barycenter and median heuristics belong to the best in practice [22, 24]. General MLCM is NP-hard, even in this restriced variant [8]. Jünger and Mutzel [22] have shown that this restricted problem can be reduced to a linear ordering problem that can be solved using an integer linear programming (ILP) approach. Combined with a branch-and-bound method, they solved 2-level MLCM instances with up to 15 vertices on the smaller level to optimality. An alternative exact approach for solving 2-level MLCM, based on *semidefinite programming (SDP)*, has been suggested by Buchheim et al. [2]. They model the problem as a quadratic linear ordering problem which in turn can be reduced to a maximum cut problem. They suggest new ILP- and SDP-based algorithms exploiting their observation. Their experiments show that their SDP-based branch-and-bound algorithm outperforms various versions of ILP-based branch-and-cut algorithms, and is able to solve 2-level instances with up to 18 vertices per level to optimality

[*]Institute of Computer Science, Friedrich-Schiller-University of Jena, Germany, markus.chimani@uni-jena.de. Markus Chimani was funded by a Carl-Zeiss-Foundation juniorprofessorship.

[†]Institute of Mathematics, Alpen-Adria Universität Klagenfurt, Austria, philipp.hungerlaender@uni-klu.ac.at.

[‡]Department of Computer Science, University of Cologne, Germany, mjuenger@informatik.uni-koeln.de.

[§]Department of Computer Science, Technische Universität Dortmund, Germany, petra.mutzel@cs.tu-dortmund.de.

within reasonable computing time.

The first ILP formulation for general MLCM has been suggested in [21]. At that time, generic ILP solvers have not been able to solve practically relevant instances. Healy and Kuusik extended the ILP formulation by constraints arising from the so-called vertex-exchange graph [15]. For the first time they have been able to solve some practically relevant instances of MLCM to optimality [16].

Contribution. We suggest a new SDP-based approach for MLCM and prove its polyhedral advantages over the known ILP models. For two levels, our semidefinite relaxation is stronger than the one considered in [2]. We use a primal-dual interior point method for solving the semidefinite relaxation leading to lower bounds to the optimal value of MLCM. Our extensive computational experiments on a large benchmark set of graphs show that this new approach in combination with an SDP-based heuristic very often provides optimal solutions. We are able to compute optimal solutions for graph instances from the literature that have not been solved to optimality before.

We also compared our approach to a standard ILP formulation, solved via branch-and-cut within a generic ILP solver. Surprisingly, while the SDP approach dominates for denser graphs, the ILP turns out to be very fast for sparse, practical instances. It solves almost all instances of the Rome benchmark set, a standard graph drawing library. Yet, our experiments show that the SDP approach solves more instances to optimality than the ILP approach, although the former is not combined with a branch-and-bound scheme. This also suggests a new heuristic for MLCM based on SDP which clearly outperforms the classical heuristics.

Having obtained optimal solutions for graphs of interesting size, we can for the first time evaluate heuristic solutions. We show that the upward planarization approach is very close to the optimum concerning the given leveling, while this is not true for the standard barycenter heuristic. For our studies, we collected a large benchmark set of leveled graphs, available at `http://www.ae.uni-jena.de/Research_Pubs/MLCM.html`.

In the following, we will always consider a *proper level graph* $G = (V, E)$, with vertex set $V = \dot{\bigcup}_{r=1}^{p} V_r$ and edge set $E = \dot{\bigcup}_{r=1}^{p-1} E_r$ with $E_r \subseteq V_r \times V_{r+1}$. Let us further denote by $N(v)$ the set of vertices on level $r + 1$ that are the destinations of edges adjacent to vertex v on level r. We ask for an ordering of the vertices (when drawn on their respective level) such that the number of crossings between straight-line edges is minimized. For notational simplicity, we will assume that the vertices are uniquely numbered (increasing by level), starting from 1.

2 ILP Formulations

MLCM has a natural formulation as a quadratic linear program in 0-1 variables [21]. We introduce binary variables

$$x_{ij}^r \in \{0,1\}, \qquad 1 \le r \le p,\ i,j \in V_r,\ i < j \tag{2.1}$$

that shall be 1 if vertex i comes before j on layer r, and 0 otherwise (i.e., j comes before i). We may use the substitution $x_{ji}^r := 1 - x_{ij}^r$ for $1 \le i < j \le |V_r|$.Then the following *3-cycle constraints* describe linear orderings on the layers of a given proper level graph:

$$0 \le x_{ij}^r + x_{jk}^r - x_{ik}^r \le 1, \qquad 1 \le r \le p,\ i,j,k \in V_r,\ i < j < k. \tag{2.2}$$

They rule out the existence of directed 3-cycles and are sufficient to insure that there are no directed cycles. Hence the feasible binary solutions w.r.t. (2.2) describe complete acyclic digraphs on the layers. Minimizing

$$\sum_{1 \le r < p} \sum_{1 \le i < j \le |V_r|} \sum_{k \in N(i), l \in N(j)} (x_{ij}^r x_{lk}^{r+1} + x_{ji}^r x_{kl}^{r+1})$$

over the constraints (2.1) and (2.2) therefore solves MLCM. We can linearize the objective function by introducing binary crossing variables

$$c_{ij,kl}^r \in \{0,1\}, \qquad 1 \le r < p,\ (i,k),(j,l) \in E_r \tag{2.3}$$

that shall be 1 if the edges (i,k) and (j,l) cross and 0 otherwise. To bind the crossing variables with the linear ordering variables, we need

$$-c_{ij,kl}^r \le x_{kl}^{r+1} - x_{ij}^r \le c_{ij,kl}^r, \qquad (i,k),(j,l) \in E_r,\ k < l; \tag{2.4}$$

$$1 - c_{ij,kl}^r \le x_{lk}^{r+1} + x_{ij}^r \le 1 + c_{ij,kl}^r, \qquad (i,k),(j,l) \in E_r,\ k > l. \tag{2.5}$$

Let x be the vector collecting the variables x_{ij}^r and $c \in \mathbb{B}^t$ be the vector of crossing variables, with t the total number of crossing variables for the graph. Then we can formulate MLCM as a binary linear program as

$$z_* = \min \left\{ \sum_{1 \le r < p} \sum_{(i,k)(j,l) \in E_r} c_{ij,kl}^r \ :\ (x,c) \in \mathcal{I}_{CR}(G) \right\}$$

where

$\mathcal{I}_{CR}(G) := \{\ (x,c) \in \{0,1\}$ that satisfy (2.2), (2.4), (2.5)$\}$.

Replacing the integrality conditions with 0-1 bounds gives the linear relaxation denoted by `(LP)` with objective value z_{lp}. An exact algorithm using `(LP)` has been introduced by Jünger and Mutzel [22] and was further extended by [21] and [15]. All these algorithms perform best on sparse instances ($d \leq 0.1$) (see [2]) as for higher densities the gaps between z_* and the LP bounds become too large for efficient pruning in Branch-and-Bound or Branch-and-Cut algorithms. In fact, by setting all x variables to 0.5, we can always—independently on the instance—obtain a feasible fractional solution with $z_{LP} = 0$. To prevent this, we propose to fix a *single* linear ordering variable to 1 or 0, to break the symmetry without loosing all optimal solutions; yet in practice the obtained relaxed solution still gives only a weak bound. We will investigate this in more detail in Section 4. Thus it would be desirable to have some tighter approximation available in these cases.

3 The Semidefinite Program

In this section we concentrate on the lower bound computation for MLCM by analyzing matrix liftings of ordering problems. For this purpose it is convenient to transform the linear ordering variables x_{ij} into variables taking the values -1 and 1:

$$(3.6) \quad y_{ij}^r = 2x_{ij}^r - 1, \qquad 1 \leq r \leq p,\ i,j \in V_r,\ i < j.$$

This leads to inequalities equivalent to (2.2)

$$(3.7) \quad -1 \leq y_{ij}^r + y_{jk}^r - y_{ik}^r \leq 1, \qquad 1 \leq r \leq p,\ i,j,k \in V_r,\ i < j < k.$$

In [17] it is shown that one can easily switch between the $\{0,1\}$ and $\{-1,1\}$ formulations of bivalent problems so that the resulting bounds remain the same and structural properties are preserved.

3.1 Lower bound: semidefinite relaxation The matrix lifting approach takes a vector y collecting the variables y_{ij}^r and considers the matrix $Y = yy^T$. Our object of interest now is

$$\mathcal{P}_{QC} := \text{conv}\ \{\ yy^T\ :\ y \in \{-1,1\},\ y \text{ satisfies (3.7)}\ \}.$$

We relax the nonconvex equation $Y - yy^T = 0$ to the constraint $Y - yy^T \succeq 0$, which is convex due to the Schur-complement lemma. Moreover, the main diagonal entries of Y correspond to y_{ij}^2, and hence $\text{diag}(Y) = e$, the vector of all ones. We therefore conclude that any $Y \in \mathcal{P}_{QC}$ satisfies

$$(3.8) \qquad Y - yy^T \succeq 0, \quad \text{diag}(Y) = e.$$

To simplify our notation, we introduce

$$(3.9) \qquad Z = Z(y, Y) := \begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix},$$

where $\zeta := \dim(Z) = 1 + \sum_{i=1}^p \binom{|V_i|}{2}$ and $Z = (z_{ij})$. In this case $Y - yy^T \succeq 0 \Leftrightarrow Z \succeq 0$. Hence, the following basic set $\mathcal{B}$ contains $\mathcal{P}_{QC}$ (in the submatrix of Z corresponding to Y),

$$(3.10) \qquad \mathcal{B} := \{\ Z :\ \text{diag}(Z) = e, Z \succeq 0\ \}.$$

In order to express constraints on y in terms of Y, they have to be reformulated as quadratic conditions in y where we denote the product $y_{ij}^r y_{kl}^s$ by $y_{ij,kl}^{r,s}$. Using $y \in \{-1,1\}$ in (3.7) gives $|y_{ij}^r + y_{jk}^r - y_{ik}^r| = 1$. A natural way for a quadratic reformulation of this equation is to square both sides, which, since $y_{ij}^2 = 1$, simplifies to

$$(3.11) \quad y_{ij,jk}^{r,r} - y_{ij,ik}^{r,r} - y_{ik,jk}^{r,r} = -1, \qquad 1 \leq r \leq p,\ i,j,k \in V_r,\ i < j < k.$$

In [2] it is shown that these equations (formulated in the $\{0,1\}$ model) describe the smallest linear subspace that contains $\mathcal{P}_{QC}$. We formulate MLCM as a semidefinite optimization problem in bivalent variables, where Z is given by (3.9), and C is a symmetric matrix of order ζ assigned to count the number of crossings for any given feasible ordering y.

THEOREM 3.1. *MLCM is equivalent to the problem* $z_* = \min\ \{\ \langle C, Z\rangle\ :\ Z \in \mathcal{I}_{QC}\ \}$, *where*

$$\mathcal{I}_{QC} = \ \{\ Z \textit{ partitioned as in } (3.9) \textit{ and satisfies } (3.11),\ Z \in \mathcal{B},\ y \in \{-1,1\}\ \}.$$

When we drop the integrality condition on y, we get the following basic semidefinite relaxation `(SDP`$_b$`)` for MLCM:

$$(\texttt{SDP}_b) \quad \min\ \{\ \langle C, Z\rangle\ :\ Z \text{ partitioned as in (3.9) and satisfies (3.11)},\ Z \in \mathcal{B}\ \}.$$

There are some obvious ways to tighten the relaxation `(SDP`$_b$`)`. First of all we observe that Y, and therefore Z, is actually a matrix with $\{-1,1\}$ entries in the ordering formulation. Hence it satisfies the triangle inequalities, defining the metric polytope

$$(3.12) \quad \mathcal{M} := \left\{ \begin{array}{l} Z\ :\ \begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{jk} \\ z_{ik} \end{pmatrix} \leq e, \\ \hfill \forall 1 \leq i < j < k \leq \zeta \end{array} \right\}.$$

We note that this polytope is defined through $4\binom{\zeta}{3} = O((\sum_{i=1}^p |V_i|^2)^3)$ facets. They are used as triangle inequalities of the max-cut polytope in [23, 27, 29]. The basic relaxation `(SDP`$_b$`)` can therefore be improved by additionally asking that $Z \in \mathcal{M}$.

Another generic improvement was suggested by Lovász and Schrijver in [23]. Applied to our problem, their approach suggests to multiply the 3-cycle inequalities (3.7) by the nonnegative expressions $(1-y^s_{lm})$ and $(1+y^s_{lm})$. This results in the following inequalities for all $1 \le r, s \le p$, $i, j, k \in V_r$, and $l, m \in V_s$ with $i < j < k$ and $l < m$:

$$\begin{aligned} & -1 - y^s_{lm} \le \\ & \quad y^r_{ij} + y^r_{jk} - y^r_{ik} + y^{r,s}_{ij,lm} + y^{r,s}_{jk,lm} - y^{r,s}_{ik,lm} \\ & \qquad\qquad\qquad\qquad \le 1 + y^s_{lm}, \\ & -1 + y^s_{lm} \le \\ & \quad y^r_{ij} + y^r_{jk} - y^r_{ik} - y^{r,s}_{ij,lm} - y^{r,s}_{jk,lm} + y^{r,s}_{ik,lm} \\ & \qquad\qquad\qquad\qquad \le 1 - y^s_{lm}. \end{aligned} \tag{3.13}$$

There are overall $4\left(\sum_{i=1}^p \binom{|V_i|}{3}\right)\left(\sum_{i=1}^p \binom{|V_i|}{2}\right) = O((\sum_{i=1}^p |V_i|^3)(\sum_{i=1}^p |V_i|^2))$ such inequalities and we define

$$\mathcal{LS} := \{\ Z : Z \text{ satisfies (3.13)}\ \}.$$

In summary, we get the following tractable relaxation of $\mathcal{P}_{QC}$, part of which (without the matrix cuts (3.13)) has been investigated in [2] for bipartite crossing minimization problems, in [1] for single-row layout problems, and, including (3.13), in [20] for general quadratic linear ordering problems.

$$(\texttt{SDP}_i) \quad \min \{\ \langle C, Z\rangle\ :\ Z \text{ partitioned as in (3.9) and satisfies (3.11)},\ Z \in \mathcal{B},\ Z \in \mathcal{M},\ Z \in \mathcal{LS}\ \}$$

We close this section by relating $(\texttt{SDP}_b)$ (and therefore also $(\texttt{SDP}_i)$) to the linear relaxation (LP) of MLCM (proof in the appendix):

THEOREM 3.2. *The basic semidefinite relaxation* $(\texttt{SDP}_b)$ *together with the constraints* $(1 \pm y^r_{ij})(1 \pm y^{r+1}_{kl}) \ge 0$ *(and therefore also* $(\texttt{SDP}_i)$*) is at least as strong as the linear relaxation* (LP).

3.2 Further facets used for separation We omit the proofs of the following theorems; they can be found in the appendix. We know from [21] that $\mathcal{P}_{CR}(G) = \text{conv}(\mathcal{I}_{CR}(G)) \in \mathbb{B}^{(\zeta-1)+t}$ is full dimensional. We first relate $\mathcal{P}_{CR}(G)$ to $\mathcal{P}_{QC}$; then we present various classes of facet-defining inequalities for $\mathcal{P}_{CR}(G)$ and show that $(\texttt{SDP}_i)$ contains them. For the former, we consider the lifting $\mathcal{P}^*_{CR}$ of $\mathcal{P}_{CR}$ by extending the variable vector c to incorporate all possible crossing variables, not only for vertex pairs of adjacent layers with associated costs $\neq 0$, and by adding upper bounds for these variables. Formally, let $I := \{(i,j,k,l) \mid i < j \text{ and } k < l \text{ and } (i < k \text{ or } (i = k \text{ and } j < l))\}$, and consider the constraints

$$c_{ij,kl} \le x_{ij}, \quad c_{ij,kl} \le x_{kl}, \quad c_{ij,kl} \le 1 - x_{ij} - x_{kl}. \tag{3.14}$$

for all $(i,j,k,l) \in I$. Considering the extended variable vector c, we define

$$\mathcal{P}^*_{CR} := \text{conv}\{(x,c) \text{ satisfy (2.2), (3.14)}, (x,c) \in \{0,1\}\}.$$

THEOREM 3.3. *$\mathcal{P}_{CR}(G)$ contains $\mathcal{P}^*_{CR}$, when projected onto the common variables. $\mathcal{P}^*_{CR}(G)$ and $\mathcal{P}_{QC}$ describe the same polytope, modulo transformation* (3.6).

COROLLARY 3.1. *Facet-defining inequalities of $\mathcal{P}_{CR}(G)$ are valid inequalities for $\mathcal{P}_{QC}$.*

PROPOSITION 3.1. ([21]) *Let C be a cycle and W_q a q-claw in G. The following inequalities are valid for $\mathcal{P}_{CR}(G)$:*

$$\sum_{(i,k)(j,l) \in C} c_{ij,kl} \ge |C|/2 - 1, \tag{3.15}$$

$$\sum_{(i,k)(j,l) \in W_q} c_{ij,kl} \ge \begin{cases} \frac{q}{2}\left(\frac{q}{2} - 1\right) & q \text{ even} \\ \left(\frac{q-1}{2}\right)^2 & q \text{ odd} \end{cases} \tag{3.16}$$

Let T be the set consisting of all pairs of edges of a W_3 except those pairs of edges that are either both within the lower part of the 3-claw. The following inequalities are facet-defining for $\mathcal{P}_{CR}(G)$:

$$\sum_{(k,l) \in C, k \neq i, l \neq i+1} c_{ik(i+1)l} + c_{i(i+1)(i-1)(i+2)} \ge 1, \tag{3.17}$$

$$\sum_{(i,j),(k,l) \in T} c_{ikjl} \ge 1. \tag{3.18}$$

For $i, j, k \in V_r$, $i < j < k$, $l, m \in V_{r+1}$, $1 \le r < p$ the following inequalities constructed from dome paths are facet-defining for $\mathcal{P}_{CR}(G)$:

$$\begin{aligned} x^r_{ij} - 2x^r_{ik} + x^r_{jk} - c^r_{ij,lm} - c^r_{jk,lm} &\le 0, \\ -x^r_{ij} + 2x^r_{ik} - x^r_{jk} - c^r_{ij,lm} - c^r_{jk,lm} &\le 0, \\ 2x^r_{ij} - x^r_{ik} + x^r_{jk} - c^r_{ik,lm} - c^r_{kj,lm} &\le 1, \\ -2x^r_{ij} + x^r_{ik} - x^r_{jk} - c^r_{ik,lm} - c^r_{kj,lm} &\le -1, \\ x^r_{ij} - x^r_{ik} + 2x^r_{jk} - c^r_{ji,lm} - c^r_{ik,lm} &\le 1, \\ -x^r_{ij} + x^r_{ik} - 2x^r_{jk} - c^r_{ji,lm} - c^r_{ik,lm} &\le -1. \end{aligned} \tag{3.19}$$

THEOREM 3.4. $(\texttt{SDP}_i)$ *satisfies* (3.15)–(3.19) *except* (3.16) *for $q \ge 5$ and odd.*

COROLLARY 3.2. $(\texttt{SDP}_i)$ *is as least as tight as* (LP) *together with* (3.16)–(3.19) *except* (3.16) *for $q \ge 5$ and odd.*

In summary, any of the inequality types considered in $(\texttt{SDP}_i)$ is required to ensure facet-defining inequalities for $\mathcal{P}_{CR}(G)$.[1] On the other hand, if we want an SDP relaxation

[1]As $(3.13)_{r\neq s}$ is not relevant for identifying known facets of $\mathcal{P}_{QC}$ and because of efficiency considerations, we do not use $(3.13)_{r\neq s}$ in our computational experiences.

that ensures more known facets of $\mathcal{P}_{CR}(G)$ than $(\texttt{SDP}_i)$, we have to consider additionally clique inequalities of size $q \geq 5$ odd in the relaxation. As separating them is far too expensive, this supports our model choice.

3.3 Solving $(\texttt{SDP}_i)$ Looking at the constraint classes and their sizes in the relaxation $(\texttt{SDP}_i)$, it should be clear that maintaining explicitly $O(\sum_{i=1}^{p} |V_i|^3)$ or more constraints is not an attractive option. We therefore consider an approach suggested in [10] and adapt it to our problem. Firstly, we only aim at maintaining the constraint $Z \in \mathcal{B}$ (i.e., $Z \succeq 0 \wedge \text{diag}(Z) = e$) explicitly, which can be achieved with standard interior point methods, see for instance [18].

All other constraints are dealt through Lagrangian duality. For notational convenience, let us formally denote the 3-cycle equations (3.11) by $e - \mathcal{A}(Z) = 0$. Similarly we write $\mathcal{M} \cap \mathcal{LS}$ as $e - \mathcal{D}(Z) \geq 0$. We consider the partial Lagrangian dual defined through the Lagrangian

$$\mathcal{L}(Z, \lambda, \mu) := \langle C, Z \rangle + \lambda^\top (e - \mathcal{A}(Z)) + \mu^\top (e - \mathcal{D}(Z)).$$

The dual function is thus given by

$$\begin{aligned} f(\lambda, \mu) &:= \min_{Z \in \mathcal{B}} \mathcal{L}(Z, \lambda, \mu) = \\ &= e^\top \lambda + e^\top \mu + \min_{Z \in \mathcal{B}} \langle C - \mathcal{A}^\top(\lambda) - \mathcal{D}^\top(\mu), Z \rangle. \end{aligned}$$

It is not hard to verify that $(\texttt{SDP}_i)$ has strictly feasible points, so strong duality holds and we can solve the relaxation through

$$\max_{\mu \geq 0, \lambda} f(\lambda, \mu). \tag{3.20}$$

The function f is well-known to be convex but non-smooth. For a given feasible point (λ, μ) the evaluation of $f(\lambda, \mu)$ amounts to solving a problem over $\mathcal{B}$.

In our experiments, we use a primal-dual interior-point method, that also provides the primal optimum $Z_{\lambda,\mu}$ yielding a subgradient of f. Having a subroutine available that evaluates f and an element of the subdifferential of f, it is straightforward to get an approximate minimizer of f using subgradient optimization techniques, such as the bundle method [10]. Since these methods have rather weak local convergence behavior, we limit the number of function evaluations to control the overall computational effort. In fact these evaluations constitute the computational bottleneck as they are responsible for more than 95% of the required running time.

3.4 Upper bound computation Standard heuristics and also some metaheuristics perform quite poorly for MLCM-instances of sizes of our interest [22, 24]. Therefore we apply a heuristic that exploits information obtained during the bundle method in the following way. Initially, we consider a vector x', that encodes a feasible, random ordering on all layers. The algorithm stops after 1000 executions[2] of step 2; x' is then the heuristic solution. If the duality gap is not closed after the heuristic, we continue with further bundle iterations and then retry the heuristic (retaining the last vector x').

1. Let X'' be the current primal fractional solution of $(\texttt{SDP}_i)$ obtained by the bundle method. Compute the convex combination $R := \lambda(x'x'^\top) + (1-\lambda)X''$, using some random $\lambda \in [0.3, 0.7]$. Compute the Cholesky decomposition $DD^\top$ of R.
2. Apply Goemans-Williamson hyperplane rounding [13] to D and obtain a $-1/{+1}$ vector $\overline{x}$ (cf. [26]).
3. Compute the induced crossing number $z(\overline{x})$. If $z(\overline{x}) \geq z(x')$: goto step 2.
4. If $\overline{x}$ satisfies all 3-cycle inequalities: set $x' := \overline{x}$ and goto 2. Else: modify $\overline{x}$ by changing the signs of one of three variables in all violated inequalities and goto step 3.

4 Computational Experience

Due to licensing issues and overall CPU time we conducted our experiments on two different machines. All SDP computations where conducted on an Intel Xeon E5160 (Dual-Core) with 24 GB RAM, running Debian 5.0. The algorithm itself runs on top of MatLab 7.7.

For comparison, we also considered a newly written ILP implementation (along the lines of [21]) using Branch-and-Cut. Thereby the triangle inequalities are separated on the fly, instead of adding all of them initially. We do not specifically separate further inequalities as the ones described in Proposition 3.1: It was observed in [16] that even though the number of branch-and-bound nodes decreases, the additional effort needed to identify violated constraints—even of the simple cycle types (3.15) and (3.17)—leads to overall increased running times. We also evaluated the ILP variant without separation; as this approach resulted clearly worse running times, we only report on the code with separation. These experiments were conducted on an Intel Xeon E5520 (Dual-CPU, Quad-Core) with 72 GB RAM, running Debian 6.0. The C++ code uses CPLEX 12.1 (with default settings) as a B&C framework.

Both codes were run in 32bit mode, restricting them to effectively 2GB RAM. Note that both the second machine as well as the implementation language C++ and the highly tuned commercial (I)LP solver can be expected to be faster than their SDP counterparts. Herein we are not so much interested in the exact running times, but in the order of magnitude. Not only can we assume that our setting can achieve such a comparison, we will in fact see that the SDP

[2]Before its 501st execution, we perform step 1 again. As this is quite expensive, we refrain from executing it too often.

approach outperforms the ILP approach despite this setting.

We restrict the SDP approach to 1500 function evaluations of $f(\lambda, \mu)$, as the convergence process of the bundle method mostly slows down before that point, independent of problem size ζ. After every fifth function evaluation we search for newly violated constraints at our current primal point. We add all constraints with violation > 0.001 to the bundle and additionally remove constraints with relatively speaking small associated Lagrangian multipliers ($\lambda_i < 0.01 \cdot \lambda_{\text{mean}}$). A further critical operation is the first-time initialization of the dual variables, where we choose the initial λ_i as "$\frac{2 \cdot \text{initial duality gap}}{\|\text{ total constraint violation }\|^2} \cdot$ violation of constraint i".

4.1 Graphs with varying densities We argued that the LP-gap becomes too large for dense instances, in order to allow practically efficient ILP methods to succeed in such cases; this argument is supported by the known results for two-layer crossing minimization [2]. Hence we start out with considering a synthetic benchmark where we have control over this density parameter: We generated a set of instances having $p \in \{2, \ldots, 20\}$ layers and $n \in \{8, \ldots, 25\}$ vertices on each layer. For each combination of p and n, we consider random instances with densities $d \in \{0.1, 0.2, \ldots, 0.9\}$, i.e., with $\lfloor dn^2 \rfloor$ edges per layer, where all possible edges have equal probability of being selected. For each triple (p, n, d), we report the average over 10 generated instances.

Table 1 summarizes our results for some representative values of p, n, d. We restricted the ILP approach to 1 hour of computation per instance: We observe that the solved instances always require less than 1 minute (except for four instances with 24, 6, 3 and 1.5 minutes, respectively); for the unsolved instances the gaps are still very large after 1 hour and progress stagnates.

Not surprisingly, we can observe that the graph density is not so important for the SDP; while very sparse and dense graphs allow the SDP to find solutions quickly, even most of the instances with a more complicated cost structure ($d \approx 0.5$) can be solved within seconds. On the other hand, the ILP approach is applicable only to very sparse graphs: it can solve all instances with $d = 0.1$. Thereby it is by an order of magnitude faster than the SDP. Yet, it solves not a single instance with $d \geq 0.2$ within 1 hour.

Regarding the two-level case, we note that the similar approach [2] using a weaker SDP relaxation was not able to solve all $n = 18$ instances[3]. The ILP approach suggested in [15] considered 10 random instances with $p = 8, n = 12, d = 0.109$. We also tested the only still available instance of these, observing equivalent behavior to our random instances.

[3]The effects of including the matrix cuts are also already demonstrated on exemplary instances in [20, Subsection 4.5].

4.2 Real-world graphs Motivated by the above results we now turn our attention to commonly used benchmark sets in the area of graph drawing, where the considered graphs are relatively sparse, and investigate our algorithm more deeply. Both instance sets described below are said to be at least similar to real-world instances; to our knowledge this is the first time that these instances are tackled in the context of any *exact* multi-level crossing minimization.

Rome graphs. The Rome graphs, originally proposed in [6], were obtained from a basic set of 112 real-world graphs. The collection contains 11,528 instances with 10–100 vertices and 9–158 edges and, although originally undirected, can be unambiguously interpreted as directed acyclic graphs, as proposed in [9].

North DAGs. The North DAGs have been introduced in an experimental comparison of algorithms for drawing DAGs [5]. The benchmark set contains 1,158 DAGs collected by Stephen North that were slightly modified by Di Battista et al. The graphs are grouped into 9 sets, where set i contains graphs with $10i$ to $10i+9$ arcs for $i = 1, \ldots, 9$.

Both instance sets contain regular graphs, which are not proper level graphs. As they have been regularly used as benchmarks for Sugiyama style drawings, we consider two different leveling approaches:

GKNV. As indicated in the introduction, the first step of the traditional Sugiyama approach is to level the given graph. There are multiple strategies to decide on a leveling; in these experiments, we consider the optimal LP-based algorithm [12]. In this context, we can also evaluate traditional multi-level crossing minimization strategies: In the tables below, we will also give the number of crossings B obtained by the level-wise barycenter heuristic (sweeping over all level until the solution does not further improve).

UPL. Recent algorithms have combined the first and the second step of Sugiyama's framework to obtain an *upward planarization* algorithm [3]. Thereby, a *planarization* P with few crossings is computed without the need for levels. Afterwards, P is fitted into the smallest leveling allowing the specified crossing configuration [4], in order to be applicable for Sugiyama's third step. We will also consider the layering obtained by this approach, as it allows a much smaller number of crossings in practice. This also allows us to deduce if (thinking inversely) the UPL approach gives a (near-)optimal number of crossings with respect to the finally computed layering.

Results. Recall that the matrix dimension ζ does not only depend on the original number of vertices (or edges), but on the derived proper level graph, i.e., also on the number of artificial vertices and the vertex distribution over a number of layers. Hence the algorithm will be mostly dependent on ζ rather than the original size. Figure 1 shows the dependency between these different metrics. We calculated

		SDP						ILP
	$d =$	0.1	0.2	0.3	0.5	0.7	0.9	0.1
p	n, ζ	✓, time	✓, time	✓, time	✓, time	✓, time	✓, time	✓, time
2	**20**, 381	**10**, 45.1	**10**, 223.1	**10**, 578.3	**10**, 478.6	**10**, 513.5	**10**, 319.6	**10**, 3.40
	21, 421	**10**, 58.2	**10**, 246.5	**10**, 912.9	**8**, 1601.5	**10**, 1292.0	**10**, 458.4	**10**, 15.67
	22, 463	**10**, 117.8	**9**, 505.9	**9**, 788.5	**7**, 3621.3	**9**, 2866.2	**10**, 923.4	**10**, 28.15
3	**16**, 361	**10**, 56.0	**10**, 238.5	**10**, 601.9	**10**, 559.1	**9**, 858.2	**10**, 209.6	**10**, 3.99
	17, 409	**10**, 130.4	**10**, 324.6	**8**, 738.8	**7**, 863.9	**9**, 1300.2	**10**, 557.2	**10**, 11.22
	18, 460	**10**, 112.1	**10**, 804.5	**9**, 1278.9	**7**, 1684.0	**8**, 2187.5	**10**, 534.2	**10**, 23.70
6	**12**, 397	**10**, 52.4	**10**, 279.3	**8**, 158.8	**9**, 641.0	**10**, 1262.7	**10**, 281.1	**10**, 0.70
	13, 469	**10**, 149.2	**9**, 772.7	**10**, 978.1	**7**, 2898.5	**10**, 1079.8	**10**, 688.4	**10**, 4.90
	14, 547	**10**, 347.5	**8**, 923.9	**6**, 1898.0	**2**, 2037.4	**5**, 3643.6	**10**, 1602.3	**10**, 189.04
11	**10**, 495	**10**, 97.1	**10**, 383.8	**10**, 987.8	**9**, 2539.3	**10**, 1289.7	**10**, 722.4	**10**, 0.45
	11, 605	**10**, 248.1	**10**, 1189.3	**10**, 1861.7	**9**, 4443.3	**6**, 5518.5	**10**, 1171.3	**10**, 2.07
	12, 726	**10**, 615.1	**9**, 1843.7	**6**, 7864.2	**8**, 8490.7	**4**, 8940.0	**10**, 2959.9	**10**, 50.52
20	**8**, 561	**10**, 13.5	**10**, 356.7	**10**, 701.6	**10**, 1145.9	**10**, 1220.6	**10**, 591.5	**10**, 0.01
	9, 721	**10**, 149.4	**10**, 1395.8	**10**, 2284.4	**9**, 3023.7	**10**, 3523.8	**10**, 2605.1	**10**, 0.21
	10, 901	**10**, 1000.9	**10**, 3213.6	**10**, 5979.1	**4**, 11407.5	**10**, 13976.2	**10**, 6430.9	**10**, 2.46

Table 1: SDP and ILP approaches on random graphs with representatively chosen values for d, n and p. "✓" denotes the number of instances solved to optimality (out of 10) ,"time" gives the average time (in seconds) over the solved instances. For the ILP, no instance with $d \geq 0.2$ could be solved.

all graphs with $\zeta < 900$ and $\zeta < 1500$ for the Rome and North instances, respectively, and summarize the results in Table 2. Our benchmark instances, except for very small graphs, are all sparse: The average density of the considered instances with $\zeta > 300$ is 0.10, 0.11, 0.12, and 0.12 for the combinations, Rome-GKNV, Rome-UPL, North-GKNV, and North-UPL, respectively. For the ILP approach, we applied a time limit of 4 hours for each instance with $\zeta < 900$, and 16 hours for $\zeta < 1500$.[4] These ILP time limits were chosen such that the SDP approach always finished its at most 1500 function evaluations within that time, i.e., the ILP approach has at least as much CPU time as the SDP approach.

Table 2 summarizes our experiments. The first and most surprising result is that both approaches are in fact very successful on these real-world instances, as only few instances at all remain unsolved by either of these approaches. In accordance to our finding with the random graphs, we observe that the ILP is usually faster. Yet we also observe that the SDP is stronger with respect to overall solvability: It solves all instances except for 2 North-GKNV instances; the ILP approach fails for 21 graphs, including the aforementioned 2. When both algorithms fail, the SDP approach obtained tighter pairs of upper/lower bounds: 498/518 and 853/854 in contrast to the ILP's 418/499 and 336/854. We conclude that for sparse graphs one should usually try the ILP first; when it fails to prove an optimal solution within reasonable time, the SDP approach has a good chance to succeed for such hard instances.

Analyzing the distinct benchmark sets, we observe that the traditional leveling and crossing minimization heuristics leave plenty of room for improvement when considering the minimum number of crossings. In contrast to this, the graphs leveled by the UPL approach only allow much smaller improvements. In fact, it shows that the upward planarization approach [3] gives near-optimal solutions for its respective leveling. We also observe that the fact that UPL produces more but smaller levels and requires less crossings is beneficial for both exact approaches: they solve all UPL instances, while the GKNV instances are harder.

4.3 Polytopes and further instances from literature We close our experimental study with looking at further instances of interest. Often, one considers the graphs modeling the incidence relation between faces (corner, edge, 2D-face,...) of an (LP-)polytope, and hence we are interested in drawing them within a Sugiyama framework. These graphs are naturally very dense. Table 3 shows that we can solve such instances as long as the dimension of our matrix is within reasonable bounds. We observe that the SDP approach is clearly beneficial over the ILP. Even for polytopes that cannot be solved to optimality by either approach, the bounds obtained by the SDP are clearly stronger.

We also considered the instances from the Graphviz gallery [14] as it was recently suggested in [11]. We only report on the non-trivial instances. We observe that our ILP implementation gives comparable running times to those in [11]; thereby our approach is much simpler.

[4]These large computation times did not leave time for the Rome instances with $\zeta \in [900 - 1500)$, so we restricted ourselves to the more diverse North graphs.

			optimal, imp					optimal, ni			imp/ni	ILP	
		$\zeta <$	#	cr (std)	diff (max)	t_{lb}	t_{ub}	#	t_{lb}	t_{ub}	#/#	¬opt	time
GKNV	Rome	300	**2015**	11.14 (8.24)	3.7 (18)	20.55	0.25	**1538**	2.38	0.02	0/0	0	0.26
GKNV	Rome	600	**2572**	26.01 (14.55)	11.2 (52)	0:04:13	9.34	**11**	0:02:03	0.19	0/0	0	7.46
GKNV	Rome	900	**1325**	42.01 (23.02)	24.87 (75)	0:31:55	101.14	**0**	–	–	0/0	**11**	3:57
GKNV	North	300	**90**	16.78 (33.05)	2.04 (10)	16.77	0.09	**316**	3.77	0.03	0/0	**1**	8.20
GKNV	North	600	**80**	25.26 (51.86)	3.81 (20)	0:04:05	5.88	**35**	0:03:04	0.32	0/**1**	**6**	9.82
GKNV	North	900	**36**	29.25 (32.99)	8.03 (29)	0:17:11	22.56	**13**	0:17:33	2.36	0/0	**1**	1:14
GKNV	North	1200	**29**	47.48 (54.01)	8.24 (32)	3:28:34	3:39	**7**	0:06:24	4.49	0/**1**	**2**	1:57
GKNV	North	1500	**11**	64.18 (63.22)	8.45 (17)	6:35:53	6:35	**2**	0:05:14	6.98	0/0	0	8:58
UPL	Rome	300	**136**	1.38 (1.62)	1.18 (4)	5.84	0.10	**442**	3.97	0.05	0/0	0	0.04
UPL	Rome	600	**617**	3.12 (2.11)	1.52 (8)	0:02:44	2.71	**711**	0:02:36	0.92	0/0	0	0.25
UPL	Rome	900	**731**	5.23 (2.95)	2.12 (9)	0:25:13	21.13	**338**	0:18:29	3.35	0/0	0	0.93
UPL	North	300	**30**	4.73 (4.27)	1.57 (5)	9.85	0.09	**126**	3.02	0.03	0/0	0	0.04
UPL	North	600	**45**	5.98 (4.68)	1.91 (5)	0:03:14	1.10	**43**	0:01:31	0.3	0/0	0	0.61
UPL	North	900	**14**	9.50 (6.73)	2.79 (6)	0:18:24	25.79	**20**	0:13:24	2.65	0/0	0	18.86
UPL	North	1200	**11**	11.55 (17.30)	2.27 (6)	1:31:38	38.17	**9**	2:04:10	5.53	0/0	0	1:03
UPL	North	1500	**11**	35.00 (27.17)	4.64 (9)	4:40:28	0:05:04	**5**	2:17:12	7.74	0/0	0	10:19

Table 2: The results for the SDP approach on real-world benchmark instances with $cr > 0$. The results are split into four categories: whether or not SDP found a proven optimal solution ("optimal"), and whether this solution was better than the one from the respective heuristic ("imp" vs. "ni"=no improvement) (see benchmark description). The instances are grouped by matrix dimension ζ in intervals of 300, where ζ is less than the given number. "#" denotes the number of instances, "cr (std)" reports mean and standard deviation of the optimal crossing numbers, "diff (max)" gives the average and maximal difference between the optimal and the heuristic solution. t_{lb} and t_{ub} give the average time (in seconds) to compute the lower bound (via the relaxation $(\mathtt{SDP}_i)$) and the upper bound (via the rounding heuristic described in Subsection 3.4), respectively. We also give the number of instances not solved to optimality by the ILP approach (¬opt) as well as the average solution time over the other instances.

This observation validates the finding in [15] which already suggested that additional separation routines need not pay off in practice. Finally we report on the traditional real-world instances MS88 [25] and SM96 [28]. For the latter, the prior publications only considered a subgraph consisting of three layers, due to the graph's complexity. For the first time, we also report optimal results for the full graph (depicted in Fig. 2). Again we can observe that the SDP approach is beneficial when considering the more complex instances.

5 Conclusions

We presented a systematic investigation and comparison of different exact approaches for MLCM. We demonstrated that the semidefinite relaxation provides essentially tighter lower bounds than the linear programming relaxation. Although computing the former relaxation is more time consuming, our experiments demonstrate that it pays off in practice. As the SDP approach is relatively independent on the graph's density, it is not surprising that the SDP approach clearly dominates the ILP approach on dense graphs. Yet, we also showed that the SDP approach is beneficial on sparse, real-world benchmark sets.

In a very recent, yet unpublished conference paper, Gange et al. [11] suggested a SAT-based approach and compared it to a reimplementation of [15]. They concluded that the ILP dominates the SAT-based alternative. Their experimental performance of the ILP, performed on a comparable PC (Intel Xeon X5472), shows analogous results to our ILP implementation: this can be, e.g., seen when comparing the running times for the Graphviz instances, as proposed in their paper. The random graphs considered in their paper have 7-10 vertices per level and 3-10 levels.

Although not necessary in our experiments, we want to note that the SDP approach could be combined with a branch-and-bound scheme to guarantee an optimal solution upon termination, even when the SDP lower bound does not induce the optimum. Furthermore, $(\mathtt{SDP}_i)$ could be further tightened by multiplying multiply pairs of different 3-cycle inequalities or by pentagonal inequalities, cf. [19].

In the course of developing the SDP algorithm, we also obtained a seemingly strong SDP-based upper bound heuristic. This heuristic never resulted in worse solutions than traditionally used heuristics, often improving their solutions to optimality. Even on the layering obtained to suit the result of the layer-free upward-planarization heuristic, it was regularly able to find even better solutions. This allowed us to prove many optimal solutions in conjunction with the tight SDP lower bound. We think that this new heuristic itself, being very fast to compute, is already a promising new tool to obtain good solutions even for instances that are too large for the full SDP approach. Although beyond the scope of this

Type	Instance	p	d	ζ	SDP z_*	SDP t_{lb}	SDP t_{ub}	ILP time	prev ILP time
Polytopes	Tetrahedron	3	0.5	28	22	0.09	0.03	0.08	—
	Octahedron	3	0.29	110	80	10.61	0.05	2.62	—
	Cube3	3	0.29	110	80	10.87	0.06	3.14	TO (1h) [21]
	Dodecahedron	3	0.13	692	393/394	4:40:07	2.03	18h: 132/427	—
	Icosahedron	3	0.13	692	393/394	4:37:23	1.98	16h: 174/401	—
	Cube4	4	0.14	921	1192/1195	7:10:19	7.10	51h: 197/1334	—
	Soccer ball	3	0.04	6272	1627/2353	91:23:06	0:11:10	52h: 118/2630	—
Graphviz	switch	6	0.2	169	20	1.87	0.05	0.66	0.75 [11]
	unix	11	0.19	176	0	0.24	0.01	0.01	—
	world	9	0.09	815	46	1:11:42	0:02:07	0:02:45	TO (1 min) [11]
	profile	9	0.08	846	37	0:51:32	0:02:02	2.84	6.81 [11]
Other	MS88	3	0.24	217	91	2.62	0.17	5.02	0:04:29 [21]
	SM96-3L	3	0.07	615	13	0:01:19	7.14	0.18	0:05:32 [21]
	SM96-full	7	0.10	915	162	0:53:16	12.75	3:05:05	—

Table 3: Results for Polytopes and further known instances. *Cube3(4)* denotes a 3(4)-dimensional cube. z_* gives the optimal objective value (or final lower and upper bound), and t_{lb}, t_{ub} the time for the lower and upper bound, respectively. *ILP-time* gives the time of the ILP approach; when the process terminated due to insufficient memory (2GB restriction due to 32bit), we give the respective time up to this point and the final lower and upper bound. Due to its complexity, we only computed 50 function evaluations of $f(\lambda, \mu)$ for soccer ball. The last column gives the reported running time in the cited paper to obtain the optimal solution (or TO=timeout). All times are given in seconds or as h:min:sec.

study, it would be interesting to further compare it to other, more involved (meta-)heuristics [24].

References

[1] M. F. Anjos and A. Vannelli. Computing Globally Optimal Solutions for Single-Row Layout Problems Using Semidefinite Programming and Cutting Planes. *INFORMS Journal On Computing*, 20(4):611–617, 2008.

[2] C. Buchheim, A. Wiegele, and L. Zheng. Exact Algorithms for the Quadratic Linear Ordering Problem. *INFORMS Journal On Computing*, pages 168–177, 2009.

[3] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Layer-free upward crossing minimization. *ACM Journal of Experimental Algorithmics*, 15, 2010.

[4] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Upward planarization layout. In *Proceedings of the Symposium on Graph Drawing [GD'09]*, volume 5849 of *LNCS*, pages 94–106. Springer, 2010.

[5] G. Di Battista, A. Garg, G. Liotta, A. Parise, R. Tamassia, E. Tassinari, F. Vargiu, and L. Vismara. Drawing directed acyclic graphs: An experimental study. *International Journal of Computational Geometry and Applications*, 10(6):623–648, 2000.

[6] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry: Theory and Applications*, 7(5-6):303–325, 1997.

[7] V. Dujmović, H. Fernau, and M. Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *J. of Discrete Algorithms*, 6(2):313–323, 2008.

[8] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.

[9] M. Eiglsperger, M. Kaufmann, and F. Eppinger. An approach for mixed upward planarization. *Journal of Graph Algorithms and Applications*, 7(2):203–220, 2003.

[10] I. Fischer, G. Gruber, F. Rendl, and R. Sotirov. Computational experience with a bundle method for semidefinite cutting plane relaxations of max-cut and equipartition. *Mathematical Programming*, 105:451–469, 2006.

[11] G. Gange, P. J. Stuckey, and K. Marriott. Optimal k-level planarization and crossing minimization. In *Proceedings of the Symposium on Graph Drawing [GD'10]*, LNCS. Springer, 2010. to appear.

[12] E. R. Gansner, E. Koutsofios, S. C. North, and K. P. Vo. A technique for drawing directed graphs. *IEEE Trans. Softw. Eng.*, 19(3):214–230, 1993.

[13] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

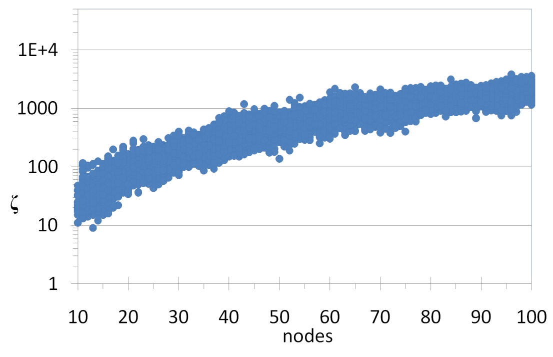
[14] Graphviz gallery, `http://www.graphviz.org/Gallery.php`, Oktober 2010.

[15] P. Healy and A. Kuusik. The vertex-exchange graph: A new concept for multilevel crossing minimisation. In *Proceedings of the Symposium on Graph Drawing [GD'99]*, volume 1731 of *LNCS*, pages 205–216. Springer, 1999.

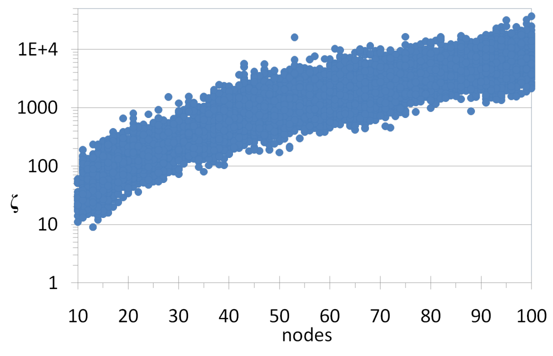
[16] P. Healy and A. Kuusik. The vertex-exchange graph and its use in multi-level graph layout. Technical Report UL-CSIS-99-1, Department of Computer Science and Information Systems, University of Limerick, Ireland, 1999.

[17] C. Helmberg. Fixing variables in semidefinite relaxations. *SIAM Journal on Matrix Analysis and Applications*, 21(3):952–969, 2000.

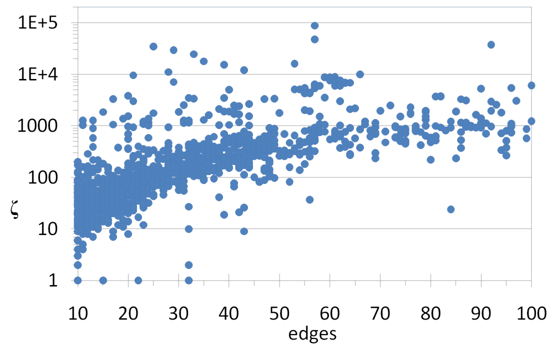
[18] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM*



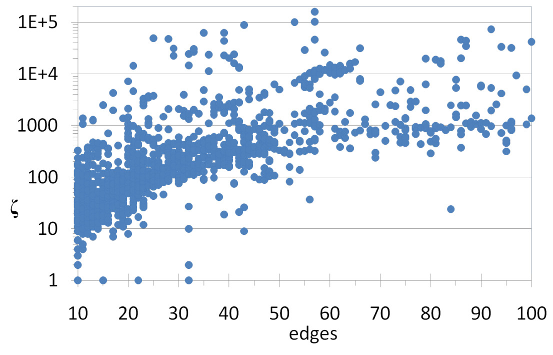
(a) Rome graphs, GKNV



(b) Rome graphs, UPL



(c) North DAGs, GKNV



(d) North DAGs, UPL

Figure 1: Correlation between original graph size and ζ.

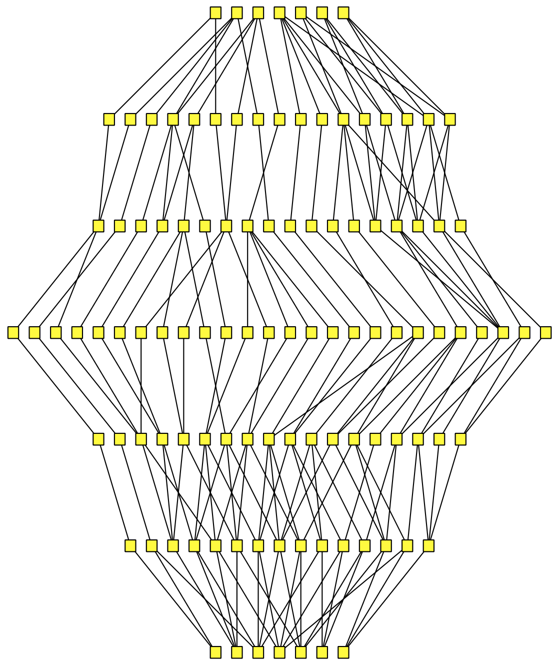


Figure 2: Example drawing: optimal ordering for *SM96-full* (graph proposed in [28]), requiring 149 crossings.

Journal on Optimization, 6:342–361, 1996.

[19] P. Hungerländer. *Algorithms and Applications for Convex Optimization*. PhD thesis, Alpen-Adria Universität Klagenfurt, 2010.

[20] P. Hungerländer and F. Rendl. Semidefinite relaxations of ordering problems. *Submitted; preprint available at http://www.optimization-online.org/DB_HTML/2010/08/2696.html*, 2010.

[21] M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In *Proceedings of the Symposium on Graph Drawing [GD'97]*, volume 1353 of *LNCS*, pages 13–24. Springer, 1997.

[22] M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, 1:1–25, 1997.

[23] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[24] R. Martí and M. Laguna. Heuristics and meta-heuristics for 2-layer straight line crossing minimization. *Discrete Applied Mathematics*, 127(3):665–678, 2003.

[25] M. May and K. Szkatula. On the bipartite crossing number. *Control and Cybernetics*, 72:85–97, 1988.

[26] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 212:307–335, 2010.

[27] H. D. Sherali and W. P. Adams. A hierarchy of relaxations

between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[28] F. Shieh and C. McCreary. Directed graphs drawing by clan-based decomposition. In *Proceedings of the Symposium on Graph Drawing [GD'95]*, volume 1027 of *LNCS*, pages 472–482. Springer, 1996.

[29] C. D. Simone. The cut polytope and the Boolean quadric polytope. *Discrete Mathematics*, 79(1):71–75, 1990.

[30] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.