Maastricht University

Alexander Grigoriev, Hans Ensinck,
Natalya Usotskaya

Integer linear programming formulations for treewidth

RM/11/030

METEOR

Maastricht University School of Business and Economics
Maastricht Research School of Economics
of Technology and Organization

P.O. Box 616
NL - 6200 MD Maastricht
The Netherlands

Integer Linear Programming Formulations for Treewidth

Alexander Grigoriev, Hans Ensinck and Natalya Usotskaya

Department of Quantitative Economics, Maastricht University,
P.O. Box 616, 6200 MD Maastricht, The Netherlands
{a.grigoriev,j.ensinck,n.usotskaya}@maastrichtuniversity.nl

Abstract. In this paper we consider an ILP-based approach to tackle the problem of determining a treewidth of the graph. We give an overview of existing attempts and develop further LP-based techniques for the problem. We present two different ILP formulations for the treewidth. The first one is the merge of chordalization-based ILP by Koster and Bodlaender and flow metrics approach by Bornstein and Vempala. The second brand-new ILP is based on the structural properties of the tree-decomposition. It has a nice application of the local branching techniques by Fischetti and Lodi.

1 Introduction

There are various algorithms and heuristics for the problem of finding the treewidth of a graph. One of the most known exact algorithms is Bodlaender's algorithm running in $O(2^{tw^2}n)$-time, where tw is the treewidth of a graph, see [2]. Unfortunately it can not be used in practice because of the big constant hidden in O-notation. The best known exponential-time algorithm is the dynamic programming by Fomin et al. [12] with the running time $O(1.8899^n)$.

For a long time the best known approximation algorithm for the treewidth had a factor $O(\log n)$, see [4], and $O(\log tw)$, see [1]. Recently, Feige et al. developed a polynomial-time $O(\sqrt{\log tw})$-approximation algorithm, see [9]. It is still an intriguing open question whether the treewidth can be approximated within a constant factor for an arbitrary graph.

So far only limited attempts were made to attack the treewidth problem using Integer Linear Programming techniques. In this paper we present an overview of these attempts and develop further LP-based techniques for the problem. Integer Linear Programming is widely used to obtain good exact and approximation algorithms for different problems, see i.e. [7, 10, 15–17]. The advantage of this approach is that it is very generic and, therefore, we can apply many different well-developed LP-techniques to obtain better quality solutions or to speed-up the algorithm.

We present two different ILP formulations for the treewidth. The first ILP, *Elimination Order formulation*, is proposed by Koster and Bodlaender [14]. It is based on the vertex elimination order and the connection between the treewidth and the chordalization of the graph, see Bodlaender [3] for the details. We briefly show that the integrality gap of this formulation is at least $\Omega(\sqrt{n})$. We improve this formulation by merging it with the flow metrics approach by Bornstein and Vempala [7]. The resulting new ILP cuts some feasible symmetric solutions in the linear relaxation of the ILP by Koster and Bodlaender by new inequalities. Further, we introduce a brand-new ILP formulation for the treewidth problem based on "geometry" of the tree-decomposition, so called *Tree Drawing formulation*. This new formulation has nice geometric properties which we use to apply the local branching techniques by Fischetti and Lodi [11], a novel branching technique for branch-and-bound algorithms.

The paper is organized as follows: in Section 2 we give some basic definitions and theorems. Section 3 is devoted to Elimination Order formulations and its improvement using flow metrics techniques by Bornstein and Vempala. Section 4 contains new Tree Drawing formulation and the application of the local branching technique by Fischetti and Lodi to the proposed ILP. In Section 5

we compare two formulations and speculate on their applicability for different types of graphs. The last section contains the results and open questions.

2 Preliminaries

In this section, we formally define treewidth and a few related terms. The notions of treewidth and tree decomposition were introduced by Robertson and Seymour in [18]. Apart from their definitions, we introduce some fundamental lemma's regarding treewidth that are essential below.

Definition 1. *A* tree decomposition *of a graph G is a pair (T, X), where T is a tree and $X = (X_t : t \in V(T))$ is a family of subsets of $V(G)$, with the following properties:*

- $\bigcup_{t \in V(T)} X_t = V(G)$.
- $\forall uv \in E(G)$, $\exists t \in V(T)$ *such that* $\{u, v\} \subseteq X_t$.
- *For $t, t', t'' \in V(T)$, if t' is on the unique path in T between t and t'' then $X_t \cap X_{t''} \subseteq X_{t'}$.*

For $t \in V(T)$, the set X_t is also referred to as a bag *of the tree decomposition (T, X).*

The *width* of the tree decomposition (T, X) is

$$\max_{t \in V(t)} (|X_t| - 1).$$

Definition 2. *The* treewidth *$tw(G)$ of graph G is the minimum width over all tree decompositions of G.*

A lot of research has been dedicated to the fixed parameter case for treewidth, i.e., check whether the treewidth of a graph is at most some constant w and if so, return a tree decomposition of width at most w. For an overview of this work we refer to [3]. Finally, the following result was obtained.

Theorem 1. *[2] Given a graph of treewidth at most w, a tree decomposition of width at most w can be obtained in linear time.*

As we noticed above, from a practical viewpoint the algorithm is only useful for very low values of w.

An edge which joins two vertices of a cycle but is not itself an edge of the cycle is called a *chord* of that cycle. A *chordless* cycle in G is an induced cycle in G, i.e., a cycle that forms an induced subgraph of G. A graph is called *chordal* (or *triangulated*) if it does not contain chordless cycles of length greater than 3, i.e., if all induced cycles in the graph are 3-cycles (also called triangles). A graph can be transformed into a chordal graph by adding edges to it up to the point where every cycle contains a chord. Edges that are added to achieve chordality are called *fill-in edges.* We denote the set of fill-in edges by F.

Definition 3. *A graph $H = (V, E \cup F)$ is called a* chordalization *(or* triangulation*) of $G = (V, E)$ if H is chordal.*

The *width* of a chordal graph H is equal to $\omega(H) - 1$, i.e., the size of the largest clique in H minus one. Using the notion of chordalization, treewidth can be alternatively defined in the following way.

Theorem 2. *see, e.g. [3] The treewidth $tw(G)$ of graph G is the minimum width over all chordalizations of G.*

Algorithm 1.1: chordalization

Input: graph $G = (V, E)$ and vertex ordering α
Output: chordalization of G

for $i = 1$ *to* $n - 2$ **do**
  make $S_i = \{ v \in V \mid v \in N(\alpha(i)) \;\&\&\; \alpha(v) > i) \}$ a clique by adding fill-in edges (if necessary);

An *ordering* of the vertex set V is a bijection $\alpha : \{1, 2, \ldots, n\} \longleftrightarrow V$. If α is an ordering on n vertices, then we will also refer to α as $\alpha(1)\alpha(2)\ldots\alpha(n)$. A chordalization of $G = (V, E)$ can be obtained using vertex ordering α of V by application of Algorithm 1.1.

In words, we run through the vertex ordering and turn the higher ordered neighbors of the vertex at hand into a clique by adding fill-in edges to G. Note that the added fill-in edges also define neighbor relations in subsequent steps of the algorithm.

The process of turning the higher ordered neighbors of vertex $\alpha(i)$ into a clique is often called the *elimination* of vertex $\alpha(i)$. In the context of chordalizations, vertex orderings are therefore often referred to as *elimination orderings*. An elimination ordering α on the vertices of G is called *perfect* if during Algorithm 1.1 no fill-in edges are added to G, i.e., if for all vertices $v \in V$ the set of higher ordered neighbors already forms a clique in G.

3 Elimination Order formulation and flow metrics

Elimination Order formulation for the treewidth is introduced by Bodlaender and Koster [14]. The formulation is based on the relationship between treewidth and chordalizations of graphs, see Section 2 and Bodlaender [3] for the details. Theorem 2 states that finding the treewidth of a graph G is equivalent to finding a triangulation of the graph G with minimum clique size. A graph is triangulated if and only if it has a *perfect elimination scheme*. The idea is to determine the best elimination order of the vertices. The maximum outdegree among the vertices in an elimination scheme is equal to the width of the tree decomposition of G associated with the triangulation. Modeling perfect elimination orders let the decision variables of ILP be defined as follows:

$$x_{ij} = \begin{cases} 1, \text{ if } i \text{ is ordered before } j \text{ in the perfect elimination scheme,} \\ 0, \text{ otherwise.} \end{cases}$$

$$y_{ij} = \begin{cases} 1, \text{ if } i \text{ is ordered before } j \text{ in the perfect elimination scheme} \\ \quad \text{and if } ij \text{ is an edge of the triangulation,} \\ 0, \text{ otherwise.} \end{cases}$$

We assume that $i \neq j$ for all x_{ij} and y_{ij}.

Now, the elimination order based formulation (EOF) reads:

$$\min w \tag{1}$$

subject to

$$w \geq \sum_{j \in V} y_{ij}, \ \forall i \in V, \tag{2}$$

$$x_{ij} + x_{ji} = 1, \ \forall \{i,j\} \subseteq V, \tag{3}$$

$$x_{ij} + x_{jk} - x_{ik} \leq 1, \ \forall \{i,j,k\} \subseteq V, \tag{4}$$

$$y_{ij} \leq x_{ij}, \ \forall \{i,j\} \subseteq V, \tag{5}$$

$$y_{ij} = x_{ij}, \ \forall ij \in E, \tag{6}$$

$$x_{jk} + y_{ij} + y_{ik} - y_{jk} \leq 2, \ \forall \{i,j,k\} \subseteq V, \tag{7}$$

$$x_{ij} \in \{0,1\}, y_{ij} \in \{0,1\}, \ \forall i,j \in V. \tag{8}$$

Constraints (3), (4) and (8) guarantee that variables x determine a linear order. Constraints (5) force the y_{ij} variable to be zero as soon as i is not ordered before j. Constraints (6) impose that edges of G are counted in the triangulation. Constraints (7) imply that if edges ij and ik are in the triangulation with i ordered before j and i ordered before k, then there must exist an edge jk in the triangulation.

First, let us estimate the integrality gap of EOF. Consider $m \times m$-grid graph which is a Cartesian product of two simple paths, $P_m \times P_m$ (Cartesian product is the direct product of two sets). The number of vertices in this graph is $n = m^2$ and $tw = m = \sqrt{n}$. It is easy to check that the symmetric solution

$$x_{ij} = \frac{1}{2}, \forall \{i,j\} \subseteq V,$$

$$y_{ij} = \begin{cases} \frac{1}{2}, \text{ if } ij \in E, \\ 0, \text{ otherwise.} \end{cases}$$

is feasible for the LP-relaxation where Constraints (8) are substituted just by non-negativity requirement. Given this solution w is maximal on the internal vertices of the grid which have degree 4, thus $LP \leq 4 * \frac{1}{2} = 2$ where LP is the optimal solution of the linear relaxation of EOF. Let OPT denotes the treewidth of the given graph. Then, the integrality gap of EOF is at least $\frac{OPT}{LP} \geq \frac{m}{2} = \frac{\sqrt{n}}{2}$.

3.1 Introduction to flow metrics

In this subsection we give a brief introduction to the flow metrics technique introduces by Bornstein and Vempala [7] where the flow metrics is the relaxation of the path metrics (i.e. linear ordering). A lot of problems in combinatorial optimization can be formulated as finding an assignment of pairwise distances of a specified type which minimizes a special cost function on the distances. The problems modeled in this way are often NP-hard and one approach to solve them is to consider relaxations of the associated metrics. The proposed method uses the same relaxation and essentially the same rounding procedure to solve the problem; only the objective function is varied.

Consider a path on n vertices numbered $1, 2, \ldots, n$. The distance between u and v in the corresponding path metrics is $|u - v|$.

Now imagine that we send a flow of one unit from each vertex u to every vertex v to the right of u ($v > u$). These flows satisfy the following properties:

- For each pair of vertices u and v, the value of the flow from u to v plus the value of the flow from v to u is equal to one.
- For every triplet u, v, w, the flow between u and v that goes through w, the flow between u and w that goes through v and the flow between v and w that goes through u sum to one.

Any metric that satisfies the above properties is called a flow metric. To define this formally, let the variables $f_{ij}^{u,v}$ represent the value of the flow from u to v on the edge ij. These variables satisfy flow conservation at every vertex except the source and the sink. We pay attention that the variables $f_{ij}^{u,v}$ are determined only for edges $ij \in E$. Define a set of auxiliary variables

$$g_w^{u,v} = \sum_{i:\, iw \in E} f_{iw}^{u,v}.$$

In other words, $g_w^{u,v}$ represents the flow from u to v that goes through the vertex w.

Definition 4. *A* flow metric *is a solution to the following linear program (FP), where f and g are flow variables as defined above.*

$$g_j^{i,j} + g_i^{j,i} = 1, \ \forall \{i,j\} \subseteq V, \tag{9}$$

$$(g_k^{i,j} + g_k^{j,i}) + (g_i^{j,k} + g_i^{k,j}) + (g_j^{i,k} + g_j^{k,i}) = 1, \ \forall \{i,j,k\} \subseteq V, \tag{10}$$

$$d_f(i,j) = \sum_{k \in V} g_k^{i,j}, \ \forall i,j \in V, \tag{11}$$

$$d_f(i,j) + d_f(j,k) \geq d_f(i,k), \ \forall i,j,k \in V. \tag{12}$$

Constraints (11) define a set of directed distances that form the "metric" and Constraints (12) impose the triangle inequality on these distances.

An important property of a flow metric is that it satisfies the one-dimensional spreading constraints as stated in the following theorem:

Theorem 3. *[7] For any subset $S \subseteq V$ of vertices,*

$$\sum_{u,v \in S} d_f(u,v) \geq \binom{|S|}{3}.$$

Roughly speaking, for a spreading metric the average distance in any subset of k vertices is about k, as in the case of a path. This property can be modeled using a set of linear constraints above. Although the number of constraints in exponential in the size of the input graph, they admit an efficient separation oracle [13] and so the resulting relaxations can be solved in polynomial time.

3.2 Improvement of EOF using flow metrics

Flow metric technique is a powerful tool of polishing the ordering polytope in the neighborhood of the optimal solution. We use its power to improve Elimination Order formulation for treewidth. The way to merge ILP and flow metrics technique is based on the comparison of the variables and identification of some of them. Consider two relaxations of linear ordering polytopes (3)–(4) and (9)–(10) we can straightforwardly combine these constraints into one single polytope in the following way:

Proposition 1.

$$x_{ij} = g_j^{i,j}.$$

Proof. We remind that x_{ij} and $g_j^{i,j}$ are defined as follows:

$$x_{ij} = \begin{cases} 1, \text{ if } i \text{ is ordered before } j \text{ in the perfect elimination scheme,} \\ 0, \text{ otherwise.} \end{cases}$$

$$g_j^{i,j} = \sum_{k:\, kj \in E} f_{kj}^{i,j} = \begin{cases} 1, \text{ if there is the flow from } i \text{ to } j, \\ 0, \text{ otherwise.} \end{cases} =$$

$$= \begin{cases} 1, \text{ if } i \text{ is before } j \text{ in the vertex linear ordering,} \\ 0, \text{ otherwise.} \end{cases}$$

Thus, both x_{ij} and $g_j^{i,j}$ are equal to one if i is placed before j in the linear ordering and both are equal to zero otherwise. We conclude that x_{ij} and $g_j^{i,j}$ are equal in the integral solution.

In fact, by Proposition 1 we notice that Equation (3) and Equation (9) are identical. We formulate the merged ILP as follows:

$$\min w \tag{13}$$

subject to

$$w \geq \sum_{j \in V} y_{ij}, \ \forall i \in V, \tag{14}$$

$$g_k^{i,j} = \sum_{m \in V:\, mk \in E} f_{mk}^{i,j}, \ \forall i,j,k \in V, \tag{15}$$

$$g_j^{i,j} + g_i^{j,i} = 1, \ \forall \{i,j\} \subseteq V, \tag{16}$$

$$g_j^{i,j} + g_k^{j,k} - g_k^{i,k} \leq 1, \ \forall \{i,j,k\} \subseteq V, \tag{17}$$

$$y_{ij} \leq g_j^{i,j}, \ \forall \{i,j\} \subseteq V, \tag{18}$$

$$y_{ij} = g_j^{i,j}, \ \forall ij \in E, \tag{19}$$

$$g_k^{j,k} + y_{ij} + y_{ik} - y_{jk} \leq 2, \ \forall \{i,j,k\} \subseteq V, \tag{20}$$

$$(g_k^{i,j} + g_k^{j,i}) + (g_j^{i,k} + g_j^{k,i}) + (g_i^{j,k} + g_i^{k,j}) = 1, \ \forall \{i,j,k\} \subseteq V, \tag{21}$$

$$d_f(i,j) = \sum_{k \in V} g_k^{i,j}, \ \forall i,j \in V, \tag{22}$$

$$d_f(i,j) + d_f(j,k) \geq d_f(i,k), \ \forall i,j,k \in V, \tag{23}$$

$$g_k^{i,j} \in \{0,1\}, y_{ij} \in \{0,1\}. \tag{24}$$

We make the following conjecture about the presented ILP:

Conjecture 1. Applying rounding algorithm by Bornstein and Vempala [7] to the linear relaxation of ILP (13)–(24), one derives an $O(\log n)$-approximate solution to the treewidth problem in polynomial time.

To show that the proposed method is promising and the conjecture is plausible, we present the following $O(\log^3 n)$-approximation for the treewidth. Consider the pathwidth problem, where condition on the tree T in Definition 1 is strengthened to the condition that T is a simple path. The pathwidth problem is formulated as an ILP on the polytope contained in (14)–(24), see Bornstein and Vempala [7]. They proposed the rounding algorithm which provides a solution to the pathwidth

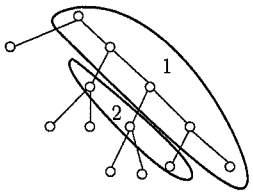

Fig. 1. The construction is symmetric with respect to the root. We choose the longest path from the root to the bottom and merge it into one bag 1 of the path-decomposition. Bag 2 is connected to bag 1, and so on.

of value $APX_{pw} \leq pw \times O(\log^2 n)$. Clearly, $tw \leq pw$. It is well known that $pw \leq tw \times O(\log n)$. The argument is as follows: without loss of generality we consider the binary tree-decomposition on n vertices. The depth of this tree is at most $O(\log n)$. The way to pack the tree-decomposition into the path-decomposition is demonstrated on Figure 1. Every new bag of the path-decomposition is at most $tw \times O(\log n)$.

Therefore, if we simply take the derived value of pathwidth as a feasible approximate solution to the treewidth we have than

$$tw \leq pw \leq APX_{pw} = APX_{tw} \leq pw \times O(\log^2 n) \leq tw \times O(\log^3 n).$$

Thus, already the direct usage of the rounding algorithm for the pathwidth guarantees the polylogarithmic approximation for the treewidth. Conjecture 1 stays open as a question for the further research. Notice, this polylogarithmic approximation immediately implies that we improved upon (1)–(8) by adding the flow metrics constraints. This is because the symmetric solution enforces the integrality gap be at least $\Omega(\sqrt{n})$ are not longer feasible as the new linear relaxation values are different from the optimum by at most a polylogarithmic factor.

4 Tree drawing formulation

In this section we introduce brand-new ILP formulation for the treewidth problem purely based on geometry of tree-decomposition. The problem is formulated as a network design problem on a grid and the underlying integer linear program formulation describes the geometric properties of the optimal tree decomposition.

It is known that for any graph $G = (V, E)$ on n vertices there is a binary tree on $O(n)$ vertices representing the optimal tree decomposition of G. Without loss of generality, assume that such binary optimal tree decomposition of G on n vertices is given. By Crescenzi et al. [8] any binary tree on n vertices can be properly embedded in the $n \times (\log n + 1)$ grid in the right-down fashion, thus the optimal binary tree decomposition also has such drawing. Hence, we make all further drawings on the $n \times (\log n + 1)$ grid (N, A) with the node set N and the edge set A. We assume that the horizontal grid layers are numbered $\{0, \ldots, n-1\}$ in the top-to-bottom manner and the vertical layers are numbered $\{0, \ldots, \log n\}$ in the left-to-right manner. Let $y = (i, j)$ be the grid node lying on the horizontal layer i and vertical layer j. We denote y_r, y_u, y_ℓ and y_b the right, upper, left and bottom neighbors of y respectively.

Definition 5. *For a pair of grid nodes $y = (i, j)$ and $y' = (i', j')$, we write $y \preceq y'$ if $i \leq i'$ and $j \leq j'$. Consistently, we write $y \prec y'$ if $y \preceq y'$ and $y \neq y'$.*

For every vertex $v \in V$ we construct a connected tree T_v which is a subtree of the optimal tree decomposition where every node contains vertex v. T_v is embedded on the edges of the grid such

that there is no node in the tree which is adjacent simultaneously to its upper and left neighbors. We call such a tree drawing the *proper right-down drawing*. Notice that in any tree drawn in proper right-down way, there is always a unique left upper node of the tree.

By constraints of the integer linear program below we guarantee that the combined drawing of all vertex subtrees does not contain any cycles, i.e. it is a tree, and for any two adjacent vertices in G the corresponding vertex subtrees are overlapping, i.e. there is a bag of the optimal tree decomposition which contains both vertices.

To model the proper right-down drawing of the vertex subtrees, for every $v \in V$, $e \in A$ and $y \in N$ we introduce the following binary variables:

$$s_{vy} = \begin{cases} 1, \text{ if } y \text{ is the unique left upper corner of } T_v, \\ 0, \text{ otherwise.} \end{cases}$$

$$x_{ve} = \begin{cases} 1, \text{ if } e \in E(T_v), \\ 0, \text{ otherwise.} \end{cases}$$

$$u_{vy} = \begin{cases} 1, \text{ if } y \in V(T_v), \\ 0, \text{ otherwise.} \end{cases}$$

Constraints which model the problem are the following. Existence of a unique left upper corner of the subtree can be straightforwardly described by the equation

$$\sum_{y \in N} s_{vy} = 1, \ \forall v \in V. \tag{25}$$

Next constraints ensure that in the drawing there is no node y simultaneously adjacent to its left and upper neighbors:

$$x_{v(y_\ell,y)} + x_{u(y,y_u)} \leq 1, \ \forall v, u \in V, y \in N. \tag{26}$$

Connectivity of the subtrees we guarantee by the constraints

$$u_{vy} = s_{vy} + x_{v(y_\ell,y)} + x_{v(y,y_u)}, \ \forall v \in V, y \in N, \tag{27}$$

$$x_{v(y,y_r)} \leq u_{vy}, \ \forall v \in V, y \in N, \tag{28}$$

$$x_{v(y_b,y)} \leq u_{vy}, \ \forall v \in V, y \in N. \tag{29}$$

Now, when the subtrees are modeled, we set up the inequalities which ensure that for any two adjacent vertices in G the corresponding two subtrees are overlapping. To do this, we notice that in any proper right-down drawing two subtrees are overlapping if and only if the upper-left corner of one of the subtrees belongs to another subtree. Thus, we complete the construction of the treewidth polytope by setting the following inequalities:

$$s_{uy} + \sum_{y' \preceq y} s_{vy'} \leq 1 + u_{vy}, \ \forall (v,u) \in E, y \in N, \tag{30}$$

$$s_{uy} + \sum_{y' \in N : y' \not\preceq y, y \not\preceq y'} s_{vy'} \leq 1, \ \forall (v,u) \in E, y \in N. \tag{31}$$

The integrality constrains are:

$$s_{vy}, \ x_{ve}, \ u_{vy} \ \in \{0,1\}. \tag{32}$$

To put the problem in an optimization framework, we add one more variable w for the treewidth of G and require

$$w + 1 \geq \sum_{v \in V} u_{vy}, \ \forall y \in N. \tag{33}$$

The objective function is given straightforwardly:

$$\min w. \tag{34}$$

We refer to the tree drawing ILP formulation as TDF. There is the following relation between the solution of TDF and the treewidth defining vertex elimination order. We find the partial order of vertices using the following proposition and linearize the partial order to obtain an elimination scheme:

Proposition 2. *Let π be the partial order of vertices in G such that $\pi(v) \leq \pi(u)$ if $y \prec y'$ for $s_{vy} = 1$ and $s_{uy'} = 1$ in the optimal solution to (25)-(34). Then π is a treewidth defining vertex elimination order for graph G.*

4.1 Introduction to local branching

In this subsection we give an overview of the local branching technique introduced by Fischetti and Lodi [11]. The procedure is in the spirit of well-known local search metaheuristics, but the neighborhoods are obtained through the introduction into the Mixed Integer Programming (MIP) model the set of completely general linear inequalities, called *local branching cuts*. This allows the use of a general-purpose MIP solver as a black-box "tactical" tool to explore effectively suitable solution subspaces defined and controlled at a "strategic" level by a simple external branching framework.

We consider a generic MIP of the form:

$$\min c^T x \tag{35}$$

subject to

$$Ax \geq b, \tag{36}$$

$$x_j \in \{0,1\}, \ \forall j \in \mathcal{B} \neq \emptyset, \tag{37}$$

$$x_j \geq 0, x_j \in \mathbf{Z}, \ \forall j \in \mathcal{G}, \tag{38}$$

$$x_j \geq 0, \ \forall j \in \mathcal{C}. \tag{39}$$

Here, the variable index set $\mathcal{N}$ is partitioned into $(\mathcal{B}, \mathcal{G}, \mathcal{C})$, where $\mathcal{B} \neq \emptyset$ is the index set of the binary variables, while the possibly empty sets $\mathcal{G}$ and $\mathcal{C}$ index the general integer and the continuous variables, respectively. Given a feasible reference solution $\bar{x}$ of the problem, let $\bar{S} = \{j \in \mathcal{B} \mid \bar{x}_j = 1\}$ denote the binary support of $\bar{x}$. For a given positive integer parameter k, we define the k-OPT neighborhood $N(\bar{x}, k)$ of $\bar{x}$ as the set of the feasible solutions satisfying the additional local branching constraint:

$$\Delta(x, \bar{x}) = \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \bar{S}} x_j \leq k, \tag{40}$$

where the two terms in left-hand side count the number of binary variables flipping their value (with respect to $\bar{x}$) either from 1 to 0 or from 0 to 1, respectively.

The local branching constraint can be used as a branching criterion within an enumerative scheme for MIP: given a solution $\bar{x}$, the solution space associated with the current branching node can be partitioned by means of the disjunction:

$$\Delta(x, \bar{x}) \leq k \text{ left branch } \mathbf{or} \ \Delta(x, \bar{x}) \geq k + 1 \text{ right branch.} \tag{41}$$

The idea is that the neighborhood $N(\bar{x}, k)$ corresponding to the left branch must be "sufficiently small" to be optimized within short computing time, but still "large enough" to likely contain better solutions than $\bar{x}$. According to the computational results [11], the choice of k in range [10, 20] was effective in most cases.

4.2 Local branching technique for TDF

In this subsection we apply the local branching technique [11] to TDF for the treewidth.

The intuition for the local branching method applied to the treewidth is as follows. We start with some (not optimal) tree decomposition T_0 which is a combination of the vertex subtrees T_v in the grid. We define the neighborhood of T_0 as the set of tree decompositions where at most k vertex subtrees T_v might have the left-upper corners different from their locations in T_0. By a MIP solver or by any other suitable method, we find in the constructed neighborhood a tree decomposition T_1 having the minimal treewidth. Making clear the connection to the vertex elimination orders announced in Proposition 2, this step of the local branching corresponds to finding the (locally) optimal tree decomposition obtained by shifting at most k vertices in a given vertex elimination order. Now, given T_1, we define its neighborhood by the set of tree decompositions where at least $k + 1$ subtrees are different from the subtrees in T_0 and at most k subtrees are different from the subtrees in T_1. This can be done by adding two inequalities to the integer linear program. To the extended ILP, we again find a tree decomposition T_2 having the minimal treewidth. We recurse on the integer linear programs, i.e. at step i we obtain a local optimum T_i adding to the current ILP

- inequalities determining a small neighborhood of T_{i-1},
- inequalities cutting off the neighborhood of T_{i-2}.

We stop the process when the extended ILP does not have any solution. Since the algorithm starts with the whole set of feasible solutions and proceeds till the empty set, the best found solution is a provable global optimum. Below we give a detailed description of the algorithm.

Given a feasible integer *reference solution*(s^*, x^*, u^*, w^*) to (25)–(34), let $S^* := \{(v, y) : s^*_{vy} = 1\}$ denote the binary support of the solution. For a given positive integer parameter k, we define the k-optimal neighborhood $\mathcal{N}(s^*, x^*, u^*, w^*, k)$ of (s^*, x^*, u^*, w^*) as the set of feasible solutions to the integer program (25)–(34) satisfying the additional *local branching constraint*

$$\Delta(s, s^*) := \sum_{(v,y) \in S^*} (1 - s_{vy}) + \sum_{(v,y) \notin S^*} s_{vy} \leq k, \tag{42}$$

where the two terms in the left-hand side count the number of binary variables s flipping their value with respect to s^*.

To prune the algorithm, preprocessing rules reducing the size of the graph and constructing safe separators must be applied, see [5, 6]. Clearly, the stoppage criteria of the algorithm can be easily extended: if at some step of the algorithm the optimal solution of the linear relaxation to the current ILP is greater than the width of the best found tree decomposition then we can stop since further adding inequalities will only increase the width of the decompositions.

Algorithm 1.2: LocalBranching

Generate any feasible $(s_0^*, x_0^*, u_0^*, w_0^*)$.

Generate $(s_1^*, x_1^*, u_1^*, w_1^*)$ by adding the constraint $\Delta(s, s_0^*) \leq k$ to (14)–(24) and solving the resulting ILP by a MIP-solver.

Let $i := 2$.

While The set of feasible solutions of the current ILP is not empty **Do**
  In the current ILP, replace $\Delta(s, s_{i-2}^*) \leq k$ by $\Delta(s, s_{i-2}^*) \geq k+1$
    and add $\Delta(s, s_{i-1}^*) \leq k$.
  Solve the ILP by MIP to generate $(s_i^*, x_i^*, u_i^*, w_i^*)$.
  Set $i := i + 1$.
End While

Output the solution (s^*, x^*, u^*, w^*) found during the previous steps with the minimum value w^*.

5 Comparison of formulations and methods

We start the comparison of ILP formulations EOF and TDF with the program sizes. Given graph G on n vertices TDF has $O(n^2 \log n)$ variables while EOF needs $O(n^4)$ variables. The situation is opposite with respect to the number of constraints: TDF contains $O(n^3 \log n)$ constraints, EOF has $O(n^3)$ constraints. Hence, size-wise TDF is slightly more compact than EOF.

It is noticeable that despite its generality, the local branching technique of Fischetti and Lodi [11] is hardly applicable to the linear ordering polytopes in general and to EOF in particular. This is due to the fact that all variables in the linear ordering polytopes are related to each other, e.g. shift of one element in the linear order implies changes in $n/2$ variables on average. Moreover, the choice on which variable to branch is also very difficult. This was one of the basic reasons to introduce TDF with variables allowing more freedom for local changes and which are more suitable for the local branching procedure. In the local branching, if k is not very large and only deviations of k elements in the current integer solution are allowed, the spread of the fractional values is not very high and the local optimum can be found quickly.

On the other hand, EOF can be nicely placed within the flow metrics framework as it already assumes some "path-like" structure which is the basis of the flow approach. The "flow metrics" constraints polish the linear ordering polytope in the neighborhood of the optimal solution which provides better approximation guarantees and speeds up the search. At the same time it is difficult to find a straightforward way to integrate the flow metrics approach into TDF as it is based on geometrical and structural properties of the tree-decomposition more than on ordering properties.

It is known that linear ordering formulations, like EOF, work quite well on dense graphs as finding a clique leads to a good progress of the objective function bounds; but are typically bad on sparse graphs. The situation is somewhat opposite for TDF. It can get stuck in some node of the branching tree if input graph is a clique. Furthermore, the arbitrary spread of k trees T_v which are allowed to be changed is very bad for the input clique graph as the optimal tree-decomposition is just one bag. But this is also the reason for the more effective search on the sparse graphs as we allow the big "jumps" and changes in one step of the algorithm.

6 Conclusions and open questions

We considered two Integer Linear Programming formulations for the treewidth. The first one is based on the vertex elimination order. It is merged with the flow metric approach by Bornstein

and Vempala [7] to obtain the approximation bounds. We show that using this formulation one can obtain straightforwardly a polylogarithmic approximation for the treewidth. The second ILP is structural and it is based on the drawing of the optimal tree decomposition on the grid. This representation leads to some nice geometric properties which we exploit in a local branching procedure proposed by Fischetti and Lodi [11] to obtain an exact algorithm for the Minimum Treewidth problem. The computational experiments on different graph classes can bring more insights about the weaknesses and strengths of the proposed formulations. This is left for the further research.

References

1. E. Amir: *Efficient Approximation for Triangulation of Minimum Treewidth.* In Proc. of the 17th Conference in Uncertainty in Artificial Intelligence (UAI01), (2001), pp. 7–15.
2. H.L. Bodlaender: *A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth.* SIAM J. Comput., **25**(6) (1996), pp. 1305–1317.
3. H.L. Bodlaender: *Discovering Treewidth.* In Proc. of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2005), Lecture Notes in Computer Science, **3381**, Springer (2005), pp. 1–16.
4. H.L. Bodlaender, Gilbert J.R., Hafsteinsson H. and Kloks T.: *Approximating Treewidth, Pathwidth, Frontsize, and Shortest Elimination Tree.* J. Algorithms, **18**(2) (1995), pp. 238–255.
5. H.L. Bodlaender and Koster A.M.C.A.: *Safe Separators for Treewidth.* Discrete Mathematics, **306**(3) (2006), pp. 337–350.
6. H.L. Bodlaender, Koster A.M.C.A. and Eijkhof F. van den: *Pre-processing Rules for Triangulation of Probabilistic Networks.* Computational Intelligence, **21** (2005), pp. 286-305.
7. C.F. Bornstein and Vempala S.: *Flow Metrics.* Theor. Comput. Sci., **321**(1) (2004), pp. 13–24.
8. P. Crescenzi, Battista G. D. and Piperno A.: *A Note on Optimal Area Algorithms for Upward Drawings of Binary Trees.* Comput. Geom., **2** (1992), pp. 187–200.
9. U. Feige, Hajiaghayi M. and Lee J.R.: *Improved Approximation Algorithms for Minimum Weight Vertex Separators.* SIAM J.Comput., **38**(2) (2008), pp. 629–657.
10. U. Feige and Lee J.R.: *An Improved Approximation Ratio for the Minimum Linear Arrangement Problem.* Inf. Process. Lett., **101**(1) (2007), pp. 26–29
11. M. Fischetti, and Lodi A.: *Local Branching.* Math. Program., **98**(1-3) (2003), pp. 23–47.
12. F.V. Fomin, Kratsch D, Todinca I., Villanger Y.: *Exact Algorithms for Treewidth and Minimum Fill-In.* SIAM J. Comput., **38**(3) (2008), pp. 1058–1079.
13. M. Grötschel, Lovász L. and Schrijver A.: *Geometric Algorithms and Combinatorial Optimization.* Ser. Algorithms and Combinatorics, **2**, Springer (1988).
14. A.M.C.A. Koster and Bodlaender H.L.: *Private communication.*
15. J. Palsberg and Naik M.:*ILP-based Resource-aware Compilation.* In A. Jerraya and Wolf W. (Eds) *Multiprocessor Systems-on-Chips*, Elsevier, (2004).
16. G. Pokam and Bodin F.: *Energy-Delay Tradeoff Analysis of ILP-based Compilation Techniques on a VLIW Architecture.* INRIA Research Report, **RR-5026**, (2003).
17. S. Rao, and Richa A.W.: *New Approximation Techniques for Some Linear Ordering Problems.* SIAM J. Comput., **34**(2) (2004), pp. 388-404.
18. N. Robertson and Seymour P.D.: *Graph Minors. III. Planar Tree-width.* J. Comb. Theory, Ser. B, **36**(1) (1984), pp. 49–64.