Tractable Parameterizations for the Minimum Linear Arrangement Problem

Michael R. Fellows[1], Danny Hermelin[2],
Frances Rosamond[1], and Hadas Shachnai[3]

[1] School of Engineering and IT, Charles Darwin University, Darwin, Australia.
Email: {michael.fellows,frances.roasmond}@cdu.edu.au
[2] Department of Industrial Engineering and Management, Ben-Gurion University, Beer-Sheva, Israel. Email: hermelin@bgu.ac.il
[3] Computer Science Department, Technion, Haifa 32000, Israel.
Email: hadas@cs.technion.ac.il

Abstract. The Minimum Linear Arrangement (MLA) problem asks to embed a given graph on the integer line so that the sum of the edge lengths of the embedded graph is minimized. Most layout problems are either intractable, or not known to be tractable, parameterized by the *treewidth* of the input graphs. We investigate MLA with respect to three parameters that provide more structure than treewidth. In particular, we give a factor $(1+\varepsilon)$-approximation algorithm for MLA parameterized by (ε, k), where k is the *vertex cover number* of the input graph. By a similar approach, we describe two FPT algorithms that exactly solve MLA parameterized by, respectively, the *max-leaf* and *clique-cover* numbers of the input graph.

1 Introduction

Given a graph $G = (V, E)$, a *linear arrangement* is a linear ordering on the set of vertices V of G which is specified by a permutation $\pi : V \to \{1, 2, \dots, |V|\}$. The *cost* of the arrangement is defined by $cost(\pi) = \sum_{(u,v) \in E} |\pi(u) - \pi(v)|$; that is, the cost of the ordering is the sum total of the edge lengths under the ordering. In typical applications one is interested in linear arrangements of low cost. The Minimum Linear Arrangement (MLA) problem is the problem of finding a linear arrangement of minimum cost, and the *standard parameterization* of the problem is to determine if an input graph G has a layout of cost at most k (the parameter).

MLA is one of the most important and well studied graph ordering problems. It is in some sense closely related to the Bandwidth problem, which seeks to minimize the maximum edge length of an ordering of the vertices. It has various applications, most of which stem from the domain of VLSI circuit design. As it was known to be NP-complete already from the mid 70's [11], most of the early work on this problem focused on designing heuristics and approximation algorithms. For a graph with n vertices, the best known approximation ratio for MLA is $O(\sqrt{\log n} \log \log n)$, due to Feige and Lee [5], and Charikar, Hajiaghayi, Karloff and Rao [4]. Earlier notable work include the paper by Rao and Richa [17] who presented an $O(\log n)$-approximation algorithm for the problem, along with another algorithm for planar graphs that achieves a ratio of $O(\log \log n)$. Recently, Ambühl, Mastrolilli, and Svensson [1] showed that MLA does not admit a PTAS unless NP-complete problems can be solved in randomized subexponential time. We refer the reader to [17] for a further account of earlier work on MLA.

In terms of parameterized complexity, the standard parameterization of MLA is trivially *fixed-parameter tractable (FPT)*. Gutin *et al.* presented in [13] an FPT algorithm which, given a graph on n vertices and m edges, and a parameter k, outputs a linear arrangement of cost at most $m + k$, if one exists. Fernau in [10] described efficient bounded search tree FPT algorithms for MLA under its standard parameterization. Bodlaender *et al.* [3] investigated exact exponential-time algorithms for several ordering problems, including MLA.

Most parameterized problems are FPT parameterized by the treewidth of the input graph. However, graph layout and width problems are a notable exception (but see also [6] for further examples of parameterized problems that are $W[1]$-hard when parameterized by the input treewidth, or even the input vertex cover number). Parameterized by the treewidth of the input graph, Bandwidth is known to be hard for $W[t]$ for all t (this follows from the results in [2]). Whether something similar holds for MLA is unknown. This general situation motivates studying the complexity of these problems, parameterized by structural parameters even stronger than treewidth, a program that is sometimes called *parameter ecology*. See [7] for a recent survey of this area.

In this paper, we consider the complexity of MLA parameterized by three structural parameters that have a certain commonality: all three, when bounded, force the input graph G to have a structure, that essentially consists of an elaboration of some small (parametrically bounded) "seed" graph H, that gives us sufficient information about the entire graph G to be able to derive efficient algorithms. The three graph structural parameters we consider are:

- (i) The *vertex cover number* of G, denoted $vc(G)$, which is the size of the smallest set of vertices intersecting every edge in G.
- (ii) The *maximum leaf number* of G, denoted $m\ell(G)$, which is the maximum number of leaves in any spanning tree of G.
- (iii) The *edge clique cover number* of G, denoted $ecc(G)$, which is defined to be the minimum number of clique required to cover all the edges of G.

The question of whether MLA is FPT by the vertex cover number of the input graph has been prominently raised in [9], where it is shown that a number of graph layout and width problems such as Bandwidth, Cutwidth and Distortion are FPT by this parameter, and this question has been a noted open problem in parameterized algorithmics. Here we offer a partial positive answer *by taking an approach that combines parameterization and approximation.* One of our main results shows that MLA can be approximated to within a factor of $(1+\varepsilon)$ of optimal, in FPT time for the aggregate parameter $(1/\varepsilon, k)$, where k is the vertex cover number of the input graph. (Whether the MLA problem can be exactly solved in FPT time for the parameter k alone still remains open.) In [8] it is shown that BANDWIDTH is FPT, parameterized by the *max leaf number* of the input graph. Here we obtain a matching result for MLA; it can be exactly solved in FPT time for this parameter. While the techniques used in both results look similar from a bird's eye, there are several major differences hiding in the details. Finally, our last result shows that the *edge clique cover number* can potentially be a useful parameterization for other graph layout problems.

The paper is organized as follows. In Section 2 we give a $(1+\varepsilon)$-approximation for MLA parameterized by $vc(G)$. In Sections 3 and 4 we present FPT algorithms for MLA using the parameters $m\ell(G)$ and $ecc(G)$, respectively. We conclude in Section 5 with some open problems. Due to space constraints some of the proofs are given in the Appendix.

2 MLA parameterized by Vertex-Cover Number

In this section we present an algorithm which yields a $(1+\varepsilon)$-approximation for MLA in FPT-time with respect to $k = vc(G)$ and $1/\varepsilon$, where G is the input graph and $\varepsilon > 0$. Our algorithm proceeds as follows.

W.l.o.g., we assume that G has no isolated vertices, and let $m = |E|$. The first step of our algorithm is to compute a vertex cover $V' \subseteq V$ of G of size k. Note that each vertex in $V \setminus V'$ has neighbors only in V'. We say that two vertices $v_1, v_2 \in V \setminus V'$ are of the same *type* if $N(v_1) = N(v_2)$, where $N(u)$ denotes the set of neighbors of a vertex u in G. Clearly there are $T \leq 2^k$ different types of vertices in $V \setminus V'$. We let n_t denote the number of vertices of type t, $1 \leq t \leq T$. The main idea of our algorithm is as follows. We put together vertices of the same type into groups of an appropriately chosen size, and then compute an optimal linear arrangement for the graph obtained by merging each group into a single *mega-vertex*. The analysis of our algorithm relies on an interesting homogeneity lemma which relates to the behavior of vertices of identical type inside "gaps" formed by the vertices of V' in an optimal arrangement for G. A detailed description of the algorithm is given below.

2.1 Analysis

We now prove that the algorithm yields a solution that is within factor $(1+\varepsilon)$ from the optimal.

Theorem 1. *Algorithm 1 computes a $(1+\varepsilon)$-approximate linear arrangement for G in FPT time with respect to k and $1/\varepsilon$.*

We use in the proof of the theorem a few lemmas. Given a layout π for G, let $u_1, \ldots, u_k$ denote the vertices in V' as ordered in π. We say that a vertex $v \in V \setminus V'$ is in *gap i*, $1 \leq i \leq k-1$, if $\pi(u_i) < \pi(v) < \pi(u_{i+1})$. Similarly, v is in *gap 0* if $\pi(v) < \pi(u_1)$, and in *gap k* if $\pi(u_k) < \pi(v)$. The following lemma shows that the vertices of $V' \setminus V$ appear homogenously according to their type in some optimal linear arrangement of G.

Algorithm 1 MLA parameterized by $vc(G)$

Input: (G, ε, k).
Output: A linear ordering $\pi = (\pi(1), \ldots, \pi(n))$ for the vertices in G.

1: Set $s = \frac{\varepsilon m}{4k^2(k+1)2^k}$.
2: Apply an FPT algorithm to find a minimum vertex cover $V' \subseteq V$ of G. If $|V'| > k$ then STOP.
3: Partition the vertices in $V \setminus V'$ into T types.
4: **for all** $1 \le t \le T$ **do**
5:     Partition the vertices of type t arbitrarily to groups (*mega-vertices*), where each group is of size s (except, maybe, for the last group).
6:     Set the neighbors of each mega-vertex to be the neighbors (in V') of a vertex of type t.
7: Let $G' = (V_M \cup V', E_M)$ be the graph formed by the mega-vertices, where V_M is the set of mega-vertices and E_M is the set of edges connecting V_M and V'.
8: **for all** linear arrangements π' of G' **do**
9:     Lift π' to a linear arrangement π of G in the obvious manner, and calculate the resulting cost of π.
10: **return** the layout π found in Step 9 yielding the minimum cost.

Lemma 1 (Homogeneity for $vc(G)$). *There exists an optimal solution in which the vertices in each type appear in gap i consecutively, for all $1 \le i \le k-1$.*

Proof. Let $\pi : V \to \{1, \ldots, n\}$ denote an optimal linear arrangement of G, and let $\delta(v)$ denote the *force* of v (with respect to π) defined by

$$\delta(v) = |\{u \in N(v) |\ \pi(u) > \pi(v)\}| - |\{u \in N(v) |\ \pi(u) < \pi(v)\}|.$$

We note that for any gap i, $0 \le i \le k$, the placing the vertices in I from left to right in non-decreasing order by $\delta(v)$ is optimal. Indeed, if there exists a pair of vertices $u, v \in I$ which are adjacent according to π in gap i with $\delta(v) > \delta(u)$ and $\pi(v) < \pi(u)$, we can swap v and u and obtain a linear arrangement of smaller cost. It follows that all of the vertices v with equal force in gap i will be placed consecutively. We can thus place all of the vertices of type t as a contiguous block in gap i without harming the optimality of the arrangement, since all of these vertices have the same force in gap i. $\square$

Lemma 2. *The cost of any linear arrangement for G is at least $\frac{m^2}{4k^2}$.*

Proof. It is not difficult to see that a graph G with $vc(G) = k$ and m edges attaining the minimum possible cost of a linear arrangement is the disjoint union of k stars. Consider such a graph G^*, and let ℓ_r denote the number of vertices in the r-th star of G^*, $1 \le r \le k$. Then the cost of a minimum linear arrangement of G^* is lower-bounded by

$$MLA(G^*) \ge \sum_{r=1}^{k} \sum_{d=1}^{\lfloor \ell_r/2 \rfloor} 2d \ge \sum_{r=1}^{k} \frac{\ell_r^2}{4}.$$

The function $\sum_{r=1}^{k} \frac{\ell^2}{4}$ above is minimized when $\ell_r = \frac{m}{k}$ for all $1 \le r \le k$. Thus, $\text{MLA}(G^*) \ge \frac{m^2}{4k^2}$, and the lemma follows. $\square$

Lemma 3. *Let MLA(G) denote the cost of the optimal arrangement for G, and let MLA(G′) denote the cost of the layout returned by Algorithm 1. Then* $MLA(G') \le MLA(G) + \frac{\varepsilon m^2}{4k^2}$.

Proof of Theorem 1: Observe that the running time of Algorithm 1 can roughly be bounded by the number of linear arrangements of G'. Note that $|E'| = \lceil m/s \rceil = O(2^k k^3)$ and $|V'| \le 2|E'|$, as G' has no isolated vertices. Thus, the total running time of the algorithm can be bounded by $O^*((2^k k^3)!)$. Furthermore, by Lemmas 2 and 3, we have

$$\frac{MLA(G')}{MLA(G)} \le 1 + \frac{\varepsilon m^2}{4k^2 \cdot MLA(G)} = 1 + \varepsilon,$$

where $MLA(G')$ denotes the minimum cost of the layout returned by the algorithm, and $MLA(G)$ denotes the optimal cost. Thus, Algorithm 1 returns a $(1+\varepsilon)$-approximate solution in FPT time with respect to (ε, k). $\square$

3 MLA parameterized by Max-Leaf Number

In this section we give an FPT algorithm for MLA parameterized by $k = m\ell(G)$, for $k > 1$. We start with some definitions. Given a graph $H = (V', E')$, a *subdivision* of an edge $e' = (u, v) \in E'$ replaces e' by a path of length $(n_{e'} + 2)$, for some $n_{e'} \ge 1$, given by $(u, v_1, \ldots, v_{n_{e'}}, v)$. Thus, the edge e' becomes an *edge-path*, and we add to H' $n_{e'}$ vertices. We say that a graph G is *a subdivision of a graph H* if G was generated by subdivision operations on the edges of H.

As shown in [15], if $m\ell(G) = k$ then G is a subdivision of a graph H on at most $4k-2$ vertices. We call $H = (V', E')$ where $V' \subseteq V$ and $|V'| = k' \le 4k-2$ the *seed* graph of G. Let $S \subseteq E'$ denote the set of subdivided edges in H. Then, $V = V' \cup (\cup_{e' \in S}\{v_1, \ldots, v_{n_{e'}}\})$, and $E = E' \setminus S \cup \{(z, w)|\ z \in V \setminus V', w \in V\}$. We say that a vertex $w \in V \setminus V'$ belongs to e', where $e' = (u, v) \in E'$, if w is on the path $(u, v_1, \ldots v_{n'_e}, v)$ in G.

Our algorithm for MLA, parameterized by $m\ell(G)$, branches on an exhaustive set of solution patterns which has size bounded by a function of k. This set of patterns contains at least one optimal solution for the graph G, if one exists. Then, to determine when a solution pattern can be realized, we solve a linear integer program which outputs placements for all vertices of G in a permutation that is consistent with the given pattern. A detailed description is given in Algorithm 2.

We define a permutation π of the vertices in G, by extending a permutation σ of the vertices in H. Given such a permutation σ, we now formulate an integer linear program to find the position of vertices in $V \setminus V'$ among the vertices of H. Recall that any permutation $\sigma = (v_{j_1}, \ldots, v_{j_{k'}})$ of V' defines $(k'+1)$ gaps. Let π be a permutation of V that is consistent with σ. Let $\pi(v) \in [n]$ denote the

Algorithm 2 MLA parametrized by $m\ell(G)$

Input: (G, k).
Output: A linear ordering $\pi = (\pi(1), \ldots, \pi(n))$ for the vertices in G.

Let $H = (V', E')$ be the seed graph of G, where $E' = \{e_1, \ldots, e_{m'}\}$
for all permutations $\sigma \in S_{k'}$ of the vertices in V' **do**
    for all configurations of E', $\bar{C} = (\bar{C}_{e_1}, \ldots, \bar{C}_{e_{m'}})$ **do**
        Solve an integer linear program to determine the position of vertices in $V \setminus V'$ among vertices in V', such that the total cost is minimized (see details below).

return a permutation σ of V' which yields minimum cost for G.

position of vertex $v \in V$ in π, where $n = |V|$. We say that a vertex $w \in V \setminus V'$ is placed in gap i in π , $1 \leq i \leq k' - 1$, if $\pi(v_{j_i}) < \pi(w) < \pi(v_{j_{i+1}})$, where $v_{j_i}, v_{j_{i+1}} \in V'$; w is placed in gap 0 (k') if $\pi(w) < \pi(v_{j_1})$ ($\pi(w) > \pi(v_{j_{k'}})$).

A *configuration for an edge-path* of $e' \in E'$ is a $(k' + 1)$-vector, $\bar{C}_{e'} = (c_{e',0}, \ldots, c_{e',k'})$, in which the i-th entry is equal to '1' if gap i contains a vertex in e', and '0' otherwise. Let $|E'| = m'$. Then, a *configuration for E'* is a set of m' configuration vectors for all edge-paths in E'. We use in the integer program the variables $x_{e',i}$ to indicate the number of vertices on the edge-path of $e' \in E'$ in the i-th gap, $0 \leq i \leq k'$. Let $\sigma \in S_{k'}$ be a permutation of the vertices in V'. We denote by $Cost(G|\sigma, \bar{x})$ the total cost of the linear arrangement of the vertices in G, with the given permutation σ and the variable values. This cost can be computed in FPT time, using Lemma 4 (see Section 3.1). Then our program can be formulated as follows.

$$
\begin{array}{llll}
(LP_{m\ell}): & \text{minimize} & Cost(G|\ \sigma, \bar{x}) & \\
& \text{subject to:} & \displaystyle\sum_{i=0}^{k'} x_{e',i} = n_{e'} & \forall\ e' \in E' \\
& & x_{e',i} = 0 & \text{if } \ c_{e',i} = 0 \\
& & x_{e',i} \in \{1, \ldots, n_{e'}\} & \text{if } \ c_{e',i} = 1
\end{array}
$$

3.1 Analysis

We now prove that Algorithm 2 yields an optimal solution. Our analysis crucially relies on the next two lemmas.

Lemma 4 (Homogeneity for $m\ell(G)$). *There exists an optimal layout in which vertices in $V \setminus V'$ that belong to the same edge-path appear consecutively in gap i, for all $0 \leq i \leq k'$.*

Proof. Given a permutation π of the vertices in G, we show below that if the ordering of vertices in gap i does not satisfy the contiguity property in the lemma then we can modify the order of vertices to satisfy the property, without increasing the total cost. Given the permutation π, we say that a vertex $v \in V \setminus V'$ is a *right-bend* of an edge-path if its two neighbors, u, w, satisfy $\pi(u) < \pi(v)$ and $\pi(w) < \pi(v)$, i.e., both u and w appear to the left of v in π. Similarly, v is a *left-bend* if its two neighbors appear to its right in π.

In proving the lemma, we distinguish between two cases:

(i) Gap i does not contain a vertex that is a bend of an edge-path; thus, for any vertex v in gap i and its two neighbors, u, w, either $\pi(u) < \pi(v) < \pi(w)$, or $\pi(w) < \pi(v) < \pi(u)$. Suppose that gap i contains two adjacent vertices z, v, that belong to different edge-paths, generated from $e'_1, e'_2 \in E'$, respectively. More precisely, let $\pi = (v_{i_1}, \ldots, x, \ldots, u, \ldots v, z, \ldots, w, \ldots r, \ldots, v_{i_n})$, where r, x are the neighbors of z on the edge-path of e'_1, and u, w are the neighbors of v on the edge-path of e'_2. Let π be the permutation resulting from a swap of z and v in π, then the difference between the costs of the two permutations is given by

$$\begin{aligned} cost(\pi') - cost(\pi) = & [(\pi(r) - \pi(z) + 1) + (\pi(w) - \pi(v) - 1) \\ & + (\pi(z) - \pi(x) - 1) + (\pi(v) - \pi(u) + 1)] \\ & - [(\pi(r) - \pi(z)) + (\pi(w) - \pi(v)) \\ & + (\pi(z) - \pi(x)) + (\pi(v) - \pi(u))] = 0 \end{aligned}$$

Hence, we can perform a sequence of such swaps, until we have that all of the vertices that belong to the same edge-path appear consecutively in gap i, without increasing the cost.

(ii) Suppose that gap i contains a vertex v that belongs to an edge-path of $e'_1 \in E'$, where v is a bend. W.l.o.g. we assume that v is a right bend, i.e., its two neighbors, u and w, appear to the left of v in π (the proof is similar for the case where v is a left bend). Suppose that $\pi(w) < \pi(u) < \pi(v)$, and assume that the vertex $z \notin e'_1$ precedes v in gap i, i.e., we have $\pi = (v_{i_1}, \ldots, w \ldots, u, \ldots, z, v, \ldots, v_{i_n})$. We show that swapping z and v does not increase the total cost. Let π' be the resulting permutation. We consider the following cases:

   (a) If z is not a bend, then it has two neighbors x, r, such that $\pi(x) < \pi(z) < \pi(r)$. Hence, we have that $cost(\pi') - cost(\pi) < 0$.

   (b) Suppose that z is a right bend, then its two neighbors in the edge-path, x, r, are positioned to the left of z in π. W.l.o.g., assume that $\pi(x) < \pi(r) < \pi(z)$. More precisely, let $\pi = (v_{i_1}, \ldots, x \ldots, w, \ldots, r, \ldots, u, \ldots z, v, \ldots, v_{i_n})$. Then, it is easy to verify that $cost(\pi') - cost(\pi) = 0$.

(c) If z is a left bend then its two neighbors, x, r, are positioned to its right in π. W.l.o.g., assume that $\pi(z) < \pi(r) < \pi(x)$, and $\pi = (v_{i_1}, \ldots, w \ldots, u, \ldots, z, v, \ldots r, \ldots, x, \ldots, v_{i_n})$. Indeed, as a result of the swap, both v and z become closer to their neighbors. Hence, $cost(\pi') - cost(\pi) < 0$. □

Lemma 5. *Given a permutation $\sigma \in S_{k'}$ for V', and the number of vertices of edge-path e' in gap i, $x_{e',i}$, $0 \leq i \leq k'$, there exists an optimal order for V where each gap i for which $x_{e',i} > 0$ contains exactly one or two contiguous segments of e'.*

Theorem 2. *There is an FPT algorithm for MLA parameterized by the maximum leaf number.*

We use in the proof the next lemma.

Lemma 6. *Given a permutation σ and the vector $\bar{x}$, $Cost(G|\ \sigma, \bar{x})$ is linear and can be computed in FPT time.*

Proof. We can write $Cost(G|\ \sigma, \bar{x}) = \sum_{i=0}^{k'} Cost(\text{gap } i|\sigma, \bar{x})$, where $Cost(\text{gap } i|\sigma, \bar{x})$ is the contribution of gap i to the total cost. Specifically, suppose that the two ends of gap i are $x, y \in V'$, where $\pi(x) < \pi(y)$. Let $(v_1, \ldots, v_h)$ be a segment of edge-path e' in gap i. W.l.o.g., we assume that v_1 is closer to x. Then we compute the contribution of this segment to the cost of gap i as follows. We compute the cost incurred by each internal edge on this segment and add to that a term that depends on the type of the segment. By Lemma 5, we may assume that the segment $(v_1, \ldots, v_h)$ is contiguous.

(a) If the segment is straight (i.e., has no bend in gap i), we add the distance from v_1 to x and from v_h to y. This term would then partially account for the cost incurred by the edges connecting the two ends of the segment, v_1 and v_h, to their neighbors in other gaps.
(b) If the segment has a right bend, we add the distance between v_1 and x plus the distance between v_h and x.
(c) If the segment has a left bend, we add the distance between v_1 and y and the distance between v_h and y.

Thus, it suffices to show that $Cost(\text{gap } i|\sigma, \bar{x})$ is linear and can be computed in FPT time. We now show how to order optimally the segments in gap i. Let B_ℓ, B_r and S denote the sets of segments of types *left-bend*, *right-bend* and *straight* in gap i, respectively. We denote by $z_i = \sum_{e' \in E'} x_{e',i}$ the number of vertices assigned to gap i. We give the proofs of the next claims in the Appendix.

Claim 3 *The cost incurred by any segment of type S in gap i is equal to z_i.*

Claim 4 *Given a set of segments in B_r in gap i, of lengths $x_{e_1,i} \leq x_{e_2,i} \leq \cdots \leq x_{e_r,i}$, the minimum total cost incurred by these segments in gap i is given by $r \cdot x_{e_1,i} + (r-1)x_{e_2,i} + \cdots + x_{e_r,i}$. Similarly, given a set of segments in B_ℓ, of lengths $x_{e_1,i} \leq x_{e_2,i} \leq \cdots \leq x_{e_\ell,i}$, the minimum cost incurred by these segments is $\ell \cdot x_{e_1,i} + (\ell-1)x_{e_2,i} + \cdots + x_{e_\ell,i}$.*

Thus, letting $|B_r| = r$ and $|B_\ell| = \ell$, by Claims 3 and 4 we have that the total cost of gap i is given by

$$Cost(\text{gap } i|\sigma, \bar{x}) = |S| \cdot \sum_{e' \in E'} x_{e',i} + \sum_{j=1}^{r}(r - j + 1)x_{e_j,i} + \sum_{j=1}^{\ell}(\ell - j + 1)x_{e_j,i}, \quad (1)$$

where $e_1, \ldots, e_r$ are the edge-paths having in gap i segments in B_r, and $e_1, \ldots, e_\ell$ are the edge-paths having in gap i segments in B_ℓ. For a vector $\bar{x}$ that is consistent with a given configuration of E', we have that the values of r and ℓ in (1) are fixed, thus $Cost(\text{gap } i|\sigma, \bar{x})$ is linear, and so is $Cost(G|\sigma, \bar{x})$. Hence, we can solve the integer program in FPT time (see, e.g., [16, 14]). □

Proof of Theorem 2: Clearly, by Lemmas 4 and 5, our algorithm exhaustively searches over the set of solution patterns which contains an optimal one. It remains to show that the algorithm runs in FPT time. We note that the two outer loops of Algorithm 2 require $O(k'! \cdot 2^{k^2})$ iterations, each requires solving an integer linear program having $O(k^2)$ variables. This gives the statement of the theorem. □

4 MLA parameterized by Edge-Clique-Cover Number

In this section we show that MLA parameterized by the edge clique cover number of a the graph is in FPT. More precisely, we prove the following theorem:

Theorem 5. *There is an $O^*(2^k!)$-time algorithm for MLA where $k = eec(G)$ is the edge clique cover number of the input graph G.*

Two vertices $u, v \in V$ of G are said to be of the same *type* if $N[u] = \{u\} \cup N(u) = \{v\} \cup N(v) = N[v]$, where $N(u)$ and $N(v)$ respectively denote the set of neighbors of u and v in G. Note that our notion of type here is different from the one we use in Section 2, as vertices of the same type are necessarily adjacent. Nevertheless the two different concepts of types are conceptually very similar, as we can prove a certain homogenous lemma for this notion of type as well.

Lemma 7 (Homogeneity for $ecc(G)$). *There exists an optimal vertex ordering which is* homogenous. *That is, an ordering in which vertices of each type appear consecutively.*

Proof of Theorem 5 [assuming Lemma 7]: By Lemma 7, there exists an optimal solution in which vertices of each type appear consecutively. Observe that in any such homogenous ordering, the ordering of vertices of the same type can be arbitrary. That is, reordering vertices of a given type does not affect the total edge lengths of the ordering. Now, it is well known that a graph with edge clique cover number at most k has at most 2^k different types [12]. Thus, our algorithm searches through all $O(2^k!)$ homogenous vertex orderings and outputs the best one. □

Algorithm 3 MLA parametrized by $ecc(G)$

Input: (G, k).
Output: A linear ordering π for the vertices of G.

1: Compute the type of each vertex of G.
2: Compute the cost of each homogenous ordering of types.
3: **return** a homogenous ordering with minimum cost.

To prove Lemma 7, we introduce some additional notation. For an ordering $\pi = V \to \{1, \ldots, n\}$ and a pair of vertices $u, v \in V$ such that $\pi(u) < \pi(v)$, we let $\overleftarrow{\pi_{u,v}}$ and $\overrightarrow{\pi_{u,v}}$ denote the following permutations:

$$\overleftarrow{\pi_{u,v}}(x) = \begin{cases} \pi(x) & \text{if } \pi(x) \leq \pi(u) \text{ or } \pi(v) < \pi(x), \\ \pi(u) + 1 & \text{if } x = v, \\ \pi(x) + 1 & \text{if } \pi(u) < \pi(x) < \pi(v), \end{cases}$$

and

$$\overrightarrow{\pi_{u,v}}(x) = \begin{cases} \pi(x) & \text{if } \pi(x) < \pi(u) \text{ or } \pi(v) \leq \pi(x), \\ \pi(v) - 1 & \text{if } x = u, \\ \pi(x) - 1 & \text{if } \pi(u) < \pi(x) < \pi(v). \end{cases}$$

Thus, $\overleftarrow{\pi_{u,v}}$ is the vertex ordering obtained from π by placing v directly after u, and $\overrightarrow{\pi_{u,v}}$ is the ordering obtained from π by placing u directly before v.

Lemma 8. *Let $\pi = V \to \{1, \ldots, n\}$ be an optimal vertex ordering, and let u and v be two vertices of the same type with $\pi(u) < \pi(v)$. Then both $\overleftarrow{\pi_{u,v}}$ and $\overrightarrow{\pi_{u,v}}$ are optimal as well.*

Proof. Define three set of vertices $A = \{x \in V : \pi(x) < \pi(u)\}$, $B = \{x \in V : \pi(u) < x < \pi(v)\}$, and $C = \{x \in V : \pi(v) < \pi(x)\}$, and consider the two quantities $\overleftarrow{\Delta} = cost(\overleftarrow{\pi_{u,v}}) - cost(\pi)$ and $\overrightarrow{\Delta} = cost(\overrightarrow{\pi_{u,v}}) - cost(\pi)$. As π is optimal, both these quantities are non-negative, *i.e.* $\overleftarrow{\Delta} \geq 0$ and $\overrightarrow{\Delta} \geq 0$. If the lemma were false, at least one of these quantities would be strictly positive, *i.e.* we would either have $\overleftarrow{\Delta} > 0$ or $\overrightarrow{\Delta} > 0$. Aiming towards a contradiction let us assume that $\overleftarrow{\Delta} > 0$ (the proof in case $\overrightarrow{\Delta} > 0$ is symmetric).

Let us examine which edges contribute to $\overleftarrow{\Delta}$, *i.e.* which edges $\{x, y\} \in E$ have different lengths in $\overrightarrow{\pi_{u,v}}$ and π ($|\overrightarrow{\pi_{u,v}}(x) - \overrightarrow{\pi_{u,v}}(y)| \neq |\pi(x) - \pi(y)|$). Clearly, edges in $E(A, A)$, $E(B, B)$, $E(C, C)$ and $E(A, C)$ do note contribute to $\overleftarrow{\Delta}$. All other edge lengths are different in the two orderings. The length of each edge in $E(v, C)$ increases by $|B|$ in $\overleftarrow{\pi_{u,v}}$ when compared to its length in π, while the length of each edge in $E(v, A)$ decreases by $|B|$. Similarly, the length of each edge in $E(A, B)$ increases by 1, while the length of each edge in $E(B, C)$ decreases by 1.

What remains to account for are the edges involving u and v. Let $\ell(u, B) = \sum_{b \in B} |\pi(u) - \pi(b)|$ and $\ell(v, B) = \sum_{b \in B} |\pi(v) - \pi(b)|$ denote the total length of the edges of $E(u,B)$ and $E(v,B)$ with respect to π. Observe that as u and v are twins (and are thus adjacent to the same set of vertices in B), the total length of all edges of $E(v,B)$ becomes $\ell(u,B)$ in $\overleftarrow{\pi_{u,v}}$, while the length of each edge of $E(u,B)$ increases by 1. Thus, the total contribution of edges in $E(u,B) \cup E(v,B)$ to $\overleftarrow{\Delta}$ is $|E(u,B)| + \ell(u,B) - \ell(v,B)$. Finally, the last edge contributing to $\overleftarrow{\Delta}$ is the edge $\{u,v\}$ itself (which necessarily exists as u and v are from the same type) whose length decreases by $|B|$ in $\overleftarrow{\pi_{u,v}}$ when compared to π.

Summing up all these contributions, we get the following equality for $\overleftarrow{\Delta}$:

$$\begin{aligned}\overleftarrow{\Delta} = {} & |E(v,C)| \cdot |B| \, + \, |E(A,B)| \, + \, \ell(u,B) \, + \, |E(u,B)| \\ & - |E(v,A)| \cdot |B| \, - \, |E(B,C)| \, - \, \ell(v,B) \, - \, |B|. \end{aligned} \tag{2}$$

Symmetrically, a similar calculation will give us the following equation for $\overrightarrow{\Delta}$:

$$\begin{aligned}\overrightarrow{\Delta} = {} & |E(u,A)| \cdot |B| \, + \, |E(B,C)| \, + \, \ell(v,B) \, + \, |E(v,B)| \\ & - |E(u,C)| \cdot |B| \, - \, |E(A,B)| \, - \, \ell(u,B) \, - \, |B|. \end{aligned} \tag{3}$$

Now as u and v are twins, we know that $|E(u,A)| = |E(v,A)|$ and $|E(u,C)| = |E(v,C)|$. Furthermore, we also have $|B| \geq |E(u,B)| = |E(v,B)$. Thus, using equations (2) and (3), and our assumption that $\overleftarrow{\Delta} > 0$ and $\overrightarrow{\Delta} \geq 0$, we get

$$\begin{aligned} |E(v,C)| \cdot |B| \, + \, |E(A,B)| \, + \, \ell(u,B) \, + \, |E(u,B)| &> \\ |E(v,A)| \cdot |B| \, + \, |E(B,C)| \, + \, \ell(v,B) \, + \, |B| &\geq \\ |E(u,A)| \cdot |B| \, + \, |E(B,C)| \, + \, \ell(v,B) \, + \, |E(v,B)| &\geq \\ |E(u,C)| \cdot |B| \, + \, |E(A,B)| \, + \, \ell(u,B) \, + \, |B| &\geq \\ |E(v,C)| \cdot |B| \, + \, |E(A,B)| \, + \, \ell(u,B) \, + \, |E(u,B)| & \end{aligned}$$

our desired contradiction. □

Proof of Lemma 7: Let π be an optimal vertex ordering. If π is not homogenous we can use Lemma 8 repeatedly to transform it into one. The lemma thus follows.

5 Summary and Open Problems

We have shed light on the complexity of MLA for structural parameterizations stronger than treewidth, including a nice example of a successful combination of parameterization and approximation. We believe our algorithmic strategy in this may be widely applicable. Can this success be extended to the aggregate parameterization based on treewidth? In this paper we have aimed only at qualitative FPT results. We leave the best running times for such FPT algorithms as an open problem.

References

1. C. Ambühl, M. Mastrolilli, and O. Svensson. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM J. Comput.*, 40(2):567–596, 2011.
2. H. L. Bodlaender, M. R. Fellows, and M. T. Hallett. Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy. In *Proceedings of the 26th Annual Symposium on the Theory of Computing (STOC)*, pages 449–458, 1994.
3. H. L. Bodlaender, F. V. Fomin, A. M. C. A. Koster, D. Kratsch, and D. M. Thilikos. A note on exact algorithms for vertex ordering problems on graphs. *Theory of Computing Systems*, 50(3):420–432, 2012.
4. M. Charikar, M. T. Hajiaghayi, H. J. Karloff, and S. Rao. $l^2{}_2$ spreading metrics for vertex ordering problems. In *SODA*, pages 1018–1027, 2006.
5. U. Feige and J. R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Inf. Process. Lett.*, 101(1):26–29, 2007.
6. M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. A. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *IInformation and Computation*, 209(2):143–153, 2011.
7. M. R. Fellows, B. M. P. Jansen, and F. A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013.
8. M. R. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. A. Rosamond, and S. Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory Comput. Syst.*, 45(4):822–848, 2009.
9. M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *ISAAC*, pages 294–305, 2008.
10. H. Fernau. Parameterized algorithmics for linear arrangement problems. *Discrete Applied Mathematics*, 156(17):3166–3177, 2008.
11. M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.
12. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics*, 13, 2008.
13. G. Gutin, A. Rafiey, S. Szeider, and A. Yeo. The linear arrangement problem parameterized above guaranteed value. *Theory Comput. Syst.*, 41(3):521–538, 2007.
14. R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.
15. D. J. Kleitman and D. B. West. Spanning trees with many leaves. *SIAM J. Discrete Math.*, 4(1):99–106, 1991.
16. H. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
17. S. Rao and A. W. Richa. New approximation techniques for some linear ordering problems. *SIAM J. Comput.*, 34(2):388–404, 2004.

A Some Proofs

Proof of Lemma 3: Consider an arbitrary optimal arrangement π for G. We note that in Step 9 of the algorithm we consider only assignments with an integral number of mega-vertices of each type in the gaps. However, it may be the case that π contains fractional assignments of mega-vertices in some gap. Nevertheless, the increase of the number of vertices in any of the gaps in the corresponding integral assignment is at most $s \cdot 2^k = \frac{\varepsilon' m}{4k^2(k+1)}$. Thus, Algorithm 1 considers an assignment where the length of each edge increases by at most $\frac{\varepsilon m}{4k^2}$ in comparison to its length in π, and the lemma follows. □

Proof of Lemma 5: Suppose that the edge-path of e' contains the vertices $(a, v_1, \ldots, v_h, b)$, where $a, b \in V'$ and $v_1, \ldots, v_h \in V \setminus V'$. Thus, e' incurs cost for $h + 1$ edges. For any edge on this path, the cost of connecting two neighboring vertices v_i, v_j that appear in different gaps is 2 or higher, since at least one vertex in V' separates between these vertices in the vertex ordering, π, .there is an optimal ordering for V which assigns we in each gap either a single contiguous segment, or two contiguous segments of e'. Let $\sigma(a) = f$ and $\sigma(b) = g$, and assume w.l.o.g. that $1 \leq f < g \leq k'$. Let $L = min\{i \mid x_{e',i} > 0\}$, and $L = max\{i \mid x_{e',i} > 0\}$, i.e., L and R are the indices of the leftmost and rightmost gaps that contain vertices of e', respectively. We first consider the cost incurred by *internal* edges, i.e., edges connecting neighboring vertices of e' that are placed in the same gap.

W.l.o.g., we may assume that $x_{e',i} > 1$ We distinuish between the following cases.

1. $R \geq g$ and $L \leq f - 1$ (see Figure 1). Then we consider four types of gaps.
   - (i) In any gap $f \leq i \leq g - 1$, e' must contain a vertex $z \in \{v_1, \ldots, v_h\}$ having neighbor in gap $j \geq i + 1$, and a vertex $x \in \{v_1, \ldots, v_h\}$ having neighbor in gap $j' \leq i - 1$, by the vector $\bar{x}_{ie'}$. We can take a contiguous segment of length $t = x_{e',i}$, given by $(v_s, \ldots, v_{s+t-1})$, and assign these vertices consecutively from left to right, i.e., $\pi(v_i) = \pi(v_{i-1}) + 1$, for all $s + 1 \leq i \leq s + t - 1$. Thus, the cost incurred by each internal edge is 1.
   - (ii) In gap R, e' must contain at least two vertices which have neighbors in gaps to the left of R. As argued above, if an edge between two vertices in e' is *external* (i.e., its ends are placed in different gaps), the cost incurred by this edge is at least 2. Let $x_{e',R} = q$, and consider a contiguous segment $(x_p, \ldots, x_{p+q-1})$. Let $d = \lfloor q/2 \rfloor$. We choose v_{p+d} as a right-bend and order the vertices of the segment from left to right as follows: $(v_p, v_{p+q-1}, v_{p+1}, v_{p+q-2}, \ldots, v_{p+d-1}, v_{p+d+1}, v_{p+d})$. Then, the cost incurred by each internal edge in this segment is at most 2.
   - (iii) Similarly, we can order the vertices of a contiguous segment in gap L with a single left-bend, such that each internal edge incurs at most the cost 2.
   - (iv) If $R > g$, or $L < f - 1$, then there exist some gap i in which two vertices of e' must have neighbors in gaps to the right of i, and two vertices must

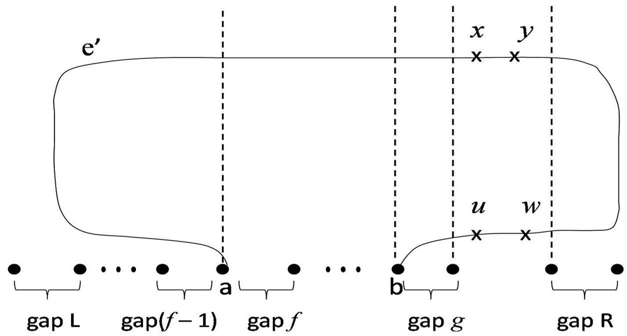

Fig. 1. The layout of an edge-path connecting a and b: Case 1. in the proof of Lemma 5

have neighbors to the left of i (see the vertices x, u and y, w in Figure 1). In each such gap, we can assign two contiguous segments of e' and order each as a straight line. The cost incurred by each internal edge in these segments is 1.

2. $R \geq g$ and $L > f-1$. Similar to Case 1., except that there is no left-bend (see Figure 2).
3. $R < g$ and $L \leq f-1$. Similar to Case 1., except that there is no right-bend.
4. $R < g$ and $L > f-1$. In this case, the edge-path of e' has neither a right-bend nor a left-bend. Therefore, we are in sub-case (i) of 1.

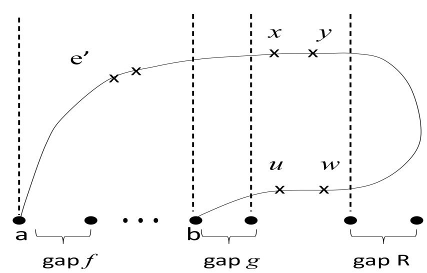

Fig. 2. Case 2. in the proof of Lemma 5

Hence, we may assume that any gap i for which $x_{e',i} > 0$ contains either one or two contiguous sgements of the edge-path of e'. It remains to select the contiguous segments placed in these gaps. We now consider the cost incurred by external edges. We note that the total cost incurred by these edges is minimized by assigning the contiguous segments of the edge-path such that the left (right) neighbor of each segment is as closest as possible from the left (right). Indeed, this follows from the fact that the cost of each external adge between neighbors on an edge-path depends on the total number of vertices in the gaps separating between these neighbors. We can obtain such optimal assignment by placing the segments continuously, starting from a, proceeding towards L, then towards R and finally to b. □

Proof of Claim 3: Consider an order in which the vertices of the segment appear consecutively in gap i. Let $(v_1, \ldots, v_h)$ be a segment in S and assume, w.l.o.g, that v_1 is closer to x, the left end of gap i. By Lemma 4, we may assume that the vertices on this segement appear consecutively in gap i. Then the cost incurred by this segment is given by the length of this segment plus the distance between v_1 and x plus the distance between v_h and y, which sum to z_i. □

Proof of Claim 4: We give the detailed proof for B_r. The proof is similar for B_ℓ. By Lemma 4, we may assume that each segment in B_r is assigned consecutively in gap i. Since each end of a segment in B_r has a neighbor in some gap $j < i$, by an interchange argument, the minimum cost for these segments is achieved by assigning them from left to right in non-decreasing order of lengths, i.e., the leftmost segment in gap i belongs to e_1 in B_r, the next one belongs to e_2 in B_r and so on.

We assign the segments of B_ℓ in gap i similarly, from right to left, by placing the shortest among them to be the rightmost in gap i, the second segment in B_ℓ to its left and so on. □