Math. Prog. Comp. (2011) 3:261–280
DOI 10.1007/s12532-011-0027-7

FULL LENGTH PAPER

Optimal linear arrangements using betweenness variables

Alberto Caprara · Marcus Oswald · Gerhard Reinelt · Robert Schwarz · Emiliano Traversi

Received: 17 January 2011 / Accepted: 30 June 2011 / Published online: 2 August 2011

Abstract We solve for the first time to proven optimality the small instances in the classical literature benchmark of Minimum Linear Arrangement. This is achieved by formulating the problem as an ILP in a somehow unintuitive way, using variables expressing the fact that a vertex is between two other adjacent vertices in the arrangement. Using (only) these variables appears to be the key idea of the approach. Indeed, with these variables already the use of very simple constraints leads to good results, which can however be improved with a more detailed study of the underlying polytope.

Keywords Linear arrangement problem · Polyhedral combinatorics · Branch-and-bound algorithm · Computational results

Mathematics Subject Classification (2000) 90C27 · 90C57 · 90C90

A. Caprara (✉) · E. Traversi
DEIS, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
e-mail: alberto.caprara@unibo.it

E. Traversi
e-mail: emiliano.traversi2@unibo.it

M. Oswald · G. Reinelt · R. Schwarz
Institut für Informatik, Universität Heidelberg, Im Neuenheimer Feld 368,
69120 Heidelberg, Germany
e-mail: marcus.oswald@informatik.uni-heidelberg.de

G. Reinelt
e-mail: gerhard.reinelt@informatik.uni-heidelberg.de

R. Schwarz
e-mail: robert.schwarz@informatik.uni-heidelberg.de

1 Introduction

Minimum Linear Arrangement (MinLA) is a basic combinatorial optimization problem and calls for placing the vertices of an undirected graph $G = (V, E)$ in positions $1, \ldots, n := |V|$ along the real line so as to minimize the sum of the distances between vertices joined by an edge. Formally, one has to find a one-to-one function $\psi : V \to \{1, \ldots, n\}$, called an *arrangement*, that minimizes $\sum_{\{i,j\} \in E} |\psi(i) - \psi(j)|$.

The first thing that every mathematical programmer would notice by browsing the literature on the problem before 2010 is the huge gap between the best-known solution value, found by the wide family of heuristics available, and the best-known lower bound, for the MinLA instances in the customary benchmark, which are associated with structured sparse graphs, were introduced in [14], and since then were used as testbed for all heuristics. This is extensively discussed in [8], to which we refer for a detailed survey on the problem and a list of references. The contribution of [8] is to provide for the first time lower bounds that certify that the best-known solution value is indeed not far from the optimum for the not-too-large instances in the benchmark mentioned above. The original purpose of the authors of [8] was in fact to solve some instances to optimality, but it turned out that the mathematical formulation they used is not suited for this purpose. As a result, also after [8], the best method to solve sparse MinLA instances to optimality apparently remained the classical dynamic programming algorithm in [12], running in $\mathcal{O}(2^n m)$ time [12], where $m := |E|$. Given that its average running time tends to be close to the worst-case one, one cannot find optimal solutions for more than about 30 vertices within reasonable computing time. This leaves out also the small instances in the MinLA benchmark, for which the optimality gap was bridged by [8] from (roughly) 50 to 10%. The latter appears still fairly large as a root node gap to hope for exact solution by a branch-and-bound method, considering that the lower bound computation is time consuming anyway.

In this paper, we make a notable step ahead by solving to proven optimality 17 out of the 19 MinLA benchmark instances having up to 100 vertices. This is achieved by a using a canonical cutting plane approach to solve a non-canonical *Integer Linear Programming* (ILP) model. The key idea is the use of variables expressing the fact that one vertex is between two other adjacent vertices in the arrangement, which are certainly not the first variables one would think of to model the problem. A priori, one natural reason to use these variables is that the MinLA objective is a linear function of them, while one natural reason not to use (only) them is that it is not at all straightforward to get a valid ILP formulation, namely, a model whose solutions correspond to arrangements. In fact, we do not work with a valid ILP model in our method.

The paper is organized as follows. The formulation we use and the previous ones in the literature are discussed in Sect. 2. A study of the underlying polytope is performed in Sect. 3, whereas in Sect. 4 we present the overall branch-and-cut algorithm along with its results, discussing its merits and limitations.

We conclude the introduction by presenting the notation used in the paper. We denote by K_n the complete graph with n vertices. Given a graph $G = (V, E)$, we let $\text{LA}(G)$ denote the optimal MinLA value for G. We will always implicitly assume that G is connected, which is without loss of generality since otherwise MinLA decomposes into independent subproblems, one for each connected component of G. A *subgraph*

G' of G is always meant as an *edge-induced* subgraph, obtained by removing some edges from G and the resulting isolated vertices. The set of vertices and edges of G' are denoted by $V(G')$ and $E(G')$. Finally, given $S \subseteq V$, $G \setminus S$ denotes the subgraph of G obtained by removing all edges incident to the vertices in S and the vertices in S themselves.

2 Mathematical programming formulations

In this section we review the existing mathematical programming formulations of MinLA, trying to outline their advantages and drawbacks, focusing on the one that we use. We first define the formulations formally and then illustrate the relevant references where they are discussed. A formal study of the associated polytope is provided in Sect. 3.

2.1 Previous formulations

All combinatorial lower bounds proposed so far for MinLA, surveyed in [15], appear to be of very bad quality. This seems to indicate that any reasonable exact method should use relaxations of mathematical programming formulations.

The first variables a mathematical programming practitioner would think of to formulate MinLA are the binary x_{ij}, for $i \in V$ and $j = 1, \ldots, n$, equal to 1 if and only if vertex i is placed in position j (i.e., if and only if $\psi(i) = j$). The resulting model, with a quadratic objective function, would be a special case of the famous *Quadratic Assignment* problem:

$$\min \sum_{\{i,j\} \in E} \sum_{p=1}^{n} \sum_{q=1}^{n} |p - q| x_{ip} x_{jq},$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i \in V,$$

$$\sum_{i \in V} x_{ij} = 1, \quad j = 1, \ldots, n,$$

$$x_{ij} \in \{0, 1\}, \quad i \in V, \ j = 1, \ldots, n.$$

Our computational experience prior to [8] (unpublished) suggested that the successful approaches for Quadratic Assignment yield poor lower bounds in this case. A linearization of the above model is considered in [3], where the largest instance solved to optimality has however only 23 vertices.

The formulation providing the only reasonable lower bounds available is in fact based on $d_{\{i,j\}}$ variables, expressing the distance between the two endpoints of each edge $\{i, j\} \in E$. This very natural idea was discussed first in [13], but only its re-elaboration in [8] turned out to be successful for the MinLA benchmark instances. Natural linear constraints to impose on the $d_{\{i,j\}}$ variables are *rank inequalities*,

imposing that for certain edge-induced subgraphs G' of G the sum of the $d_{\{i,j\}}$ variables for the edges in the subgraph must be at least equal to the MinLA value for H. In formulas:

$$\sum_{\{i,j\}\in E(G')} d_{\{i,j\}} \geq \text{LA}(G'). \tag{1}$$

Such constraints are convenient to handle as long as $\text{LA}(G')$ not only is easy to determine, but is also given by a closed-form expression. For instance, if G' is a star with a center vertex and p leaves, then $\text{LA}(G') = \lfloor (p+1)^2/4 \rfloor$. However, if G is not dense the lower bound produced by solving the LP relaxation obtained by imposing (1) for whatever reasonable collection of subgraphs of G tends to be poor. (Note that the $d_{\{i,j\}}$ would be integral but, for the computation of lower bounds, relaxing the integrality constraint does not make a big difference, as it is typically the case for integer variables that may attain "large" values.) The key idea in [8] is to consider rank inequalities (1) associated also with edge-induced subgraphs G' of the complete graph K_n that are not subgraphs of G. Moreover, rather than introducing explicit $d_{\{i,j\}}$ variables not associated with edges of G, which would lead to a very large formulation, it is much better to replace each such $d_{\{i,j\}}$ in (10) by the sum of the variables associated with the edges in a path P_{ij} from i to j in G. Letting $P_{ij} = \{i, j\}$ if $\{i, j\} \in E$, the resulting inequalities read:

$$\sum_{\{i,j\}\in E(G')} \sum_{\{h,k\}\in P_{ij}} d_{\{h,k\}} \geq \text{LA}(G'). \tag{2}$$

These are called *projected rank inequalities* as they can be obtained as the projection onto $\mathbb{R}^E$ of inequalities in the space $\mathbb{R}^{\binom{n}{2}}$ associated with K_n. The reader is referred to [8] for a detailed illustration of what we have just sketched. As anticipated, for a suitable choice of subgraphs $G' \subseteq G$ for which to impose (2), the lower bounds obtained are fairly good (roughly 10% away from the optimum), but with too large gaps to hope for exact solution given their large computing times.

A formulation with binary variables $d_{\{i,j\},k}$ equal to 1 if the distance between the two endpoints of edge $\{i, j\}$ is equal to k is studied in [17]. The associated LP relaxation yields lower bounds that are slightly better than the ones in [8], but it turns out to be extremely time consuming to solve.

2.2 Betweenness variables

For every edge $\{i, j\} \in E, i < j$, and every vertex $k \in V \setminus \{i, j\}$, we introduce a binary *betweenness variable* x_{ikj} equal to 1 if and only if k lies between i and j in the arrangement, i.e., either $\psi(i) < \psi(k) < \psi(j)$ or $\psi(j) < \psi(k) < \psi(i)$. For convenience, let $T := \{ikj : \{i, j\} \in E, i < j, k \in V \setminus \{i, j\}\}$ denote the set of the indices of the betweenness variables.

The total number of betweenness variables is $m(n-2)$, i.e., a factor $O(n)$ more than the m distance variables used in [8]. The link between the two variable types

shows that the $d_{\{i,j\}}$ variables are a sort of "aggregation" of the x_{ikj} ones:

$$d_{\{i,j\}} = 1 + \sum_{k \in V \setminus \{i,j\}} x_{ikj}, \tag{3}$$

which allows us to easily express the MinLA objective with the new variables, namely:

$$\min m + \sum_{\{i,j\} \in E} \sum_{k \in V \setminus \{i,j\}} x_{ikj}, \tag{4}$$

where the constant m is of course irrelevant for the optimization. Moreover, all the constraints expressed in terms of the $d_{\{i,j\}}$ variables can be re-written in terms of the x_{ikj} variables, which we will partially do. The main advantage of working with the new variables is that we can express constraints that we would not (easily) be able to express with the old ones, leading to stronger lower bounds. In addition, given a binary vector $\bar{x} \in \mathbb{R}^T$, testing if it is associated with an arrangement can be done in polynomial time, as discussed in Sect. 4, whereas the complexity of the same test for an integer vector $\bar{d} \in \mathbb{R}^E$ is open to the best of our knowledge. Finally, since the x_{ikj} variables yield stronger lower bounds and give more information on the structure of the solution, it happens in practice that the solutions of the LP relaxations describe an optimal layout, which was never the case in [8].

Use of variables x_{ikj} was inspired by [10], which considers the following generalization of MinLA, called the *Betweenness* problem, slightly recast here to highlight the differences. One has again to find an arrangement of a given "vertex" set $V = \{1, \ldots, n\}$. In this case, rather than the edge set E, one is given as input an explicit list of q triples (i, k, j) along with a (possibly negative) penalty p_{ikj} to be paid if k is placed between i and j in the arrangement. The objective is to find an arrangement that minimizes the sum of the penalties. Note that MinLA can be formulated as a Betweenness problem by introducing the $q := m(n-2)$ triples (i, k, j) of penalty $p_{ikj} := 1$ for every edge $\{i, j\} \in E, i < j$, and every vertex $k \in V \setminus \{i, j\}$. In [10], two formulations are considered, the first with betweenness variables and linear ordering (binary) variables y_{ij} equal to 1 if and only if i precedes j in the arrangement, and the second with betweenness variables only. In fact, [10] is mainly focused on the special case in which $i \neq i'$ and $j \neq j'$ for every pair of triples (i, k, j) and (i', k', j') (which is not a generalization of MinLA any more). The polyhedral results and the computational experiments in [10] refer to this case, and show that the formulation without linear ordering variables is much better in practice.

More recently, betweenness variables were used in [4] in the context of the *Single-Row Facility Layout* problem, which is the generalization of MinLA in which G is complete, there is a cost $c_{\{i,j\}}$ (rather than 1) for each edge $\{i, j\} \in E$, and each vertex $i \in V$ has a "length" ℓ_i (rather than 1) that it occupies on the real line. The vertices must be laid out on the segment $[0, \sum_{i \in V} \ell_i]$ of the real line so as to minimize $\sum_{\{i,j\} \in E} c_{\{i,j\}} d_{\{i,j\}}$, where now $d_{\{i,j\}}$ denotes the distance between the center points of vertices i and j. (For the literature on the exact solution of this problem the reader is also referred to [1,2,5] and the older references therein.) All the discussion in [4] is focused on having the $O(n^3)$ betweenness variables associated with the complete

graph in the formulation, which would be a very large number even for the smallest instances in the MinLA benchmark. On the other hand, one of the key ingredients for the success of our approach is to stick to the $O(mn)$ variables above, which are way smaller for sparse graphs such as those in the MinLA benchmark.

2.3 Constraints on the betweenness variables

In [10], the authors discuss the fact that without linear ordering variables it is not obvious, but possible, to get a valid ILP formulation, i.e., a model whose integer solutions correspond to arrangements. Indeed, given a binary vector $\bar{x} \in \mathbb{R}^T$, it is easy to construct a binary matrix $M(\bar{x})$ such that $\bar{x}$ defines a feasible arrangement if and only if $M(\bar{x})$ has the *consecutive ones property* (*on rows*), i.e., its columns can be permuted so that the 1's in each row appear consecutively. Since there is a characterization of matrices with this property in terms of forbidden submatrices [19], it would be enough to add one constraint (on the betweenness variables) to prevent each of this submatrices to arise. Although the number of constraints would be exponential, they could be separated efficiently. However, the authors in [10] found it much more efficient in practice not to use these constraints, performing a feasibility test for each binary $\bar{x}$ encountered and, in case it turns out not to define a feasible arrangement, adding the *feasibility inequality*:

$$\sum_{t \in O} x_t - \sum_{t \in T \setminus O} x_t \leq |O| - 1, \tag{5}$$

where $O := \{t \in T : \bar{x}_t = 1\}$ is the set of the components of $\bar{x}$ taking the value 1.

In principle, by starting with the objective function (4) and no constraints (besides the bounds on the variables) and by separating constraints (5), one could set up a branch-and-cut algorithm for MinLA. However, as it is easy to imagine, this would be essentially a weird way to enumerate all the binary vectors in $\mathbb{R}^T$ in decreasing cost order. As anticipated, in the next section we will describe in detail all the constraints that we will use in our approach. However, since the simplest of these already yield notable results, we anticipate them here.

Given three vertices $i, j, k \in V, i < j < k$, associated with a triangle of G, i.e., such that $\{i, j\}, \{i, k\}, \{j, k\} \in E$, a *triangle equation* that must clearly be satisfied is:

$$x_{ikj} + x_{ijk} + x_{jik} = 1, \tag{6}$$

as one and only one of the three vertices will lie between the other two in the arrangement. Imposing these equations only, in addition to (5), does not lead to good results in itself. However, borrowing the main idea from [8], namely the use of projected rank inequalities (2) rather than rank inequalities (1) only, one may in fact impose the equivalent of all the triangle equations for the complete graph K_n, i.e., associated to all triples of vertices in V, as follows.

Consider a triangle equation (6) and suppose that $\{i, j\} \notin E$ (and also $\{i, k\} \notin E$ and $\{j, k\} \notin E$). Such an equation cannot be used as such without introducing the

new x_{ikj} variable. Rather than doing this, observe that, for any path P_{ij} in G from i to j (seen as a subset of edges), the following *path inequality* must hold for any arrangement:

$$x_{ikj} \leq \sum_{\{h,\ell\} \in P_{ij}} x_{hk\ell}, \tag{7}$$

where all variables in the right-hand-side are in our set of betweennes variables. Therefore, by replacing x_{ikj} by $\sum_{\{h,\ell\} \in P_{ij}} x_{hk\ell}$ in the triangle equation (and doing the same for x_{ijk} and x_{jik}), we get the valid *projected triangle inequality*:

$$\sum_{\{h,\ell\} \in P_{ij}} x_{hk\ell} + \sum_{\{h,\ell\} \in P_{ik}} x_{hj\ell} + \sum_{\{h,\ell\} \in P_{jk}} x_{hi\ell} \geq 1. \tag{8}$$

This is a collection of inequalities that on one hand can be separated efficiently and on the other are overall equivalent to the whole set of triangle inequalities for the complete graph K_n, where the "=" in (6) has been replaced by "≥". While by using only (6) the results are fairly poor, the addition of (8) (and no other inequality) already leads to very good results, as explained in the computational part.

Of course, what we have just sketched for the triangle inequalities can be done for every valid inequality, and this will be discussed in detail in Sect. 3.5.

3 The Betweenness Polytope

In this section we formalize most of the concepts presented informally in Sect. 2.2. We let the *betweenness polytope* $P_{\mathrm{BTW}}(G)$ be the convex hull of all vectors $x \in \mathbb{R}^T$ corresponding to arrangements. Note that this polytope is associated only with a special case of the general Betweenness problem above; nevertheless we decided to stick to this name as it immediately recalls its meaning.

Throughout the section we will need this basic observation.

Observation 1 *Given a graph G and a subgraph $H \subseteq G$, every inequality valid for $P_{\mathrm{BTW}}(H)$ is also valid for $P_{\mathrm{BTW}}(G)$.*

However, the facet-defining property is not preserved from H to G, but this does not appear to be a crucial issue in practice.

Observation 2 *Given a graph G and a subgraph $H \subseteq G$, an inequality facet defining for $P_{\mathrm{BTW}}(H)$ is not necessarily facet defining for $P_{\mathrm{BTW}}(G)$.*

Proof Consider the complete graph with four vertices K_4 and its subgraph $K_{1,3}$ with the same vertices and edges $\{1, 2\}, \{1, 3\}, \{1, 4\}$. The results in this section imply that $x_{123} + x_{124} + x_{132} + x_{134} + x_{142} + x_{143} \geq 1$ is (valid and) facet defining for $K_{1,3}$, but not for K_4 as it is the sum of the following four inequalities valid for K_4: $x_{123} + x_{124} - x_{324} \geq 0$, $x_{132} + x_{134} - x_{234} \geq 0$, $x_{142} + x_{143} - x_{243} \geq 0$, $x_{234} + x_{243} + x_{324} \leq 1$. □

3.1 Extreme-point recognition

We already mentioned the fact that testing if a binary vector $\bar{x} \in \mathbb{R}^T$ corresponds to an arrangement, which is the same as testing if it is an extreme point of $P_{\mathrm{BTW}}(G)$, is not completely trivial. Here we describe how we do it, using a slight simplification that works in the MinLA case of the general approach for the betweenness problem in [10].

Recall that a binary matrix M has the consecutive ones property if its columns can be permuted so that in each row the entries at 1 are consecutive. Given a binary $\bar{x} \in \mathbb{R}^T$, we construct the following binary matrix $M(\bar{x})$. $M(\bar{x})$ has one column for each vertex of G and initially no row. For every edge $\{i, j\} \in E$ we let $B_{ij} := \{k \in V \mid \bar{x}_{ikj} = 1\}$ be the set of vertices between i and j in the (hypothetical) arrangement corresponding to $\bar{x}$. If $B_{ij} = \emptyset$, we know that i and j must be adjacent in the arrangement, and we add to $M(\bar{x})$ a row with 1 in positions i and j and 0 in the other positions. Otherwise, all the vertices in B_{ij} must be adjacent in the arrangement (in some order), and must be preceded and followed by i and j (in some order). So we add to $M(\bar{x})$ two rows, one with 1 in positions i and $k \in V_{ij}$ and the other with ones in positions j and $k \in V_{ij}$. It is then easy to verify the following.

Proposition 1 *A binary vector $\bar{x} \in \mathbb{R}^T$ is an extreme point of $P_{\mathrm{BTW}}(G)$ if and only if $M(\bar{x})$ has the consecutive ones property.*

The consecutive ones property can be tested in linear time in the size of $M(\bar{x})$, i.e., in $O(mn)$ time, by the PQ-tree algorithm of [7].

If $\bar{x} \notin P_{\mathrm{BTW}}(G)$, as explained above we cut it off using inequality (5). Note that these inequalities are generally not even *face* inducing for $P_{\mathrm{BTW}}(G)$, i.e., their right-hand side may be decreased. On the other hand, trying to strengthen them would go in the direction of using the characterization of matrices with the consecutive ones property, which does not appear to be the right way to proceed as discussed in Sect. 2.2.

3.2 Valid equations and dimension

We already introduced the triangle equations (6). In this section, we show that these correspond to the equation system of $P_{\mathrm{BTW}}(G)$.

Proposition 2 *Let r denote the number of triangles in G, i.e., triples $\{i, j, k\} \subseteq V$ such that $\{\{i, j\}, \{i, k\}, \{j, k\}\} \subseteq E$. The affine hull of $P_{\mathrm{BTW}}(G)$ is defined by the r equations (6) and $\dim(P_{\mathrm{BTW}}(G)) = m(n-2) - r$.*

Proof Recall that $P_{\mathrm{BTW}}(G)$ lies in the $m(n-2)$ Euclidean space $\mathbb{R}^T$. The r equations (6) are linearly independent, since no variable appears in more than one of them. This implies $\dim(P_{\mathrm{BTW}}(G)) \leq m(n-2) - r$.

In order to prove equality, we first consider the special case in which G is the complete graph K_n. In this case, every triple $\{i, j, k\} \subseteq V, i < j < k$, corresponds to a

triangle, and we consider the four arrangements

$$(i, j, k, \ell_1, \ldots, \ell_{n-3}),$$
$$(j, i, k, \ell_1, \ldots, \ell_{n-3}),$$
$$(k, i, j, \ell_1, \ldots, \ell_{n-3}),$$
$$(k, j, i, \ell_1, \ldots, \ell_{n-3}),$$

where $\ell_1, \ldots, \ell_{n-3}$ are the vertices in $V \setminus \{i, j, k\}$ in arbitrary order. Let $x^1, x^2, x^3, x^4 \in P_{\mathrm{BTW}}(G)$ be the solution vectors associated with these four arrangements and $x^{ijk} := x^1 - x^2 - x^3 + x^4$. A straightforward calculation shows that the only nonzero components of x^{ijk} are $x^{ijk}_{ijk} = 2$ and $x^{ijk}_{jik} = -2$. Exchanging i and k above yields the vector y^{ijk} whose nonzero components are $y^{ijk}_{ijk} = 2$ and $y^{ijk}_{ikj} = -2$. Considering all triples $\{i, j, k\}$, we get $2\binom{n}{3}$ vectors, two for every triangle in K_n.

Now, let z denote the sum of the betweenness vectors of all arrangements, whose components are identical, and consider the collection of $2\binom{n}{3} + 1$ vectors

$$\{z\} \cup \{z + x^{ijk}, z + y^{ijk} \ : \ \{i, j, k\} \subset V\}.$$

It is easy to verify that these vectors are affinely independent and contained in the affine hull of $P_{\mathrm{BTW}}(K_n)$. This implies $\dim(P_{\mathrm{BTW}}(K_n)) \geq 2\binom{n}{3}$. Together with $\dim(P_{\mathrm{BTW}}(G)) \leq m(n-2) - r = \binom{n}{2}(n-2) - \binom{n}{3} = 2\binom{n}{3}$ this completes the proof when $G = K_n$.

Turning back to the case of general G, we complete the proof showing a contradiction if we assume that there is an equation $a^T x = b$ that is valid for $P_{\mathrm{BTW}}(G)$ but cannot be obtained as a linear combination of the triangle equations for G. By Observation 1, $a^T x = b$ is valid also for $P_{\mathrm{BTW}}(K_n)$, which in turn implies, by the discussion above, that it can be obtained as a linear combination of the triangle equations for K_n. But then, since each variable x_{ikj} associated with a triangle not in G has coefficient 0 in $a^T x = b$ and has nonzero coefficient in exactly one triangle equation in K_n, the multiplier in the linear combination for this equation must be 0, which implies that $a^T x = b$ can be obtained as a linear combination of the triangle equations in G, a contradiction. □

3.3 Relation with the cut polytope

A *cut* in a graph $G = (V, E)$ is a subset of edges $\delta(S) := \{\{i, j\} \in E : i \in S, j \in V \setminus S\}$ for some $S \subseteq V$. The *cut polytope* $P_{\mathrm{CUT}}(G)$ is the convex hull of all incidence vectors of cuts in $\mathbb{R}^E$. The link between the cut and the betweenness polytope is fairly simple.

Proposition 3 *For every $k \in V$, the projection of $P_{\mathrm{BTW}}(G)$ onto the the variable space x_{ikj}, $\{i, j\} \in E$, is isomorphic to $P_{\mathrm{CUT}}(G \setminus \{k\})$.*

Proof Consider the betweenness variables x_{ikj}, $\{i, j\} \in E$. For every arrangement, these variables take the value of the components of the incidence vector of the cut $\delta(S)$

in $G \setminus \{k\}$, where S is the set of vertices that precede k in the arrangement. This shows that every vertex of the projection, which is the projection of a vertex of $P_{\mathrm{BTW}}(G)$, is also a vertex of $P_{\mathrm{CUT}}(G \setminus \{k\})$.

Viceversa, given the incidence vector of a cut $\delta(S)$ in $G \setminus \{k\}$, the vector defined by $x_{ikj} := 1$ for $\{i, j\} \in \delta(S)$ and $x_{ikj} := 0$ for $\{i, j\} \in E \setminus \delta(S)$ is associated with every arrangement with the vertices in S preceding k and in turn k preceding all vertices in $V \setminus S$. This shows that every vertex of $P_{\mathrm{CUT}}(G \setminus \{k\})$ is also a vertex of the projection. □

An immediate corollary is:

Proposition 4 *Every valid inequality for $P_{\mathrm{CUT}}(G \setminus \{k\})$ yields a valid inequality for $P_{\mathrm{BTW}}(G)$.*

The most relevant valid inequalities for $P_{\mathrm{CUT}}(G)$ are the following *cycle inequalities*, associated with every cycle C (seen as a subset of edges) and every $F \subseteq C$ of odd cardinality. Cast in terms of the betweenness variables, with the middle index k (associated with a vertex not visited by the cycle), these inequalities read:

$$\sum_{\{i,j\} \in F} x_{ikj} - \sum_{\{i,j\} \in C \setminus F} x_{ikj} \leq |F| - 1. \tag{9}$$

Note that we have already mentioned inequalities (9) for the case $|F| = 1$, namely they are the path inequalities (7), where $C = \{i, j\} \cup P_{ij}$ and $F = \{i, j\}$. As we will see next, the inequalities for this special case play a key role in our approach, while the other inequalities (9) do not seem to be significantly helpful in practice.

3.4 Rank inequalities from the $d_{\{i,j\}}$ formulation

The link with the $d_{\{i,j\}}$ variables given by (3) immediately points out a wide family of valid inequalities for $P_{\mathrm{BTW}}(G)$, namely those obtained by replacing the $d_{\{i,j\}}$ variables in (1).

In fact, a stronger inequality in the betweenness variables associated with (1) can be obtained by considering the fact that the distance in G' (rather than G) between two vertices $i, j \in V(G')$ is given by $1 + \sum_{k \in V(G')} x_{ikj}$. The resulting *rank inequality* is:

$$\sum_{\{i,j\} \in E(G')} \sum_{k \in V(G')} x_{ikj} \geq \mathrm{LA}(G') - |E(G')|. \tag{10}$$

Here we restrict attention to the most relevant class of inequalities addressed in [8], since using also the others does not lead to significant improvements in our method.

A *p-star* $K_{1,p}$ consists of a center vertex r and p leaf vertices $\ell_1, \ldots, \ell_p$ with edges $\{r, \ell_1\}, \ldots, \{r, \ell_p\}$. As already mentioned, we have $\mathrm{LA}(K_{1,p}) = \lfloor (p+1)^2/4 \rfloor$, leading to the *star inequality*:

$$\sum_{i \in \{1,\ldots,p\}} \sum_{j \in \{1,\ldots,p\} \setminus \{i\}} x_{r\ell_j\ell_i} \geq \left\lfloor \frac{(p-1)^2}{4} \right\rfloor. \tag{11}$$

Star inequalities are by far the most important class of inequalities used in [8]. For (the dominant of) the polyhedron addressed in [8], these inequalities turn out to be facet defining for every value of p. On the other hand, for the betweenness polytope they are non-dominated only for p odd, as we prove now.

Proposition 5 *Given a p-star $K_{1,p}$, $p \geq 3$, the inequality (11) is facet-defining for $P_{\mathrm{BTW}}(K_{1,p})$ if p is odd, and is dominated by the p star inequalities associated with $K_{1,p} \setminus \{\ell_i\}, i = 1, \ldots, p$, if p is even.*

Proof We first show that the inequality is dominated if p is even. Let $p = 2k, k \in \mathbb{N}$, and consider the p subgraphs of the form $K_{1,p} \setminus \{\ell_i\}, i = 1, \ldots, p$, obtained by removing one of the leaf vertices. For each of these subgraphs, (11) holds and we can sum up all of these inequalities to get another valid inequality of the form

$$\sum_{i \in \{1,\ldots,p\}} \sum_{j \in \{1,\ldots,p\} \setminus \{i\}} (p-2) x_{r\ell_j\ell_i} \geq p \left\lfloor \frac{(p-2)^2}{4} \right\rfloor.$$

The coefficient of $p-2$ for $x_{r\ell_j\ell_i}$ is the result of summing up p inequalities, where the variable has null coefficient twice, namely for $K_{1,p} \setminus \{\ell_i\}$ and $K_{1,p} \setminus \{\ell_j\}$. By dividing all coefficients by $p-2$, a straightforward calculation shows that we get a right-hand side of $\lfloor (p-1)^2/4 \rfloor$, which is in fact equal to the right-hand side of (11) for $K_{1,p}$. Therefore, (11) is not facet-defining for p even.

Let us now consider (11) for any odd $p = 2q+1$, $q \in \mathbb{N}$, and assume for simplicity $r < \ell_i$ for $i = 1, \ldots, p$. All arrangements with r in position $q+1$ or $q+2$ are optimal. Consider an inequality $a^T x \geq b$ that is valid for $P_{\mathrm{BTW}}(K_{1,p})$ and is satisfied at equality by all points for which (11) is satisfied at equality. Let $\{i, j, k\} \subseteq \{\ell_1, \ldots, \ell_p\}$ denote three arbitrary leaf vertices of $K_{1,p}$, and consider the five optimal arrangements:

$$\begin{aligned}
&(h_1, \ldots, h_{q-1}, k, r, j, i, h_q, \ldots, h_{p-3}),\\
&(h_1, \ldots, h_{q-1}, k, r, i, j, h_q, \ldots, h_{p-3}),\\
&(h_1, \ldots, h_{q-1}, k, i, r, j, h_q, \ldots, h_{p-3}),\\
&(h_1, \ldots, h_{q-1}, j, r, i, k, h_q, \ldots, h_{p-3}),\\
&(h_1, \ldots, h_{q-1}, j, i, r, k, h_q, \ldots, h_{p-3}),
\end{aligned}$$

where $h_1, \ldots, h_{p-3}$ are the vertices in $\{\ell_1, \ldots, \ell_p\} \setminus \{i, j, k\}$ in arbitrary order. Let $x^1, x^2, x^3, x^4, x^5 \in P_{\mathrm{BTW}}(G)$ be the solution vectors associated with these five arrangements, all satisfying $a^T x \geq b$ at equality. This implies

$$a^T x^1 - a^T x^2 = a_{rji} - a_{rij} = 0$$

and

$$a^T x^2 - a^T x^3 - a^T x^4 + a^T x^5 = 2a_{rij} - 2a_{rik}.$$

Given that the choice of $\{i, j, k\}$ is arbitrary, all left-hand side coefficients of $a^T x \geq b$ are identical, which in turn implies that the inequality is identical to (11) modulo a positive scaling factor. This implies that (11) is facet-defining. □

3.5 Projected valid inequalities of $P_{\mathrm{BTW}}(K_n)$

As anticipated in the previous section, and widely inspired by [8], the success of a cutting plane method is essentially based on considering in a cutting plane approach also inequalities that are valid for $P_{\mathrm{BTW}}(K_n)$ without enlarging the number of variables from $m(n-2)$ to $\binom{n}{3}$. Let $P^D_{\mathrm{BTW}}(G) := \{x \in \mathbb{R}^T : x \geq y \text{ for some } y \in P_{\mathrm{BTW}}(G)\}$ be the *dominant* of $P_{\mathrm{BTW}}(G)$.

Observation 3 *The minima of a linear objective function with nonnegative coefficients (such as the MinLA one) over $P_{\mathrm{BTW}}(G)$ and $P^D_{\mathrm{BTW}}(G)$ coincide, and all valid inequalities for $P^D_{\mathrm{BTW}}(G)$ are of the form $a^T x \geq b$ with $a \geq 0$.*

In some sense, this observation suggests that, in order to find good lower bounds, one may restrict attention to inequalities in "$\geq$" form with positive left-hand-side coefficients. These are precisely the inequalities that we know how to extend from $P_{\mathrm{BTW}}(K_n)$ to $P_{\mathrm{BTW}}(G)$.

Proposition 6 *Let*

$$\sum_{\{i,j\}\in E} \sum_{k\in V\setminus\{i,j\}} a_{ikj} x_{ikj} \geq b \tag{12}$$

be an inequality with $a \geq 0$ that is valid for $P_{\mathrm{BTW}}(K_n)$, and for each $i, j \in V$ let $P_{ij} \subseteq E$ be an arbitrary path from i to j in G. Then, the projected inequality

$$\sum_{\{i,j\}\in E} \sum_{k\in V\setminus\{i,j\}} a_{ikj} \sum_{\{h,\ell\}\in P_{ij}} x_{hk\ell} \geq b \tag{13}$$

is valid for $P_{\mathrm{BTW}}(G)$.

The proof of the proposition follows immediately from the validity of (7), i.e., of (9) for $|F| = 1$. The name "projected inequality" follows from the fact that these arise from the projection onto the space of $P_{\mathrm{BTW}}(G)$ of inequalities for $P_{\mathrm{BTW}}(K_n)$.

Considering the general version of (9), we may generalize Proposition 6 substituting x_{ikj} by $\sum_{\{h,\ell\}\in P_{ij}\setminus F} x_{hk\ell} - \sum_{\{h,\ell\}\in P_{ij}\cap F} x_{hk\ell} + |F| - 1$ for every $F \subseteq P_{ij} \cup \{i, j\}$ with $|F|$ odd and $\{i, j\} \in F$. Although the complexity (and practical difficulty) of the separation of the resulting inequalities for $P_{\mathrm{BTW}}(G)$ would not change substantially, after extensive computational testing we concluded that restricting attention to the substitution in Proposition 6 yields already the best results for the MinLA instances.

3.6 Separation

We briefly address the complexity of the separation problem for all the inequalities discussed so far for a given LP solution $\bar{x} \in \mathbb{R}^T$.

Observation 4 *Feasibility inequalities* (5) *can be separated in* $O(m^2n^2)$ *time.*

Proof Inequality (5) can be rewritten as $\sum_{t\in O}(1-x_t)+\sum_{t\in T\setminus O}x_t \geq 1$, i.e., separation calls for finding $O \subseteq T$ such that $\sum_{t\in O}(1-\bar{x}_t)+\sum_{t\in T\setminus O}\bar{x}_t < 1$. This may hold only if (i) $\bar{x}_t \leq 1/2$ for at most one $t \in O$, (ii) $\bar{x}_t \geq 1/2$ for at most one $t \in T \setminus O$, and (iii) $|\{t \in O : \bar{x}_t \leq 1/2\}| + |\{t \in T \setminus O : \bar{x}_t \geq 1/2\}| \leq 1$. This implies that there are only $O(|T|) = O(mn)$ candidates for O. For each of them, we can test if (5) is violated and, if so, if it is valid (i.e., if setting to 1 the components in O and to 0 the components in $T \setminus O$ yields an infeasible binary solution) in $O(mn)$ time by the PQ-tree algorithm in [7]. □

In fact, it seems to make sense in practice to add inequalities (5) only when $\bar{x}$ is integer but not feasible, so the above observation should have no practical impact.

Observation 5 *Triangle equations* (6) *can be separated in* $O(mn)$ *time.*

Proof It suffices to enumerate the $O(mn)$ triangles in G, testing violation of the associated inequality in constant time. □

Observation 6 *Path inequalities* (7) *can be separated in* $O(n\pi(n,m))$ *time, where* $\pi(n,m)$ *is the time required to find all-pairs shortest paths in a graph with n vertices and m edges.*

Proof There are n ways to fix the middle index k. For each k, we can compute the shortest path between all pairs i, j with edge lengths $\bar{x}_{hk\ell}$ for $\{h,\ell\} \in E$ and then check the inequality (7) with x_{ikj} in the left-hand side for violation. □

It is not difficult to see, and well-known from the max-cut literature [6], how to extend this to the separation of the more general cycle inequalities (9); here we do not give the details since these inequalities are not used in our approach.

As to rank inequalities (10), note that by relaxing the condition $k \in V(G')$ by $k \in V$ in the second summation, i.e., by allowing also vertices outside G' as middle indices, one may resort to the separation algorithms in [8] by defining $\bar{d}_{\{i,j\}} := 1+\sum_{k\in V\setminus\{i,j\}}\bar{x}_{ikj}$ for $\{i,j\} \in E$. In particular, for the separation of star inequalities we get.

Proposition 7 *The relaxed version of star inequalities* (11)*, in which* $j \in \{1,\ldots,p\}\setminus\{i\}$ *is replaced by* $j \in V\setminus\{i\}$ *in the second summation, can be separated in* $O(mn)$ *time.*

Proof The time is dominated by the one to define $\bar{d}$ from $\bar{x}$, which is linear in the number of (nonzero) components of x, since afterwards the algorithm mentioned in [8] runs in $O(m \log n)$ time. □

The complexity of the separation of the original (stronger) (11) remains open.

Finally, for the separation of projected inequalities, as in [8], we have:

Proposition 8 *Every separation algorithm for a class of valid inequalities for* $P_{\mathrm{BTW}}(K_n)$ *of the form* (12) *yields a separation algorithm for the corresponding valid inequalities for* $P_{\mathrm{BTW}}(G)$ *of the form* (13) *whose complexity is the same after having computed all-pairs shortest paths as in Observation 6.*

Proof Letting $\bar{P}_{ij}^k$ be the shortest path from between i and j with respect to edge lengths $\bar{x}_{ikj}$, one can restrict attention to inequalities (13) of the form

$$\sum_{\{i,j\}\in E} \sum_{k\in V\setminus\{i,j\}} a_{ikj} \sum_{\{h,\ell\}\in \bar{P}_{ij}^k} x_{hk\ell} \geq b,$$

that can be separated as (12) after having redefined $\bar{x}_{ijk} := \sum_{\{h,\ell\}\in \bar{P}_{ij}^k} \bar{x}_{hk\ell}$. □

4 The overall method and computational results

Our overall algorithm is a branch-and-cut algorithm based on the formulation presented in the previous sections. At the beginning, we call an external heuristic to initialize the best solution found. This is a multi-start local search heuristic already used in [8].

The initial LP relaxation contains all the equations (6) associated with the triangles of G (and no other constraint besides the lower and upper bounds on the variables). Preliminary computational tests have shown that it is better to add all these equations from the beginning rather than separating them afterwards.

At each iteration of the cutting-plane process at each branch-and-cut node, if the current LP solution is integer, we check if it is feasible (and optimal) and if not we add to the LP the corresponding feasibility inequality (5). Otherwise, we separate the other inequalities in the following order (i) path inequalities (7); (ii) odd star inequalities (11); (iii) projected triangle inequalities (8); (iv) projected odd star inequalities. We impose upper limits of 3,000 and 300, respectively, on the total number of cuts to be separated on the complete and original graph. If a limit is reached before all the classes of inequalities are checked, we allow in any case the addition of 600 and 60, respectively, extra inequalities for each remaining class on the complete and original graph. In the separation of projected inequalities in Steps (iii) and (iv), the all-pairs shortest paths (recall Proposition 8) have already been computed in Step (i).

4.1 Implementation details

We implemented our algorithm using SCIP as a branch-and-cut framework [20] with CPLEX 12 as LP solver. All these tools are freely available for academic use. We kept the default settings of SCIP for branching and addition of cutting planes to the current LP, but we deactivated the heuristic and cutting-plane generation parts. In other words, we generate cutting planes only with our separation procedures but let SCIP decide which ones to add to the current LP.

We made our experiments on a PC running GNU/Linux Debian 4.1.1, kernel 2.6.18, with an Intel Xeon CPU at 2.50GHz, a cache size of 6MB and a RAM of 16GB.

4.2 The benchmark instances

We considered all the benchmark instances addressed in [8] with $m \leq 1000$. These include 4 out of the 5 `gd*` instances in the classical MinLA benchmark [14]; all the other instances in this benchmark are (way) larger. Moreover, the instances in [8] also include the instances of the *Minimum Bandwidth* problem in [9], which is the variant of MinLA in which one wants to find an arrangement ψ minimizing $\max_{\{i,j\}\in E} |\psi(i) - \psi(j)|$. In addition, we considered five additional instances coming from real-world applications introduced in [17]; these instances are at the bottom of Table 1.

In Table 1 we list these instances summarizing their characteristics. The density *dens* is defined by $m/\binom{n}{2}$, *dim* is the dimension of the associated betweenness polytope $P_{\mathrm{BTW}}(G)$, and *lb* and *ub* are the best lower bound and the best heuristic solution value known before our method, to the best our knowledge. All values of *lb* are due to [8], with the exception of those for `gd95c` and `gd96c`, that were found in a few hours by the cutting-plane approach for the Single-Row Facility Layout problem in [5]. The values of *ub* are generally due to one or more heuristics in the literature for instances `gd*`, whereas they were obtained by running for one CPU day the multi-start heuristic also used (with a much smaller time limit) at the beginning of our method. We partly anticipate our final results (and contribution) by marking in boldface optimal values. This marking shows that, out of 41 instances, optimality was proved only in one case before our work and in 21 cases with our method. In 2 of the additional cases, our method found an optimal heuristic solution showing that the best known heuristic value was not optimal.

4.3 Impact of the various inequalities on the root lower bound

In Table 2 we report the lower bound *lb* found at the root node of our branch-and-cut algorithm and the associated computing time in seconds *time*, by separating various subsets of inequalities illustrated in Sect. 3 (recall that we add feasibility inequalities (5) in any case to cut off infeasible binary solutions), from *none* (keeping only the triangle equations (6), which define the initial LP relaxation) to *all* those listed above, considering also the case of *additional* inequalities discussed, e.g., in [16] and [18]. We restrict attention to instances with up to 100 vertices and impose a time limit of one day (86400 seconds).

The results show that the projected triangle inequalities (8) are essential to achieve good lower bounds: without them (column (7), (11)) the average gap with respect to the optimal (or best known) solution value is about 30%, while with them (column (7), (8)) for 20 instances the root node bound is equal to the optimum, for one additional instance `will57` it is one unit away from the optimum, and for 3 additional instances the time limit is reached with gaps of about 1% for `can___73`, 4% for `impcol_b`, and 19% for `can___96`, which is the only instance for which the gap is large, anticipating the fact that for more than 100 vertices our method does not perform very well. With respect to the separation of (7), (8) only, considering also other inequalities does not change the results in a significant way. Namely, for the 3 instances not solved to optimality, for *all* the gaps are about 1.5% for `can___73`,

Table 1 The benchmark instances

Name	n	m	dens	$\|T\|$	dim	lb	ub
gd95c	62	144	0.076	88	8552	472	**506**
gd96b	111	193	0.032	0	21037	1281	1416
gd96c	65	125	0.060	20	7855	464	**519**
gd96d	180	228	0.014	0	40584	2021	2391
bcspwr01	39	46	0.062	2	1700	91	**106**
bcspwr02	49	59	0.050	3	2770	144	**161**
bcspwr03	118	179	0.026	23	20741	588	679
bcspwr04	274	669	0.018	582	181386	3700	4705
bcsstk01	48	176	0.156	160	7936	972	**1132**
can___24	24	68	0.246	60	1436	203	**210**
can___61	61	248	0.135	396	14236	1119	**1137**
can___62	62	78	0.041	2	4678	187	212
can___73	73	152	0.057	32	10760	971	1100
can___96	96	336	0.073	320	31264	2105	2702
can___144	144	576	0.055	912	80880	2304	3224
can___161	161	608	0.047	592	96080	5657	6696
can___187	187	652	0.037	620	120000	3827	5189
can___229	229	774	0.029	690	175008	7461	9707
dwt___59	59	104	0.060	30	5898	258	**289**
dwt___66	66	127	0.059	62	8066	**192**	**192**
dwt___72	72	75	0.029	0	5250	150	**167**
dwt___87	87	227	0.060	147	19148	897	**932**
dwt___162	162	510	0.039	464	81136	2032	2431
dwt___209	209	767	0.035	707	158062	5905	6387
dwt___221	221	704	0.028	608	153568	3603	3779
dwt___245	245	608	0.020	374	147370	3422	3860
bwm200	200	298	0.014	0	59004	495	**496**
fidap005	27	126	0.358	288	2862	412	**414**
fidapm05	42	239	0.277	671	8889	998	**1003**
lshp-265	265	744	0.021	480	195192	5602	6088
nos4	100	247	0.049	36	24170	976	**1031**
pde225	225	420	0.016	0	93660	2539	3040
rdb200	200	460	0.023	0	91080	3066	3768
saylr1	238	445	0.015	0	105020	2676	3107
steam3	80	424	0.134	992	32080	1406	**1416**
tub100	100	148	0.029	0	14504	245	**246**
curtis54	54	124	0.086	78	6370	417	**454**
ibm32	32	90	0.181	28	2672	440	**485**
impcol_b	59	281	0.164	285	15732	1810	2076
pores_1	30	103	0.236	88	2796	329	**383**
will57	57	127	0.079	94	6891	331	**335**

Table 2 Root node lower bounds and running times for various combinations of the inequalities in the formulation

Name	None		(7)		(7), (11)		(7), (8)		All		Additional	
	lb	time	lb	time	lb	time	lb	time	lb	time	lb	time
gd95c	232	0.30	260	1.70	368	6.34	506	60.68	506	45.38	506	52.35
gd96c	145	0.34	185	3.29	253	3.37	519	378.70	519	591.34	519	461.71
bcspwr01	48	0.05	49	0.13	68	0.12	106	0.77	106	0.64	106	1.24
bcspwr02	62	0.12	63	0.27	100	0.41	161	3.81	161	1.90	161	4.01
bcsstk01	336	0.19	490	3.54	538	31.97	1132	8156.78	1132	6594.60	1132	7841.00
can___24	128	0.02	140	0.13	169	2.70	210	0.69	210	0.73	210	1.14
can___61	644	0.40	700	13.07	911	62.54	1137	300.92	1137	272.33	1137	388.09
can___62	80	0.25	82	0.59	122	0.93	210	10.19	210	10.81	210	12.47
can___73	184	0.49	216	4.08	382	4.45	1086	limit	1083	limit	1073	limit
can___96	656	1.47	800	12.84	916	119.03	2180	limit	2076	limit	2052	limit
dwt___59	134	0.24	148	1.23	187	1.32	289	15.23	289	11.03	289	19.97
dwt___66	189	0.33	189	0.59	192	0.93	192	0.50	192	0.74	192	1.98
dwt___72	76	0.56	76	0.57	88	0.89	167	4.28	167	5.86	167	13.80
dwt___87	374	0.98	490	11.69	583	19.41	932	1034.00	932	1895.50	932	1281.38
fidap005	414	0.05	414	3.51	414	2.89	414	0.43	414	0.44	414	0.89
fidapm05	910	0.19	914	33.39	987	116.33	1003	14.62	1003	19.95	1003	42.20
nos4	283	1.49	301	5.59	634	12.44	1031	6865.71	1031	8092.64	1031	2464.71
steam3	1416	1.00	1416	36.48	1416	6.56	1416	3.11	1416	3.06	1416	10.13
tub100	149	1.77	149	1.80	244	1.70	246	6.18	246	2.42	246	10.41
curtis54	202	0.21	223	2.02	294	5.24	454	31.14	454	33.40	454	33.72
ibm32	118	0.06	160	0.50	261	4.25	485	156.81	485	227.37	485	185.83
impcol_b	566	0.37	685	11.85	1390	1812.15	1993	limit	2016	limit	2029	limit
pores_1	191	0.05	191	0.17	287	3.90	383	4.50	383	5.72	383	7.03
will57	221	0.23	224	1.06	284	2.44	334	4.40	334	15.67	334	15.36

3% for impcol_b, and 23% for can___96, whereas for *additional* the gaps are about 2.5% for can___73, 2% for impcol_b, and 24% for can___96. For the other 21 instances, we computed the ratio between the time taken by a method and the minimum over the three methods ((7), (8), *all* and *additional*). The average value of this ratio is about 1.3 for (7), (8), 1.4 for *all* and 2 for *additional*. Therefore, *additional* is a bit slower while (7), (8) and *all* have analogous performances, and we decided to keep the latter for the experiments in the next section.

4.4 Final results

With the present technology, for our benchmark instances our method performs very well for sparse graphs with $n \leq 100$ while it is often not competitive with the method of [8] for larger n. In Table 3, we report the results obtained by the two methods.

Table 3 Comparison of the method in [8] and our method for the benchmark instances with $n \leq 100$ (above) and $n > 100$ (below)

Name	[8]		Our method		
	lb	time	lb	time	nodes
gd95c	443	113.6	506	82.24	1
gd96c	402	218.1	519	588.56	1
bcspwr01	91	0.8	106	9.09	145
bcspwr02	144	2.0	161	9.96	21
bcsstk01	971	23232.8	1132	6225.68	1
can___24	203	2.8	210	2.79	1
can___61	1119	1221.4	1137	333.41	1
can___62	187	4.2	210	39.74	39
can___73	971	2016.8	1083	limit	1
can___96	2105	27786.0	2076	limit	1
dwt___59	258	55.4	289	27.39	1
dwt___66	192	1.7	192	33.93	1
dwt___72	150	6.7	167	21.24	1
dwt___87	897	20761.1	932	1901.43	1
fidap005	412	5.4	414	4.17	1
fidapm05	998	7492.9	1003	42.36	1
nos4	976	59118.0	1031	6575.36	1
steam3	1382	limit	1416	168.44	1
tub100	245	74.3	246	69.06	1
curtis54	417	92.0	454	54.49	1
ibm32	440	1642.0	485	250.50	1
impcol_b	1810	limit	2016	limit	1
pores_1	329	4.0	383	9.59	1
will57	331	2.0	335	30.47	3
gd96b	1281	493.5	1404	limit	1711
gd96d	2021	1669.0	1578	limit	1
bcspwr03	588	189.5	662	4690.12	1
bcspwr04	3696	limit	2596	limit	1
can___144	2304	19608.4	2873	limit	1
can___161	5657	limit	3187	limit	1
can___187	3827	limit	2977	limit	1
can___229	7461	limit	3377	limit	1
dwt___162	2032	limit	2281	limit	1
dwt___209	5905	limit	3612	limit	1
dwt___221	3603	limit	2747	limit	1
dwt___245	3422	limit	2426	limit	1
bwm200	495	2409.1	496	548.20	1
lshp-265	5497	limit	3234	limit	1
pde225	2539	70519.0	1805	limit	1
rdb200	3052	limit	2077	limit	1
saylr1	2673	limit	1792	limit	1

The computing times are on different computers but with roughly comparable speeds. In both cases the time limit is again one day, and for our method we also report the number of branch-and-cut *nodes* explored.

Table 3, above, contains the the benchmark instances with $n \leq 100$. As already mentioned, we can solve almost all of them (21 out of 24) to proven optimality within a computing time that is roughly comparable to the time required by the method in [8] to find a lower bound with an average gap of about 7.5%. On the other hand, Table 3, below, contains the benchmark instances with $n > 100$ and shows that we can solve to proven optimality 2 out of the 17 instances, but for the remaining ones only in 3 cases the lower bound found by our method is better than the one found by [8].

These results essentially show that the current state-of-the-art in the practical solution of MinLA on sparse graphs is given by our method for $n \leq 100$ and by [8] for larger values of n, even if also application of the latter is currently limited to $m \leq 5{,}000$.

4.5 Concluding remarks

With our method we could solve most of the MinLA benchmark instances with up to 100 nodes, a goal that was widely out of reach for previous methods. For the larger instances in our benchmark, our method suffers from the fact that, once at the root node we are close to the final bound, on one hand addition of new inequalities does not lead to significant improvements, and on the other the gap, though small in percentage, is too large to be closed by branching (also considering that the exploration of each branch-and-bound node is slow anyway as the current LP is large). We have made a few unsuccessful attempts to overcome this drawback, in particular hoping to solve `gd96b` and `gd96d`, but without success.

As to the large instances in the classical MinLA benchmark from [14], these have to be handled with care already by the method in [8], so there seems to be no hope to obtain any meaningful result using our approach with the current hardware and software technology.

Acknowledgments This work was supported by the Ateneo Italo-Tedesco and the Deutsch-Italienisches Hochschulzentrum under the Vigoni Project 2007-2008. We are grateful to JJ Salazar for having run the code of [8] on the 5 new MinLA instances considered in this paper. Moreover, we are grateful to the two referees for their comments.

References

1. Amaral, A.R.S.: On the exact solution of a facility layout problem. Eur. J. Oper. Res. **173**, 508–518 (2006)
2. Amaral, A.R.S.: An exact approach to the one-dimensional facility layout problem. Oper. Res. **56**, 1026–1033 (2008)
3. Amaral, A.R.S.: A mixed 0-1 linear programming formulation for the exact solution of the minimum linear arrangement problem. Optim. Lett. **3**, 513–520 (2009)
4. Amaral, A.R.S.: A new lower bound for the single row facility layout problem. Discr. Appl. Math. **157**, 183–190 (2009)

5. Amaral A.R.S., Letchford A.N.: A polyhedral approach to the single row facility layout problem. http://www.optimization-online.org/DB_FILE/2008/03/1931.pdf (in preparation) (2011)
6. Barahona, F., Mahjoub, A.R.: On the cut polytope. Math. Program. **36**, 157–173 (1986)
7. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**, 335–379 (1976)
8. Caprara, A., Letchford, A.N., Salazar, J.J.: Decorous lower bounds for minimum linear arrangement. INFORMS J. Comput. **23**, 26–40 (2011)
9. Caprara, A., Salazar, J.J.: Laying out sparse graphs with provably minimum bandwidth. INFORMS J. Comput. **17**, 356–373 (2005)
10. Christof T., Oswald M., Reinelt G.: Consecutive ones and a betweenness problem in computational biology. In: Bixby R.E., Boyd E.A., Rios-Mercado R.Z. (eds.) Proceedings of 6th Conference on Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science, vol. 1412, pp. 213–228. Springer, Berlin (1998)
11. Díaz, J., Petit, J., Serna, M.: A survey of graph layout problems. ACM Comput. Surv. **34**, 313–356 (2002)
12. Koren Y., Harel D.: A multi-scale algorithm for the linear arrangement problem. In: Kucera L. (ed.) Proceedings of 28th International Workshop on Graph-Theoretic Concepts in Computer Science. Lecture Notes in Computer Science, vol. 2573, pp. 293–306. Springer, Berlin (2002)
13. Liu, W., Vannelli, A.: Generating lower bounds for the linear arrangement problem. Discr. Appl. Math. **59**, 137–151 (1995)
14. Petit J.: Layout Problems. PhD thesis, Universitat Politècnica de Catalunya, Barcelona (2001)
15. Petit J.: Experiments on the linear arrangement problem. J. Exp. Algorithmics, **8** (2003). article 2.3
16. Schwarz R.: A Branch-and-Cut Algorithm with Betweennes Variables for the Linear Arrangement Problem. Master Thesis, Universität Heidelberg (2010)
17. Seitz H.: Contributions to the Minimum Linear Arrangement Problem. PhD Thesis, Universität Heidelberg (2010)
18. Traversi E.: Orientation and Layout Problems on Graphs, with Applications. PhD Thesis, Università di Bologna (2010)
19. Tucker, A.: A structure theorem for the consecutive ones property. J. Comb. Theory Ser. B **12**, 153–162 (1972)
20. ZIB Optimization Suite (Version 1.2), Zuse Institut Berlin. http://zibopt.zib.de (2009)