Minimum Linear Arrangement of Series-Parallel Graphs*

Martina Eikel
University of Paderborn, Paderborn, Germany
martinah@upb.de

Christian Scheideler
University of Paderborn, Paderborn, Germany
scheideler@upb.de

Alexander Setzer
University of Paderborn, Paderborn, Germany
asetzer@mail.upb.de

Abstract

We present a factor $14D^2$ approximation algorithm for the minimum linear arrangement problem on series-parallel graphs, where D is the maximum degree in the graph. Given a suitable decomposition of the graph, our algorithm runs in time $O(|E|)$ and is very easy to implement. Its divide-and-conquer approach allows for an effective parallelization. Note that a suitable decomposition can also be computed in time $O(|E| \log |E|)$ (or even $O(\log |E| \log^* |E|)$ on an EREW PRAM using $O(|E|)$ processors).

For the proof of the approximation ratio, we use a sophisticated charging method that uses techniques similar to amortized analysis in advanced data structures.

On general graphs, the minimum linear arrangement problem is known to be NP-hard. To the best of our knowledge, the minimum linear arrangement problem on series-parallel graphs has not been studied before.

*This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901).

1 Introduction

The minimum linear arrangement problem is a well-known graph embedding problem, in which an arbitrary graph is mapped onto the line topology, such that the sum of the distances of nodes that share an edge is minimized. We consider the class of series-parallel graphs, which arises naturally in the context of parallel programs: modelling the execution of a parallel program yields a series-parallel graph, where sources of parallel compositions represent fork points, and sinks of parallel compositions represent join points (for the definition of a parallel composition, see Subsection 1.1). Note that in this context, series-parallel graphs typically have a very low node degree: Since spawning child processes is costly, one would usually not spawn too many of them at a time.

1.1 Problem statement and definitions

Throughout this work, we consider undirected graphs only. The following definition of the minimum linear arrangement problem is based on [21]:

Definition 1 (Linear arrangement)**.** Given a graph $G = (V, E)$, let $n = |V|$. A *linear arrangement* π of G is a one-to-one function

$$\pi : V \to \{1, \ldots, n\}.$$

For a node $v \in V$, $\pi(v)$ is also called the *position of* v in π.

Definition 2 (Cost of a linear arrangement)**.** Given a graph $G = (V, E)$ and a linear arrangement π of G, we denote the *cost* of π by

$$COST_\pi(G) := \sum_{\{u,v\} \in E} |\pi(u) - \pi(v)|.$$

Definition 3 (Minimum linear arrangement problem)**.** Given a graph $G = (V, E)$ (the *input graph*), the *minimum linear arrangement problem* (MINLA) is to find a linear arrangement π that minimizes $COST_\pi(G)$.

Next we define the class of series-parallel graphs, which we need several definitions for (the following is based on [10]):

Two-terminal graph (TTG) A *two-terminal graph* $G = (V, E)$ is a graph with node set V, edge set E, and two distinct nodes $s_G, t_G \in V$ that are called source and sink, respectively. s_G and t_G are also called the *terminals* of G.

Series composition The *series composition* SC of $k \geq 2$ TTGs $X_1, \ldots, X_k$ is a TTG created from the disjoint union of $X_1, \ldots, X_k$ with the following characteristics: The sink t_{X_i} of X_i is merged with the source $s_{X_{i+1}}$ of X_{i+1} for $1 \leq i < k$. The source s_{X_1} of X_1 becomes the source s_{SC} of SC and the sink t_{X_k} of X_k becomes the sink t_{SC} of SC.

Parallel composition The *parallel composition* PC of $k \geq 2$ two-terminal graphs $X_1, \ldots, X_k$ is a TTG created from the disjoint union of $X_1, \ldots, X_k$ with the following two characteristics: The sources $s_{X_1}, \ldots, s_{X_k}$ are merged to create s_{PC} and the sinks $t_{X_1}, \ldots, t_{X_k}$ are merged to create t_{PC}.

Two-terminal series-parallel graph (TTSPG) A *two-terminal series-parallel graph* G with source s_G and sink t_G is a graph that may be constructed by a sequence of series and parallel compositions starting from a set of copies of a single-edge two-terminal graph $G' = (\{s, t\}, \{\{s, t\}\})$.

Series-parallel graph A graph G is a *series-parallel graph* if, for some two distinct nodes s_G and t_G in G, G can be regarded as a TTSPG with source s_G and sink t_G.

Note that the series and parallel compositions are commonly defined over two input graphs only. However, it is not hard to see that our definition of a series-parallel graph is equivalent.

An example of a series-parallel graph is shown in Figure 1.

1.2 Related work

The MINLA was first stated by Harper [17]. Garey, Johnson, and Stockmeyer were the first to prove its NP-hardness on general graphs [15]. Ambühl, Mastrolilli, and Svensso showed that the MINLA on general graphs does not have a polynomial-time approximation scheme unless NP-complete problems can be solved in randomized subexponential time [3]. To the best of our knowledge, the two best polynomial-time approximation algorithms for the MINLA on general graphs are due to Charikar, Hajiaghayi, Karloff, and Rao [6], and Feige and Lee [12]. Both algorithms yield an $O(\sqrt{\log n}\log\log n)$-approximation of the MINLA. The latter algorithm is a combination of techniques of earlier works by Rao and Richa [23], and Arora, Rao, and Vazirani [4]. For planar graphs (which include the series-parallel graphs), Rao and Richa [23] also present a $O(\log\log n)$-approximation algorithm. Note that even though, for high degree graphs, these algorithms achieve a better approximation factor than the one we present in this work, there are some key differences between these algorithms and ours: First of all, the algorithm we present is a very simple divide-and-conquer algorithm and its functioning can be understood easily. The aforementioned algorithms, however, are much more complex and involve solving a linear or semidefinite program. Furthermore, our algorithm achieves a runtime of only $O(|E|)$ (if the series-parallel graph is given in a suitable format - otherwise, a more complex preprocessing is required that takes time $O(|E|\log|E|)$, but this can be parallelized down to $O(\log|E|\log^*|E|)$) making it suitable in situations where a low runtime is more important than the approximation guarantee. Still, for low graph degrees (which are reasonable to assume in certain applications), our algorithm even improves the approximation factor of Rao and Richa.

For special classes of graphs, the NP-hardness has been shown for bipartite graphs [11], interval graphs, and permutation graphs [8]. On the other hand, polynomial-time optimal algorithms have been found for hypercubes [17], trees [7], d-dimensional c-ary cliques [20], meshes [13], and chord graphs [24]. Note that many people claim that the MINLA is optimally solvable on outerplanar graphs, referring to [14]. However, the problem solved in [14] is different from the MINLA as we show in [25]. Note that the question whether the MINLA is NP-hard on series-parallel graphs is unsettled.

Applications of the MINLA include the design of error-correcting codes [17], machine job scheduling (e.g., [2]), VLSI layout (e.g., [1, 9]), and graph drawing (e.g., [26]). For an overview of heuristics for the MINLA see the survey paper by Petit [22].

The class of series-parallel graphs, first used by MacMahon [19], has been studied extensively. It turns out that many problems that are NP-complete on general graphs can be solved in linear time on series-parallel graphs. Among these are the decision version of the dominating set problem [18], the minimum vertex cover problem, the maximum outerplanar subgraph problem, and the maximum matching problem [27]. Furthermore, since the class of series-parallel graphs is a subclass of the class of planar graphs, any problem that is already in P for that class of graphs can be solved optimally in polynomial time for series-parallel graphs as well (such as the max-cut problem [16]).

Another problem regarding series-parallel graphs is to decide, given an input graph G, whether it is series-parallel and, if so, to output the operations that recursively constructed the series-parallel graph. The first step is referred to as *series-parallel graph recognition* while the second step is referred to as *constructing a decomposition tree*. A parallel linear-time algorithm for this problem on directed graphs was first presented by Valdes, Tarjan, and Lawler [28]. Later, Eppstein [10] developed a parallel

algorithm for undirected graphs using a so-called *nested ear decomposition*. The concept of an S-decomposition used in our analysis is technically similar to that concept, though we use a different notation more suitable for our purposes. The algorithm we propose for approximating the MINLA on series-parallel graphs also relies on a decomposition tree. For instances in which it is not given, the algorithm by Bodlaender and De Fluiter [5] can be used, since it runs on undirected graphs and outputs so-called *SP-tree*, which can be easily transformed into a format suitable for our algorithm.

1.3 Our contribution

We describe a simple approximation algorithm for the minimum linear arrangement problem on series-parallel graphs with an approximation ratio of $14D^2$, where D is the degree of the graph, and a running time of $O(|E|)$ if the series-parallel graph is given in a suitable format. If the series-parallel graph is not given in the required format, this format can be computed in time $O(|E| \log |E|)$ (which can even be further parallelized down to $O(\log |E| \log^* |E|)$ on an EREW PRAM using $O(|E|)$ processors). However, for certain applications it is reasonable to assume that the graph is given in the right format, e.g., when the series-parallel graph is used to model the execution of a parallel program, the desired representation can be constructed along with the model. The simplicity and the structure of the algorithm allow for an efficient distributed implementation.

Moreover, our proof of the approximation ratio introduces a sophisticated charging method following an approach that is known from the amortized analysis of advanced data structures. This technique may be applied in other analyses as well.

2 Preliminaries

The algorithm we present is defined recursively and is based on a decomposition of the series-parallel graph into components. Therefore, prior to describing the algorithm, we introduce several definitions needed to formalize this decomposition.

The following definition is similar to the one in [5].

Definition 4 (SP-tree, minimal SP-tree)**.** An SP-tree T of a series-parallel graph G is a rooted tree with the following properties:

1. Each node in T corresponds to a two-terminal subgraph of G.
2. Each leaf is a so-called L*-node* labelled as $L(k)$ and corresponds to a path with k edges.
3. Each inner node is a so-called S*-node* or P*-node*, and the two-terminal subgraph G' associated with an S-node (P-node) is the graph obtained by a series (parallel) composition of the graphs associated with the children of G', where the order of the children defines the order in which the series composition is applied (the order does not matter for a parallel composition).
4. The root node corresponds to G.

An SP-tree T of a series-parallel graph G is called *minimal* if the following two conditions hold:

1. All children of an S-node are either P-nodes or L-nodes, but at least one is a P-node.
2. All children of a P-node are either S-nodes or L-nodes.

It is easy to see that for any fixed series-parallel graph G, there exists a minimal SP-tree for G.
We are now ready to introduce the following three important notions:

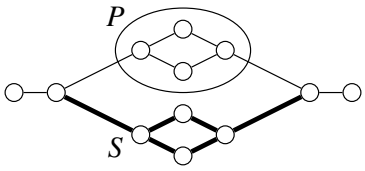


Figure 1: Example of a simple series-parallel graph. P is a parallel component consisting of two series components (more precisely, two simple node sequences with two edges each). The thick edges belong to the subgraph induced by the series component S. It consists of two single-edge simple node sequences (on the left and right end) and a parallel component.

Definition 5 (Simple node sequence, Parallel component, Series component)**.** Let G be a series-parallel graph and T be a minimal SP-tree of G. The sub-graph of G associated with a leaf $L(k)$ of T for $k \in \mathbb{N}$ is called a *simple node sequence*. The sub-graph of G associated with a P-node is called a *parallel component* of G. The sub-graph of G associated with a S-node is called a *series component* of G. Furthermore, any simple node sequence is called a *series component*, too.

An illustration of the different types of components is given by Figure 1. The definition of a minimal SP-tree implies the following: Each parallel component P is the result of a parallel composition of two or more series components. Furthermore, each series component S is the result of a series composition of two or more parallel components or simple node sequences, but not exclusively simple node sequences. This leads to the following definition:

Definition 6 (Child component)**.** Let G be a series-parallel graph, let T be a minimal SP-tree, and let X and Y be two nodes in T such that Y is a child of X. Further, let C_i be the (series or parallel) component that is associated with Y and let C be the (parallel or series) component C that is associated with X. Then, C_i is called a *child component* of C, and we say: $C_i \in C$.

For example, the two simple node sequences that induce the parallel component P in Figure 1 are child components of P. One implication of this definition is that the terminals of a parallel component and its child components overlap.

For the rest of this work, we assume that for any fixed series-parallel graph G, the simple node sequences, series components and parallel components of G are uniquely defined by a fixed minimal SP-tree T. In Appendix A, we describe an efficient method to compute a minimal SP-tree according to our definition. It is basically an extension of an algorithm by Bodlaender and de Fluiter [5].

3 The series-parallel graph arrangement algorithm

The Series-Parallel Graph Arrangement Algorithm (SPGAA) is defined recursively. In order to arrange the nodes of a series or parallel component C, the SPGAA first determines the order of its child components recursively, and then places the child components side by side in an order that depends on their size. For any (series or parallel) component C, when the algorithm has just arranged the nodes of C, it holds that its source receives the leftmost position among all nodes of C and that its sink receives the position directly to the right of the source. However, later computations (in a higher recursion level) may re-arrange the terminals and pull them apart. More specific details are given in the corresponding subsections for the different types of components.

Illustrations of all arrangements and all different cases can be found in Appendix C.

3.1 Arrangement of a simple node sequence

For any simple node sequence L, we label the nodes of L from left to right by 1 to k. That is, the source receives label 1 and the sink receives label k. The arrangement of this sequence then is: $1, k, 2, k-1, 3, k-2, \ldots$. One can see that this arrangement fulfills the property that the source is on the leftmost position and that the sink is its right neighbor.

3.2 Arrangement of a parallel component

For any parallel component P with source u, sink v, and $m \geq 2$ child components $S_1, S_2, \ldots, S_m$ (note that any parallel component has at least two child components), the SPGAA recursively determines the arrangement of the child components. We denote the computed arrangement of S_i excluding the two terminal nodes (which would have been placed at the first two positions of the arrangement, see Subsection 3.3) by S_i^-. W.l.o.g. let S_m be a biggest child component (w.r.t. the number of nodes in it). Then, the algorithm places u at the first position, v at the second position, the nodes of S_1^- to the right of that (in their order), and the nodes of S_i^- to the right of S_{i-1}^- for $i \in \{2, \ldots, m\}$.

3.3 Arrangement of a series component

For any series component S with source u, sink v, and $m \geq 2$ child components $P_1, P_2, \ldots, P_m$ (note that any series component has at least two child components, otherwise it would be a simple node sequence), the SPGAA first recursively determines the arrangement of the child components. Second, it puts u and v at the first two positions, in this order. The third step differs from the case of a parallel component: To keep the cost of the arrangement low while ensuring that a biggest child component P_a receives the rightmost position, the general order of the child components is: $P_1, P_2, \ldots, P_{a-1}, P_m, P_{m-1}, \ldots, P_{a+2}, P_{a+1}, P_a$. Here, the components from P_m to P_{a+1} are flipped (the order of their nodes is reversed). For $m = a$, the order is $P_1, P_2, \ldots, P_m$ and for $a = 1$, the components are ordered in reverse (i.e., $P_m, P_{m-1}, \ldots, P_1$) (where all components except for P_1 are flipped).

However, since each two neighboring child components P_i and P_{i+1} share a (terminal) node, it must be decided which of the two components may "keep" its node. The strategy here is as follows: Each component P_i (except for the first component, whose source has received the leftmost position already) keeps its source and lends its sink to P_{i+1} (of which it is a source), except for P_m (whose sink has been placed at the second position already). This may stretch existing edges, which we will keep track of in the analysis.

An illustration of the arrangement for the case $1 < a < m$ can be found in Figure 2.

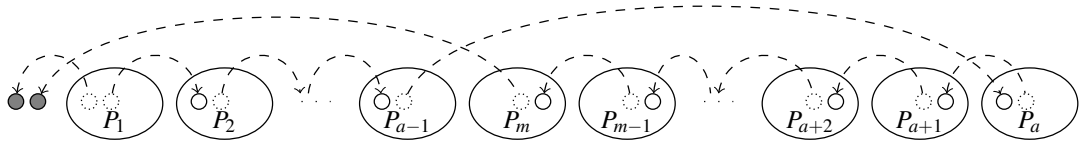


Figure 2: Order in which the SPGAA arranges a series component consisting of m child components for $1 < a < m$ (where P_a is a biggest component). Dotted nodes indicate the position at which a node would be placed according to the previous recursion level. Dashed arrows indicate the change in position at the current recursion level.

4 Analysis

In this section, we prove the approximation ratio of $14 \cdot D^2$ for the Series-Parallel Graph Arrangement Algorithm described in Section 3. As a first step, we provide lower bounds on the *amortized cost* in an optimal arrangement for each kind of component. The amortized cost of a component is the sum of two values: First, the exclusive cost of this component (cost of the current component minus the individual cost of all child components). Second, some cost that has been accounted for in a lower recursion level. This cost is chosen such that the sum of all amortized costs does not contain this cost more than three times. We use these bounds to establish a lower bound on the total cost of an optimal solution. The details are described in Subsection 4.2. As a second step, we state upper bounds on the exclusive costs generated at each recursion step of the SPGAA in order to determine an upper bound on the total cost in Subsection 4.3. Last, we use both the lower bound as well as the upper bound to relate the cost of an optimal arrangement to that of an arrangement computed by the SPGAA. This is done in Subsection 4.4. In addition to providing the approximation ratio of the SPGAA, we establish a polynomial runtime bound of our algorithm in Subsection 4.5. Note that all the proofs in this section can be found in Appendix B.

4.1 Prerequisites

For the analysis, we need several notions, which we now introduce.

Definition 7 (Length of an edge)**.** Given a graph $G = (V, E)$ and a linear arrangement π of G, let $u, v \in E$. The *length of* (u, v) *in* π, denoted by $length_\pi(u, v)$ is defined as:

$$length_\pi(u, v) = |\pi(u) - \pi(v)|.$$

Definition 8. Given a linear arrangement π of a series-parallel graph $G = (V, E)$ and a (series or parallel) component C in G, we define:

Restricted arrangement. The *arrangement* π *restricted to* C, denoted by $\pi(C)$ is obtained by removing all nodes from π that do not belong to C, as well as their incident edges, i.e., $\pi(C)$ maps the nodes from C to $\{1, \ldots, |C|\}$.

Restricted length of an edge. For any edge (u, v) that belongs to C, the *length of* (u, v) *restricted to* C, denoted by $length_{\pi(C)}(u, v)$, is the distance between u and v in $\pi(C)$.

Restricted cost of an arrangement. Let E_C be the set of all edges from G whose both endpoints are in C. The *cost of* C *restricted to* C, denoted by $R\text{-}COST_\pi(C)$, is defined as:

$$R\text{-}COST_\pi(C) := \sum_{(u,v)\in E_C} length_{\pi(C)}(u, v).$$

Definition 9 (Exclusive cost of a series / parallel component)**.** Given a linear arrangement π of a series-parallel graph G and a (series or parallel) component C in G containing $m \geq 0$ child components $C_1, \ldots, C_m$, the *exclusive cost of* C *in* π, denoted by $E\text{-}COST_\pi(C)$, is defined as

$$E\text{-}COST_\pi(C) := R\text{-}COST_\pi(C) - \sum_{i=1}^{m} R\text{-}COST_\pi(C_i).$$

Note that the exclusive cost of a simple node sequence S is equal to the restricted cost of S.

We can make the following observation regarding the relationship between the exclusive costs of the components and the total cost:

Observation 10. *Let G be a series-parallel graph and let π be a linear arrangement of G. Further, let $\mathcal{C}$ be the set of all (series or parallel) components in G. It holds:*

$$\sum_{C \in \mathcal{C}} E\text{-}COST_{\pi}(C) = COST_{\pi}(G).$$

In the analysis of the SPGAA, we need to find at least one path from s_P to t_P through P for each parallel component P such that any two such paths are edge-disjoint for two different parallel components. Therefore, we introduce the following notion of an *S-decomposition*, which yields these paths and is recursively defined as follows:

Definition 11 (A-path, S-path, S-decomposition)**.** Let P be an "innermost" parallel component in a series-parallel graph G (i.e., one whose child components are simple node sequences only) with source s, sink t, and k child components. Select an arbitrary simple path from s to t through P (i.e., select one of the simple node sequences). This path is called the *auxiliary path* or simply *A-path* of P. The remaining paths from s to t through P are called the *selected paths* or simply *S-paths* of P.

Recursively, for an arbitrary parallel component P, with source s, sink t, and $m \geq 2$ child components $S_1, \ldots, S_m$, for each child component S_i, $1 \leq i \leq m$, select a simple path Q_i from s to t through S_i in the following way: If S_i is a simple node sequence, Q_i is the whole sequence. Otherwise, S_i is a series component, which consists of $k \geq 0$ simple node sequences and $l \geq 1$ parallel components (note that $k + l \geq 2$). Denote these child components by $P_1, \ldots P_{k+l}$ in the order in which they appear in S_i. Construct the path Q_i step by step: Start with P_1 and add P_1 completely to Q_i if P_1 is a simple node sequence. If, however, P_1 is a parallel component, select the A-path of P_1 and extend Q_i by it. Continue in the same manner up to P_{k+l}. After this, the whole path Q_i is constructed. Q_1 is called the *A-path* of P and the remaining paths $Q_2, \ldots, Q_m$ are called the *S-paths* of P.

The selection of S-paths (and A-paths accordingly) for all parallel components of G is called an *S-decomposition* of G.

An example of an S-decomposition can be found in Figure 9 in Appendix C.

Intuitively, an auxiliary path of a parallel component P_j is a path through the whole component which is *reserved* to be used in higher recursion levels (to eventually become part of an S-path there). Any edges of an S-path are not used for any S-path or A-path in any higher recursion level.

The main contribution of the S-decomposition is that it gives a mapping from parallel components to paths through the respective components (the S-paths) such that all these paths are edge-disjoint. More formally:

Lemma 12. *For each series-parallel graph G, there exists an S-decomposition SD. Besides, in any S-decomposition, each edge belongs to at most one S-path in SD.*

Provided with the definition of an S-decomposition, we are ready to define the amortized cost as follows:

Definition 13 (Amortized cost)**.** Let π_{OPT} be an (optimal) linear arrangement of a series-parallel graph G, let SD be an S-decomposition of G, and let S be a series component in G. Further, let E_S be the set that contains all edges of simple node sequences that are child components of S and all edges of S-paths of the child components of S that are parallel components. The *amortized cost of S*, denoted by $A\text{-}COST_{\pi_{OPT}}(S)$, is defined as:

$$A\text{-}COST_{\pi_{OPT}}(S) := E\text{-}COST_{\pi_{OPT}}(S) + \sum_{\{x,y\} \in E_S} length_{\pi_{OPT}(S)}(x, y).$$

For any parallel component P in a series-parallel graph G and any optimal linear arrangement π_{OPT},

$$A\text{-}COST_{\pi_{OPT}}(P) := E\text{-}COST_{\pi_{OPT}}(P).$$

Note that the addend in the amortized cost for simple node sequences is zero (as the set E_S is empty in this case). As an example, the corresponding set E_{S_i} for the series component S_i from Figure 9 would contain all normally drawn edges, the two dashed S-paths of P_A and the lower path of P_B as shown in Figure 9(c). Note that E_{S_i} contains a path from s_{S_i} to t_{S_i}.

This definition will be helpful for the analysis of the minimum cost of an optimal arrangement. The amortized cost adds a certain value to the exclusive cost of a (series or parallel) component C, with the following property:

Lemma 14. *Let $G = (V, E)$ be a series-parallel graph, and π_{OPT} be an (optimal) linear arrangement for G. Further, let $\mathcal{C}$ be the set of all (series or parallel) components of G. It holds:*

$$\sum_{C \in \mathcal{C}} A\text{-}COST_{\pi_{OPT}}(C) \leq 3 \cdot \sum_{C \in \mathcal{C}} E\text{-}COST_{\pi_{OPT}}(C).$$

For the analysis of an optimal arrangement, we also need the following notation:

Definition 15 (Δ_C)**.** Given an (optimal) linear arrangement π_{OPT} of a series-parallel graph G, and a (series or parallel) component C in G, consider π_{OPT} restricted to C. We denote the smallest number of nodes to the left or to the right (depending on which number is smaller) of a terminal node of C in $\pi_{OPT}(C)$ by Δ_C.

It is convenient to define:

Definition 16 (Cardinality of a component)**.** For any series-parallel graph G and any (series or parallel) component C in G: $|C|$ is the number of nodes in C, $|C^{\ominus}|$ is the number of all nodes in C without the sink of C, and $|C^{-}|$ is the number of nodes in C without the two terminal nodes of C.

The following lemma is easy to show:

Lemma 17. *Let π_{OPT} be an (optimal) linear arrangement of a series-parallel graph G and let C be a (series or parallel) component. It holds:*

$$\Delta_C \leq \left\lfloor \frac{1}{2}(|C| - 2) \right\rfloor.$$

Last, we need the following definition:

Definition 18 (Spanning interval)**.** Given a graph G, an arrangement π of G, and a set S of nodes, let $u \in S$ such that $\pi(u) \leq \pi(s)$ for all nodes $s \in S$, and let $v \in S$ such that $\pi(v) \geq \pi(s)$ for all $s \in S$. Let I be the set of nodes such that $x \in I$ iff $\pi(x) \in [\pi(u), \pi(v)]$. Then, I is the *interval spanning S in π*.

4.2 A lower bound on the total cost of optimal solutions

In this subsection, we give lower bounds on the amortized costs of an optimal arrangement for simple node sequences, parallel components, and series components. In the end, we consolidate the results and state a general lower bound on the total cost of an optimal arrangement.

Lemma 19. *For any simple node sequence L in a series-parallel graph G in an optimal arrangement π_{OPT}, it holds:*

$$A\text{-}COST_{\pi_{OPT}}(L) \geq |L| - 1 + \Delta_L.$$

We now provide a lower bound on the amortized cost of series components:

Lemma 20. *For any series component S in a series-parallel graph G with $m \geq 2$ child components $P_1, \ldots, P_m$ in an optimal arrangement π_{OPT}, it holds:*

$$A\text{-}COST_{\pi_{OPT}}(S) \geq \frac{1}{2}\left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}|\right) + 1 + \sum_{i=1}^{m} \Delta_{P_i} - \Delta_S.$$

In the following, we present the roadmap for the proof of Lemma 19. The full proof is given in Appendix B.

Proof sketch. The idea of the proof is the following: First of all, we note that (by Definition 13) the amortized cost of S is the sum of two values: the exclusive cost of S, and the sum over certain restricted edge lengths (the latter sum we denote by σ). Thus, we take a closer look at how the exclusive cost arises and at which edge lengths contribute to the value of σ.

For the additional term σ, observe that it is the sum of lengths (restricted to S) of all edges in the set E_S. This set contains all edges of simple node sequences that are child components of S and all edges of S-paths of the parallel child components of S.

For the exclusive cost of S, observe that they arise in the following way: An edge e has a greater length in the arrangement π_{OPT} restricted to S than in the arrangement π_{OPT} restricted to the child component P_i of S that e is from. We call the difference in the length of e the *additional stretching* of e. Then, the sum of the additional stretchings of all edges is the exclusive cost of S.

These two insights now enable use to determine a lower bound on the amortized cost of S by giving a lower bound on σ and on the sum of additional stretchings individually.

For the lower bound on σ, let V_S be the set of nodes that contains all the endpoints of edges in E_S and denote the interval spanning the set V_S in $\pi_{OPT}(S)$ by I (see Def. 18). Since there exists a simple path from the leftmost node in V_S (w.r.t. $\pi_{OPT}(S)$) to the rightmost node in V_S consisting of edges from E_S only, σ is at least $|I| - 1$.

For the lower bound on the sum of the additional stretchings, we consider nodes whose positions are to the left of those in I (the *part left of* I) and nodes whose positions are to the right of those in I (the *part right of* I) individually. We show that for any node in the part left of I, except for nodes in a sequence of nodes that starts with the leftmost node (this sequence will be called A), at least one edge is stretched by an amount of one due to this node. The analog can be shown for the part right of I (there, the corresponding sequence is called B). This yields a lower bound on the sum of additional stretchings linear in the number of these nodes (which is $|S| - |I| - |A| - |B|$).

All in all, this way we can find a lower bound on the amortized cost of S that is $|S| - 1 - |A| - |B|$. Applying some upper bounds on the sizes of A and B, we can rewrite this inequation as in the original claim of the lemma.

Using a similar technique as in the previous proof, we can also derive a lower bound for the amortized cost of parallel components:

Lemma 21. *For any parallel component P in a series-parallel graph G with $m \geq 2$ child components $S_1, \ldots, S_m$, in an optimal arrangement π_{OPT}, it holds:*

$$A\text{-}COST_{\pi_{OPT}}(P) \geq \frac{1}{2}\left(\sum_{i=1}^{m} |S_i^{-}| - \max_i |S_i^{-}|\right) + \sum_{i=1}^{m} \Delta_{S_i} - \Delta_P.$$

These three lower bounds for the different types of components in any series-parallel graph can be combined into a single lower bound:

Corollary 22. *Let $G = (V, E)$ be an arbitrary series-parallel graph and π_{OPT} an optimal arrangement of G. Further, denote the total cost of π_{OPT} by $COST_{\pi_{OPT}}(G)$, the set of simple node sequences in G by L_G, the set of parallel components by P_G, the set of series components by S_G. Then, it holds:*

$$7 \cdot COST_{\pi_{OPT}}(G) \geq \sum_{L \in L_G} 2 \cdot (|L| - 1) + \sum_{P \in P_G} \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) + \sum_{S \in S_G} \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right).$$

4.3 An upper bound on the total cost of SPGAA arrangements

For the approximation ratio of the SPGAA, we also need to find an upper bound on the cost of arrangements computed by the SPGAA. One can show the following result:

Corollary 23. *Let $G = (V, E)$ be an arbitrary series-parallel graph and let π_{ALG} be an arrangement of G computed by the SPGAA. Furthermore, denote the total cost of π_{ALG} by $COST_{\pi_{ALG}}(G)$, the set of simple node sequences in G by L_G, the set of parallel components by P_G, the set of series components by S_G. Then, it holds:*

$$COST_{\pi_{ALG}}(G) \leq \sum_{L \in L_G} 2 \cdot (|L| - 1) + \sum_{P \in P_G} 2D^2 \cdot \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) + \sum_{S \in S_G} 2D \cdot \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right).$$

4.4 The approximation ratio of $14 \cdot D^2$

Finally, based on the groundwork of the previous subsections, proving the main theorem of this chapter is straightforward.

Theorem 24. *For a series-parallel graph G, let π_{ALG} be the linear arrangement of G computed by the SPGAA, and let π_{OPT} be an optimal linear arrangement of G. It holds:*

$$COST_{\pi_{ALG}}(G) \leq 14 \cdot D^2 \cdot COST_{\pi_{OPT}}(G).$$

4.5 Runtime

Regarding the runtime of the SPGAA, one can show the following result:

Theorem 25. *On a series-parallel graph $G = (V, E)$, the SPGAA has a runtime of $O(|E|)$ if a minimal SP-tree of G is given as an input, and a runtime of $O(|E| \log |E|)$ otherwise.*

References

[1] D. Adolphson and T. C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

[2] Donald L Adolphson. Single machine job sequencing with precedence constraints. *SIAM Journal on Computing*, 6(1):40–54, 1977.

[3] Christoph Ambühl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *Proc. of FOCS*, pages 329–337. IEEE Computer Society, 2007.

[4] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2):5, 2009.

[5] Hans L. Bodlaender and Babette de Fluiter. Parallel algorithms for series parallel graphs. In *Proc. of ESA*, pages 277–289, London, UK, UK, 1996. Springer-Verlag.

[6] Moses Charikar, Mohammad Taghi Hajiaghayi, Howard Karloff, and Satish Rao. L22 spreading metrics for vertex ordering problems. In *Proc. of SODA*, pages 1018–1027, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[7] F. R. K. Chung. Labelings of graphs. *Selected topics in graph theory*, 3:151–168, 1988.

[8] Johanne Cohen, Fedor Fomin, Pinar Heggernes, Dieter Kratsch, and Gregory Kucherov. Optimal linear arrangement of interval graphs. In *Mathematical Foundations of Computer Science 2006*, pages 267–279. Springer, 2006.

[9] Josep Díaz, Jordi Petit, and Maria Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, September 2002.

[10] David Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992.

[11] Shimon Even and Yossi Shiloach. NP-completeness of several arrangement problems. *Dept. Computer Science, Technion, Haifa, Israel, Tech. Rep*, 43, 1975.

[12] Uriel Feige and James R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Information Processing Letters*, 101(1):26 – 29, 2007.

[13] Peter Fishburn, Prasad Tetali, and Peter Winkler. Optimal linear arrangement of a rectangular grid. *Discrete Math.*, 213(1-3):123–139, February 2000.

[14] Greg N. Frederickson and Susanne E. Hambrusch. Planar linear arrangements of outerplanar graphs. *Circuits and Systems, IEEE Transactions on*, 35(3):323–333, 1988.

[15] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976.

[16] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

[17] L. H. Harper. Optimal assignments of numbers to vertices. *Journal of the SIAM*, 12(1):131–135, 1964.

[18] T. Kikuno, N. Yoshida, and Y. Kakuda. A linear algorithm for the domination number of a series-parallel graph. *Discrete Applied Mathematics*, 5(3):299–311, 1983.

[19] P. A. Macmahon. The combination of resistances. *Electrician*, 28:601–602, 1892.

[20] Koji Nakano. Linear layouts of generalized hypercubes. In *Graph-theoretic concepts in computer science*, pages 364–375. Springer, 1994.

[21] Jordi Petit. Experiments on the minimum linear arrangement problem. *Journal of Experimental Algorithmics*, 8:2–3, 2003.

[22] Jordi Petit. Addenda to the survey of layout problems. *Bulletin of the EATCS*, (105):177–201, 2011.

[23] Satish Rao and Andréa W. Richa. New approximation techniques for some ordering problems. In *Proc. of SODA*, pages 211–218, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[24] Habib Rostami and Jafar Habibi. Minimum linear arrangement of chord graphs. *Applied Mathematics and Computation*, 203(1):358 – 367, 2008.

[25] A. Setzer. The planar minimum linear arrangement problem is different from the minimum linear arrangement problem. *ArXiv e-prints*, September 2014.

[26] F. Shahrokhi, O. Sýkora, L. Székely, and I. Vrto. On bipartite drawings and the linear arrangement problem. *SIAM Journal on Computing*, 30(6):1773–1789, 2001.

[27] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of the ACM*, 29(3):623–641, July 1982.

[28] Jacobo Valdes, Robert E. Tarjan, and Eugene L. Lawler. The recognition of series parallel digraphs. In *Proc. of STOC*, pages 1–12, New York, NY, USA, 1979. ACM.

A Computing a minimal SP-Tree

In order to compute a minimal SP-tree for a given series-parallel graph G, the algorithm by Bodlaender and de Fluiter [5] can be used. It is a parallel algorithm that has a running time of $O(\log |E|)$ on a concurrent random-access machine (CRCW PRAM) using $O(|E|)$ processors for an input graph $G = (V, E)$. Thus it can be simulated in time $O(|E| \log |E|)$ by a sequential algorithm. Furthermore, it has a runtime of $O(\log |E| \log^* |E|)$ on an exclusive read exclusive write parallel random-access machine (EREW PRAM) using $|E|$ processors.

However, their definition (which we call Def. A) of an SP-tree slightly differs from ours (which we call Def. B): According to Def. A, each leaf in an SP-tree only represents a single edge, whereas in Def. B, leaves may represent line-graphs with $k \geq 1$ edges. However, it is not difficult to transform an SP-tree according to Def. A to an SP-tree according to Def. B: Simply transform each subtree induced by an S-node at the second lowest level and its children to an L-node $L(k)$, with k being the number of children of the S-node in the original SP-tree, and all remaining leaves that are neither P-nodes nor S-nodes to L-nodes $L(1)$.

Note that since each parallel or series composition has at least two input graphs, any non-leaf in a SP-tree has at least two children. Since each leaf in a SP-tree according to Def. A corresponds to an edge in G, this implies that the size of the minimal SP-tree returned by the algorithm of Bodlaender and de Fluiter is at most $2|E|$ and that its height is at most $O(\log |E|)$. For the sequential case, the SP-tree transformation can obviously be performed in time linear in the size of the SP-tree that it is performed on. Thus, determining a minimal SP-tree according to Def. B is possible in time $O(|E| \log |E|)$ in this case. For the parallel case (execution on an EREW PRAM with $\Theta(|E|)$ processors), it is not difficult to see that the SP-tree transformation can be implemented such that the runtime is linear in the height of the SP-tree, thus determining a minimal SP-tree according to Def. B is possible in time $O(\log |E| \log^* |E|)$ in the parallel case.

B Additional proofs

Lemma 12. *For each series-parallel graph G, there exists an S-decomposition SD. Besides, in any S-decomposition, each edge belongs to at most one S-path in SD.*

Proof. Let G be a series-parallel graph. In order to show that the S-decomposition is well-defined, i.e., we can find A-paths and S-paths as in Definition 11, we need to show that there is an A-path for any parallel component (an assumption we make when constructing the A-path through a non-"innermost" parallel component). We can do this via induction on the number of nodes of a parallel component. The induction hypothesis is: For any parallel component P, there is an A-path from the source of P to the sink of P.

By Definition 11, the induction hypothesis holds for all "innermost" components. Let P be a non-"innermost" parallel component and let the induction hypothesis hold for all parallel components of smaller size. In particular, the induction hypothesis holds for all (parallel) child components of (series) child components of P. Thus, we can construct the paths Q_i, $1 \leq i \leq m$ through all m child components of P. Since we choose Q_1 to be the A-path of P, the claim also holds for P and the induction is finished.

For the second claim, observe that by Definition 11, whenever we select a path Q to become an A-path or an S-path of a parallel component P, any section of this path has either been of no type before (which holds for the sections of the series child component S of P that are simple node sequences) or has been an A-path of a next smaller parallel component. This inductively implies that each edge can belong to at most one S-path in SD. □

Lemma 14. *Let $G = (V, E)$ be a series-parallel graph, and π_{OPT} be an (optimal) linear arrangement for G. Further, let $\mathcal{C}$ be the set of all (series or parallel) components of G. It holds:*

$$\sum_{C \in \mathcal{C}} A\text{-}COST_{\pi_{OPT}}(C) \leq 3 \cdot \sum_{C \in \mathcal{C}} E\text{-}COST_{\pi_{OPT}}(C).$$

Proof. Note that the only difference between the definition of the exclusive cost and the amortized cost is the addend ($\sum_{\{x,y\} \in E_C} length_S(x, y)$) in the definition of the amortized cost of series components. Denote by σ the sum of all these addends for all series components in $\mathcal{C}$. Then, by Definition 13: $\sum_{C \in \mathcal{C}} A\text{-}COST_{\pi_{OPT}}(C) = \sum_{C \in \mathcal{C}} E\text{-}COST_{\pi_{OPT}}(C) + \sigma$. Thus the claim of the lemma is equivalent to $\sigma \leq 2 \cdot \sum_{C \in \mathcal{C}} E\text{-}COST_{\pi_{OPT}}(C)$.

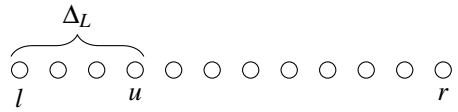
By Observation 10, this boils down to that σ is at most twice the total optimal cost. We prove the claim by showing that the length of each edge $e \in E$ is taken into account at most two times in σ. Since the length of each edge is contained exactly once in the total optimal cost (by the definition of the total cost), this yields the claim. More specifically, we show that there are at most two distinct occasions, at which the length of an edge e is considered for a calculation of the second addend.

Consider a fixed edge $e \in E$. We now identify the only possible cases where the length of e w.r.t. π_{OPT} (possibly restricted to some component) is added to σ. The first of the two cases in which this is possible is that e is contained in a simple node sequence that is a child component of a series component S. This can occur only once. The second case is that e is contained in an S-path of a parallel component P and taken into account for the calculation of the amortized cost of S, the series component whose child is P. As follows from Lemma 12, each edge can occur in one S-path only. Thus, also this case can occur only once. □

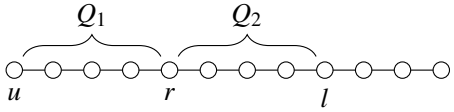
Lemma 19. *For any simple node sequence L in a series-parallel graph G in an optimal arrangement π_{OPT}, it holds:*

$$A\text{-}COST_{\pi_{OPT}}(L) \geq |L| - 1 + \Delta_L.$$

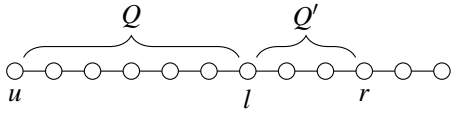
Proof. Let L be a simple node sequence in a series-parallel graph G and let π_{OPT} be an optimal linear arrangement of G. In the following, we consider the arrangement π_{OPT} restricted to L. W.l.o.g. let u be the node that defines Δ_L in $\pi_{OPT}(L)$, i.e., the terminal node with a minimum number of nodes to its right or left in $\pi_{OPT}(L)$. Furthermore, w.l.o.g. assume that u is on the left half of L in $\pi_{OPT}(L)$. Denote the leftmost node in $\pi_{OPT}(L)$ by l and the rightmost node in $\pi_{OPT}(L)$ by r. An illustration of the given situation can be found in Figure 3(a). There is a simple path Q from u to l in L. This path may or may not contain r.



(a) Schematic representation of $\pi_{OPT}(L)$ (edges are not shown).



(b) Example of the original simple node sequence L in G as in the first case.



(c) Example of the original simple node sequence L in G as in the second case.

Figure 3: Illustration of the notions of Lemma 19.

We first consider the case that Q contains r. In this case, split Q into two paths Q_1 and Q_2, the first ranging from u to r, the second ranging from r to l. This is illustrated in Figure 3(b). The first path has a length of at least $|L|/2 \geq \Delta_L$ in $\pi_{OPT}(L)$, whereas the second path has a length of at least $|L| - 1$ in $\pi_{OPT}(L)$. Since any simple node sequence consists of a single simple path only, no edge is counted twice in this calculation. All in all, this completes the proof for this case.

If Q does not contain r, then there is also a second path Q' from l to r whose nodes are disjoint from those in Q (except for l). This is illustrated in Figure 3(c). The length of Q' in $\pi_{OPT}(L)$ is at least $|L| - 1$, whereas the length of Q in $\pi_{OPT}(L)$ is at least Δ_L. Together, this yields the claim in this case, too. □

Lemma 20. *For any series component S in a series-parallel graph G with $m \geq 2$ child components $P_1, \ldots, P_m$ in an optimal arrangement π_{OPT}, it holds:*

$$A\text{-}COST_{\pi_{OPT}}(S) \geq \frac{1}{2}\left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}|\right) + 1 + \sum_{i=1}^{m} \Delta_{P_i} - \Delta_S.$$

Proof. Let S be a series component with $m \geq 2$ child components $P_1, \ldots, P_m$ in a series-parallel graph G and let π_{OPT} be an optimal linear arrangement of G. In the following, we consider the arrangement π_{OPT} restricted to S.

The amortized cost of S is defined as the sum of two values: the exclusive cost of S and a sum over certain edge lengths (the latter sum we denote by σ). We now take a closer look at how the exclusive cost arise and which edge lengths contribute to the value of σ.

The additional term σ is the sum of lengths (restricted to S) of all edges in the set E_S that contains all edges of simple node sequences that are child components of S and all edges of S-paths of the child components of S that are parallel components.

The exclusive cost of S arise in the following way: An edge e has a greater length in the arrangement π_{OPT} restricted to S than in the arrangement π_{OPT} restricted to the child component P_i of S that e is from. We call the difference in the length of e the *additional stretching* of e. Then, the sum of additional stretchings of all edges is the exclusive cost of S. Thus, for the exclusive cost we need to determine a lower bound on the sum of additional stretchings due to S.

We now give a lower bound on σ and on the sum of additional stretchings individually. The sum of these two bounds yields a lower bound of the amortized cost of S.

Let V_S be the set of nodes that contains all the endpoints of edges in E_S and denote the interval spanning the set V_S in $\pi_{OPT}(S)$ by I (see Def. 18). Since there exists a simple path from the leftmost node in V_S (w.r.t. $\pi_{OPT}(S)$) to the rightmost node in V_S consisting of edges from E_S only, σ is at least $|I| - 1$.

In order to prove the lower bound on the sum of the additional stretchings, we consider nodes whose positions are to the left of those in I (the *part left of I*) and nodes whose positions are to the right of those in I (the *part right of I*) individually. We show that for any node in the part left of I, except for nodes in a sequence of nodes that starts with the leftmost node, at least one edge is stretched by an amount of one due to this node. The analog can be shown for the part right of I. This yields a lower bound on the sum of additional stretchings linear in the number of these nodes.

Denote the leftmost node of $\pi_{OPT}(S)$ by a. In the following, assume $a \notin V_S$. Later on, we will consider the case $a \in V_S$ as well. Let A be the maximal sequence of consecutive (w.r.t. $\pi_{OPT}(S)$) nodes that starts with a and contains only nodes that belong to the same child component P_A of S as a, but not to V_S. Note that P_A may be either a simple node sequence or a parallel component. Similarly, assume that the rightmost node of $\pi_{OPT}(S)$ is not in V_S (again, we will consider the other case later on). Let B be the analogous sequence starting with the rightmost node and let P_B be defined analogously (i.e., B must not include any node from V_S). The situation is depicted in Figure 4.

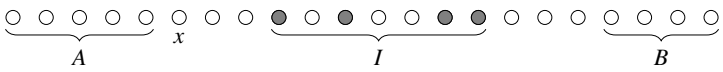


Figure 4: Example of π_{OPT} restricted to S in the proof of Lemma 20. Gray nodes are from V_S.

From the rightmost node in A, there exists a path to a closest node in V_S, which is stretched by all the nodes between A and I that do not belong to P_A. Denote the set of these nodes by O_L. Let x be the right neighbor of the rightmost node in A. If $x \notin V_S$, there exists a path from x to a closest node in V_S (since the terminals of the component P_x that x belongs to must definitely be in V_S), which is stretched by, among others, the nodes from P_A that lie between x and I. Denote the set of these nodes by O_A. If, however, $x \in V_S$, we have $O_A = \emptyset$ (and the following holds accordingly). All in all, since we identified the stretchings of disjoint paths, we have an additional stretching of $|O_A| + |O_L|$ induced by the nodes in the part left of I. The right part (starting from B) is analog, so we also have an additional stretching of $|O_B| + |O_R|$, where O_B and O_R are defined analogous to O_A and O_L, respectively.

Summing up the amortized costs we collected so far, yields $|I| - 1 + |O_A| + |O_L| + |O_B| + |O_R|$. Since $I \cup O_A \cup O_L \cup O_B \cup O_R$ is the set of all nodes that are in S but not in A or B, this sum is equal to $|S| - 1 - |A| - |B|$. It holds that $|S| = \sum_{i=1}^{m} |P_i^{\ominus}| + 1$. Thus, we have:

$$A\text{-}COST_{\pi_{OPT}}(S) \geq \sum_{i=1}^{m} |P_i^{\ominus}| - |A| - |B|.$$

Next, we apply some upper bounds for the sizes of A and B in order to rewrite this inequation as in the original claim of the lemma.

W.l.o.g. assume that $|A| \geq |B|$. This yields $|B| \leq \Delta_S$. Furthermore, it holds that $|A| \leq |P_A^{\ominus}| - \Delta_{P_A}$. To understand this, assume for contradiction $|A| > |P_A^{\ominus}| - \Delta_{P_A}$. Then, less than $|P_A| - (|P_A^{\ominus}| - \Delta_{P_A}) = \Delta_{P_A} + 1$ nodes of P_A would have been placed to the right of A in $\pi_{OPT}(P_A)$. Since the terminal nodes of P_A are in V_S and thus lie to the right of A, this would imply that less than $\Delta_{P_A} - 1$ nodes from P_A lie to the right of the rightmost terminal node of P_A in $\pi_{OPT}(P_A)$. However, this would contradict to the minimality of Δ_{P_A}.

Recall that we assumed that the leftmost and rightmost nodes are not in V_S. This is due to the fact that if the leftmost node is in V_S, the sequence A does not exist and we cannot define O_A and O_L as above. The analogous is true for O_B and O_R in case that the rightmost node is in V_S. However, with a case distinction, we can show that the above stated bounds hold in any case. First, assume that the rightmost node is in V_S, but the leftmost node is not. In this case, $I \cup O_A \cup O_L$ contains all nodes that are in S but not in A. Thus, $|I| - 1 + |O_A| + |O_L|$ is exactly $|S| - 1 - |A|$. For $|B| := 0$, we can also write this as $|S| - 1 - |A| - |B|$. Note that $|B| \leq \Delta_S$ trivially holds in this case. Second, assume that the leftmost node is in V_S. This means that A is empty, which implies (since we assumed, without loss of generality, $|A| \geq |B|$) that B is empty, too. In this case, $|I| - 1 = |S| - 1 = |S| - 1 - |A| - |B|$ if we set $|A| := |B| := 0$. Furthermore, it is true that $|A| \leq |P_A^{\ominus}| - \Delta_{P_A}$ for an arbitrary component

$P_A \in S$. Thus, regardless of which case we are in, we can continue:

$$\begin{aligned}
A\text{-}COST_{\pi_{OPT}}(S) &\geq \sum_{P_i \in S} |P_i^{\ominus}| - |A| - |B| \\
&\geq \sum_{P_i \in S} |P_i^{\ominus}| - (|P_A^{\ominus}| - \Delta_{P_A}) - \Delta_S \\
&= \sum_{P_i \in S \setminus \{P_A\}} |P_i^{\ominus}| + \Delta_{P_A} - \Delta_S \\
&\geq \frac{1}{2} \sum_{P_i \in S \setminus \{P_A\}} |P_i^{\ominus}| + \sum_{P_i \in S \setminus \{P_A\}} \Delta_{P_i} + \Delta_{P_A} - \Delta_S \\
&\geq \frac{1}{2} \left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}| \right) + \sum_{i=1}^{m} \Delta_{P_i} - \Delta_S
\end{aligned}$$

in which we use that $\frac{1}{2}|P_i^{\ominus}| \geq \Delta_{P_i}$ for any i (c.f. Lemma 17) and that $|P_A^{\ominus}| \leq \max_i |P_i^{\ominus}|$. □

Lemma 21. *For any parallel component P in a series-parallel graph G with $m \geq 2$ child components $S_1, \ldots, S_m$, in an optimal arrangement π_{OPT}, it holds:*

$$A\text{-}COST_{\pi_{OPT}}(P) \geq \frac{1}{2} \left(\sum_{i=1}^{m} |S_i^{-}| - \max_i |S_i^{-}| \right) + \sum_{i=1}^{m} \Delta_{S_i} - \Delta_P.$$

Proof. Let P be a parallel component with $m \geq 2$ child components $S_1, \ldots, S_m$ in a series-parallel graph G and let π_{OPT} be an optimal linear arrangement of G. Note that for parallel components, the amortized cost is equal to the exclusive cost. Thus, it suffices to provide a lower bound on the exclusive cost of P. Similar to the proof of Lemma 20, this is done by determining a lower bound on the sum of additional stretchings of all edges in P.

In the following, we consider the arrangement π_{OPT} restricted to P. Denote the terminals of P by u and v. Without loss of generality, assume that u is placed to the left of v. Let a be the leftmost node in $\pi_{OPT}(P)$ and b be the rightmost node in $\pi_{OPT}(P)$. For now, assume $a \neq u$ and $b \neq v$ (we will handle the other cases later on). Denote by S_A the (series) child component of P that a is from and denote by S_B the (series) child component of P that b is from. Let A be the maximal sequence of consecutive nodes in $\pi_{OPT}(P)$ that starts with a and only contains nodes that belong to S_A but do not equal u. Similarly, define a rightmost sequence B that is the analogon to A (i.e., a maximal sequence of consecutive nodes in $\pi_{OPT}(P)$ that start with b and whose nodes belong to S_B only, but v must not belong to B). It is possible that $S_A = S_B$, which does not pose a problem in the following proof. The situation is depicted in Figure 5.

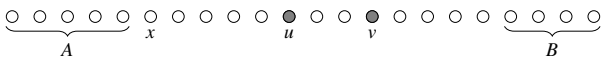


Figure 5: Example of π_{OPT} restricted to P in the proof of Lemma 21.

We now consider the nodes to the left of v and to the right of v individually. For any node whose position is left of v, except for u and the nodes belonging to A, we show that there is an additional stretching of at least one due to this node. The analogous holds for the nodes whose positions are right of v, except for those belonging to B. This yields an additional stretching of at least $|P| - 2 - |A| - |B|$, which we will use to establish the lower bound in the claim of the lemma.

Since S_A is a series component with a path from u to v, there is a path from the rightmost node of A to v consisting of edges from S_A only. Let E_L be the set of edges on this path. The additional stretching of any edge $e \in E_L$ is at least the number of nodes that lie between A and v and that are from different child components of P (other than S_A), since the edges in E_L span these nodes. Therefore, the sum of all additional stretchings of the edges in E_L is at least the number of nodes in $\pi_{OPT}(P)$ that lie to the left of v and do not belong to S_A. We denote this number by $|O_L|$.

Now we consider the node x that lies to the right of the rightmost node of A in $\pi_{OPT}(P)$. By definition of A, either x does not belong to S_A or $x = a$. In the first case, denote the series component that x belongs to by S_x. Otherwise, let S_x be an arbitrary child component of P such that $S_x \neq S_A$. There also exists a path from x to v consisting of edges from S_x only. Let E_x be the set of edges on this path. Since all the nodes of S_A that lie between x and v (including x) contribute to the additional stretchings of the edges in E_x, the sum of additional stretchings due to the edges in E_x is at least the number of these nodes, denoted by $|O_A|$.

Considering only the section from A to v, we have identified a sum of additional stretchings of at least $|O_L| + |O_A|$. Notice that only nodes that lie to the left of v are counted for this argument. Analogously, we can argue about the part from B to v and find a sum of additional stretchings of at least $|O_R| + |O_B|$, where $|O_R|$ is the number of nodes between v and B that do not belong to S_B, and $|O_B|$ is the number of nodes between v and B that do belong to S_B. Here, we only need to count nodes that lie to the right of v. In total, since $O_L \cup O_R \cup O_A \cup O_B$ is the set of all nodes in P except for the two terminal nodes and those in A or B, the sum of additional stretchings it at least the sum of the sizes of all (series) child components (not counting the two terminal nodes) minus the size of A and the size of B. That is:

$$A\text{-}COST_{\pi_{OPT}}(P) \geq \sum_{i=1}^{m} |S_i^-| - |A| - |B|.$$

W.l.o.g., assume $|A| \geq |B|$. By definition of A and B (which must contain neither u nor v), this yields $\Delta_P \geq |B|$. Furthermore, $|A| \leq |S_A^-| - \Delta_{S_A}$. To understand this, assume for contradiction $|A| > |S_A^-| - \Delta_{S_A}$. Then, less than $|S_A| - (|S_A^-| - \Delta_{S_A}) = \Delta_{S_A} + 2$ nodes of S_A would have been placed to the right of A in $\pi_{OPT}(S_A)$. Since u and v lie to the right of A and belong to S_A, this would imply that less than Δ_{S_A} nodes from S_A lie to the right of the rightmost node among u and v in $\pi_{OPT}(S_A)$. However, this would contradict to the minimality of Δ_{S_A}.

Recall that we assumed $a \neq u$ and $b \neq v$. We now handle the remaining cases. First of all, consider $a \neq u$ and $b = v$. In this case, the sequence B does not exist and we can define $O_B := O_R := \emptyset$. Then, $O_L \cup O_A$ is the set of all nodes in P except for the two terminal nodes and those in A. Thus, if we define $|B| := 0$, $A\text{-}COST_{\pi_{OPT}}(P) \geq \sum_{i=1}^{m} |S_i^-| - |A| - |B|$ holds, too. Trivially, also $|B| \leq \Delta_P$ holds.

In the case $a = u$, the sequence A is empty. Since we assumed, without loss of generality, $|A| \geq |B|$, this case can only hold if $b = v$, too. Let S_A and S_x be two arbitrary distinct child components of P. There exists a path from u to v through each of the two components. Denote the edges on this path by E_L and E_x. Similar to the proof in the case $a \neq u$, the edges in E_L are stretched by all nodes from P, except for u, v, and those in S_A, and the edges in E_x are stretched by all nodes from S_A. All in all, we do not need to count any node twice to find a stretching of $A\text{-}COST_{\pi_{OPT}}(P) \geq \sum_{i=1}^{m} |S_i^-|$ in this case, too. To proceed as in the other cases, we define $|A| := |B| := 0$, such that $A\text{-}COST_{\pi_{OPT}}(P) \geq \sum_{i=1}^{m} |S_i^-| - |A| - |B|$ and $|B| \leq \Delta_P$ holds.

All in all, we get:

$$\begin{aligned}
A\text{-}COST_{\pi_{OPT}}(P) &\geq \sum_{S_i \in P} |S_i^-| - |A| - |B| \\
&\geq \sum_{S_i \in P} |S_i^-| - |S_A^-| + \Delta_{S_A} - \Delta_P \\
&= \sum_{S_i \in P \setminus \{S_A\}} |S_i^-| + \Delta_{S_A} - \Delta_P \\
&\geq \frac{1}{2} \sum_{S_i \in P \setminus \{S_A\}} |S_i^-| + \sum_{S_i \in P \setminus \{S_A\}} \Delta_{S_i} + \Delta_{S_A} - \Delta_P \\
&\geq \frac{1}{2} \left(\sum_{i=1}^{m} |S_i^-| - \max_i |S_i^-| \right) + \sum_{i=1}^{m} \Delta_{S_i} - \Delta_P
\end{aligned}$$

In the fourth step we use $\frac{1}{2}|S_i^-| \geq \Delta_{S_i}$, which is due to Lemma 17. The fifth step holds due to $|S_A| \leq \max_i |S_i|$. □

Corollary 22. *Let $G = (V, E)$ be an arbitrary series-parallel graph and π_{OPT} an optimal arrangement of G. Further, denote the total cost of π_{OPT} by $COST_{\pi_{OPT}}(G)$, the set of simple node sequences in G by L_G, the set of parallel components by P_G, the set of series components by S_G. Then, it holds:*

$$\begin{aligned}
7 \cdot COST_{\pi_{OPT}}(G) \geq \sum_{L \in L_G} 2 \cdot (|L| - 1) + \sum_{P \in P_G} \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) \\
+ \sum_{S \in S_G} \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right).
\end{aligned}$$

Proof. By Observation 10, $COST_{\pi_{OPT}}(G)$ is the sum of all exclusive costs of any series or parallel component C, i.e., $COST_{\pi_{OPT}}(G) = \sum_{C \in \mathcal{C}} E\text{-}COST_{\pi_{OPT}}(C)$, where $\mathcal{C}$ is the set of all series or parallel components of G. Recall the relationship $3 \cdot \sum_{C \in \mathcal{C}} E\text{-}COST_{\pi_{OPT}}(C) \geq \sum_{C \in \mathcal{C}} A\text{-}COST_{\pi_{OPT}}(C)$ from Lemma 14. This gives us:

$$\begin{aligned}
&6 \cdot COST_{\pi_{OPT}}(G) \geq 2 \cdot \left(\sum_{C \in \mathcal{C}} A\text{-}COST_{\pi_{OPT}}(C) \right) \\
&= 2 \cdot \left(\sum_{L \in L_G} A\text{-}COST_{\pi_{OPT}}(L) + \sum_{P \in P_G} A\text{-}COST_{\pi_{OPT}}(P) + \sum_{S \in S_G} A\text{-}COST_{\pi_{OPT}}(S) \right) \\
&\geq 2 \cdot \Bigg(\sum_{L \in L_G} (|L| - 1 + \Delta_L) \\
&\quad + \sum_{P \in P_G} \Big(\frac{1}{2} \Big(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \Big) + \sum_{S_i \in P} \Delta_{S_i} - \Delta_P \Big) \\
&\quad + \sum_{S \in S_G} \Big(\frac{1}{2} \Big(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \Big) + \sum_{P_i \in S} \Delta_{P_i} - \Delta_S \Big) \Bigg)
\end{aligned}$$

in which we apply Lemma 19, Lemma 20, and Lemma 21. We now take a closer look at the relationship between the different sets. Denote by O the outmost (series or parallel) component. For each simple node sequence $L \in L_G \setminus \{O\}$, there is a series or parallel component which has L as a child. Furthermore, for each series component $S \in S_G \setminus \{O\}$, there is a parallel component which has S as a child component. Similarly, for each parallel component $P \in P_G \setminus \{O\}$, there is a series component which has P as a child component. Since each (parallel or series) component adds a value of Δ_{C_i} for each of its child components C_i to the cost, the subtracted Δ_{C_i} values for all components C_i (except for O) are canceled out by these. All that remains is a subtrahend of Δ_O for the outmost component. Note that $\Delta_O \leq |V|$ and $COST_{\pi_{OPT}}(G) \geq |V|$, which gives us $COST_{\pi_{OPT}}(G) \geq \Delta_O$. Thus, the seventh addition of $COST_{\pi_{OPT}}(G)$ in the original claim cancels out the Δ_O to be subtracted, which concludes the proof. □

Corollary 23. *Let $G = (V, E)$ be an arbitrary series-parallel graph and let π_{ALG} be an arrangement of G computed by the SPGAA. Furthermore, denote the total cost of π_{ALG} by $COST_{\pi_{ALG}}(G)$, the set of simple node sequences in G by L_G, the set of parallel components by P_G, the set of series components by S_G. Then, it holds:*

$$COST_{\pi_{ALG}}(G) \leq \sum_{L \in L_G} 2 \cdot (|L| - 1) + \sum_{P \in P_G} 2D^2 \cdot \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) + \sum_{S \in S_G} 2D \cdot \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right).$$

In order to prove Corollary 23, we first bound the exclusive cost of simple node sequences, parallel components and series components individually. This is given by the following three lemmata.

We start with a bound on the exclusive cost of simple node sequences:

Lemma 26. *For any simple node sequence L in an arrangement π_{ALG} computed by the SPGAA, it holds:*

$$E\text{-}COST_{\pi_{ALG}}(L) \leq 2 \cdot (|L| - 1).$$

Proof. Consider the way the SPGAA arranges a simple node sequence (c.f. Subsection 3.1). Observe that the length of each edge in $\pi_{ALG}(L)$ is at most two. Since there are exactly $|L| - 1$ edges in a simple node sequence, this implies the claim. □

Next we upper bound the exclusive cost of parallel components as arranged by the SPGAA:

Lemma 27. *For any parallel component P in a series-parallel graph G with $m \geq 2$ child components $S_1, \ldots, S_m$, in an arrangement π_{ALG} computed by the SPGAA, it holds:*

$$E\text{-}COST_{\pi_{ALG}}(P) \leq 2 \cdot D^2 \cdot \left(\sum_{i=1}^{m} |S_i^-| - \max_i |S_i^-| \right).$$

Proof. Let P be a parallel component in a series-parallel graph G with $m \geq 2$ child components $S_1, \ldots, S_m$, in an arrangement π_{ALG} computed by the SPGAA.

Recall that for parallel components, the exclusive cost arise due to the increase of distances between connected nodes in the current recursion level in comparison to the previous recursion level. From the perspective of a single child component S_i, all that changes in the current recursion step is that the two terminal nodes of S_i are moved by a certain distance to the left (c.f. the algorithm description

in Subsection 3.2). Therefore, only the lengths of edges whose one endpoint is a terminal node can increase at all (since the remaining edges keep their previous lengths). For the whole component, this can be at most $2D$ edges (since each of the terminals is connected with at most D other nodes from S_i).

The extent to which these edge lengths increase depends on the position of S_i in π_{ALG}: For the leftmost component (w.l.o.g., let this be S_1), this is zero. For the second component (w.l.o.g., let this be S_2, accordingly), it is exactly $|S_1^-|$. The third component (w.l.o.g., let this be S_3) has an increase of $|S_1^-| + |S_2^-|$ and so on. For the last component (which, according to the description of the SPGAA, is the biggest component and denoted by S_m), the extent to which the up to $2D$ edges to the terminal nodes are stretched is $\sum_{i=1}^{m-1} |S_i^-| = \sum_{i=1}^{m} |S_i^-| - \max_i |S_i^-|$.

All in all, we have that for any of the $m \leq D$ child components, at most $2D$ edges are stretched by an extent of at most $\sum_{i=1}^{m} |S_i^-| - \max_i |S_i^-|$, which completes the proof. □

The last missing piece is the upper bound of the exclusive cost of series components in arrangements of the SPGAA:

Lemma 28. *For any series component S in a series-parallel graph G with $m \geq 2$ child components $P_1, \ldots, P_m$, in an arrangement π_{ALG} computed by the SPGAA, it holds:*

$$E\text{-}COST_{\pi_{ALG}}(S) \leq 2 \cdot D \cdot \left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}| \right).$$

Proof. Let S be a series component in a series-parallel graph G with $m \geq 2$ child components $P_1, \ldots, P_m$, in an arrangement π_{ALG} computed by the SPGAA.

Again, the exclusive cost arise due to nodes that have a greater distance to other nodes when compared to the previous recursion level. We now identify the cases in which a distance is increased. First of all, observe that for a fixed child component P_i, any non-terminal node always keeps its distance to any other non-terminal node. Thus, we only need to identify the cases in which a terminal node increases its distance to other nodes and to what extent. Summing up these values yields the desired exclusive cost then.

For the purpose of finding out the values, it is helpful to consult Figure 2. Here, the dotted nodes are those that change their position (compared to the previous recursion level) and the lengths of the arrows (i.e., the number of nodes that they span) indicate the distance by which they are moved. We now just determine the lengths of the arrows for each of the dotted nodes one after another. Since each such node can be connected with at most D other nodes from its component, multiplying the length with D yields the exclusive cost caused by this node movement. We consider the case $1 < a < m$ first, with a defined as in the description of the SPGAA (see Subsection 3.3).

The first dotted node is the source of P_1 (which is also the source of S). Its distance to other nodes from P_1 increases by at most one.

The second dotted node is the sink of P_1 (which is also the source of P_2). It is moved to the right end of P_1 at which it is shifted by a distance of $|P_1^{\ominus}| - 1$. Analogously, the next dotted node (the sink of P_2) increases its distance to other nodes by $|P_2^{\ominus}| - 1$. The same holds for the sink of each child component P_j with $3 \leq j \leq a-2$: Each one is shifted by $|P_j^{\ominus}| - 1$. All in all, for the sinks of the child components from P_2 to P_{a-2}, we have a total increase in distances of up to $\sum_{i=1}^{a-2} (|P_i^{\ominus}| - 1)$.

The sink of P_{a-1} is shifted by a greater distance: It passes all other nodes of P_{a-1} and all nodes of P_j for $m \geq j \geq a+1$. That is, in total its increase in distances is at most $|P_{a-1}^{\ominus}| - 1 + \sum_{i=a+1}^{m} |P_i^{\ominus}|$ (recall that we need to add the non-dashed nodes only).

The next dashed node we have not considered so far is the sink of P_m. It is moved to the second position (from left to right) and thereby passes all other nodes of P_m (except for the source), and all

nodes of P_j for $a-1 \geq j \geq 1$. This makes up a total distance of $|P_m^{\ominus}| - 1 + \sum_{i=1}^{a-1} |P_i^{\ominus}| - 1$ (the last minus one is due to the source of P_1 whose movement has already been counted).

The following dotted nodes, starting with the sink of P_{m-1} and ending with the sink of P_{a+1} move by the size of their component (without the two terminals) each, which makes a total distance of $\sum_{i=a+1}^{m-1} (|P_i^{\ominus}| - 1)$.

Last, the sink of P_a is moved by one.

Summing over all these values yields the following upper bound on the exclusive cost:

$$\begin{aligned} \textit{E-COST}_{\pi_{ALG}}(S) \leq & 1 + \left(\sum_{i=1}^{a-2} (|P_i^{\ominus}| - 1) \right) + \left(|P_{a-1}^{\ominus}| - 1 + \sum_{i=a+1}^{m} |P_i^{\ominus}| \right) \\ & + \left(|P_m^{\ominus}| - 1 + \sum_{i=1}^{a-1} |P_i^{\ominus}| - 1 \right) + \left(\sum_{i=a+1}^{m-1} (|P_i^{\ominus}| - 1) \right) + 1 \\ = & 2 \cdot \left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}| \right) + 2 - m \leq 2 \cdot \left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}| \right) \end{aligned}$$

in which we use that $m \geq 2$. The computed value is the sum of all distances that each terminal node has moved and thus possibly increased the distance to at most D other nodes in its own component. Therefore, this completes the proof in this case.

For the case $a = 1$, the procedure is similar to the previous case. Actually, the increase in distance caused by the movement of the dotted nodes from the sink of P_{m-1} up to the sink of P_2 is exactly the same. Its value is $\sum_{i=2}^{m-1} (|P_i^{\ominus}| - 1)$. Additionally, we have to take into account the movement of the sink of P_m (by a distance of $|P_m^{\ominus}| - 1$) and of the source of P_1 (which passes the nodes of all other child components, which are $\sum_{i=2}^{m} |P_i^{\ominus}| + 1$ in total). If we assume that the source of P_1 is moved first, the sink of P_1 does not need to be moved any more. All in all, we have a total possible increase of distances (and, accordingly, an exclusive cost) of at most

$$\textit{E-COST}_{\pi_{ALG}}(P) \leq 2 \cdot \sum_{i=2}^{m} |P_i^{\ominus}| + \sum_{i=2}^{m} (-1) + 1 \leq 2 \cdot \left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}| \right),$$

which, following the argument from the previous case, completes the proof in this case, too.

For the case $a = m$, the movement of all of the dotted nodes up to the sink of $P_{a-2} = P_{m-2}$ is exactly the same. For these we have a total increase in distances of $1 + \sum_{i=1}^{m-2} (|P_i^{\ominus}| - 1)$. Next, we have the sink of P_{m-1} which is shifted by a distance of $|P_{m-1}^{\ominus}| - 1$. Last, the sink of P_m is shifted by a distance of $1 + \sum_{i=1}^{m-1} |P_i^{\ominus}| - 1$ (the last minus one originates from the assumption that we shift the source of P_1 first and thus the sink of P_m does not need to pass it any more). All in all, we have a total possible increase of distances (and, accordingly, an exclusive cost) of at most

$$\textit{E-COST}_{\pi_{ALG}}(P) \leq 2 \cdot \sum_{i=1}^{m-1} |P_i^{\ominus}| = 2 \cdot \left(\sum_{i=1}^{m} |P_i^{\ominus}| - \max_i |P_i^{\ominus}| \right).$$

With the same argument as in the previous two cases, this completes the proof in this case, too. □

These three lemmata (together with Observation 10) enable us to prove Corollary 23:

Corollary 23. *Let $G = (V, E)$ be an arbitrary series-parallel graph and let π_{ALG} be an arrangement of G computed by the SPGAA. Furthermore, denote the total cost of π_{ALG} by $COST_{\pi_{ALG}}(G)$, the set of*

simple node sequences in G by L_G, the set of parallel components by P_G, the set of series components by S_G. Then, it holds:

$$COST_{\pi_{ALG}}(G) \leq \sum_{L \in L_G} 2 \cdot (|L| - 1) + \sum_{P \in P_G} 2D^2 \cdot \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) + \sum_{S \in S_G} 2D \cdot \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right).$$

Proof. Let $\mathcal{C}$ be the set of all components of G. Recall that $COST_{\pi_{ALG}}(G) = \sum_{C \in \mathcal{C}} E\text{-}COST_{\pi_{ALG}}(C)$ (by Observation 10). Applying Lemma 26, Lemma 27, and Lemma 28 yields:

$$\begin{aligned} COST_{\pi_{ALG}}(G) &= \sum_{L \in L_G} E\text{-}COST_{\pi_{ALG}}(L) + \sum_{P \in P_G} E\text{-}COST_{\pi_{ALG}}(P) + \sum_{S \in S_G} E\text{-}COST_{\pi_{ALG}}(S) \\ &\leq \sum_{L \in L_G} 2 \cdot (|L| - 1) \\ &\quad + \sum_{P \in P_G} 2 \cdot D^2 \cdot \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) \\ &\quad + \sum_{S \in S_G} 2 \cdot D \cdot \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right), \end{aligned}$$

which completes the proof. □

Theorem 24. *For a series-parallel graph G, let π_{ALG} be the linear arrangement of G computed by the SPGAA, and let π_{OPT} be an optimal linear arrangement of G. It holds:*

$$COST_{\pi_{ALG}}(G) \leq 14 \cdot D^2 \cdot COST_{\pi_{OPT}}(G).$$

Proof. Let G be a series-parallel graph, let π_{ALG} be the linear arrangement of G computed by the SPGAA, and let π_{OPT} be an optimal linear arrangement of G. Corollary 22 implies:

$$\begin{aligned} 14 \cdot D^2 \cdot COST_{\pi_{OPT}}(G) \geq & 2D^2 \left(\sum_{L \in L_G} 2(|L| - 1) + \sum_{P \in P_G} \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) \right. \\ & \left. + \sum_{S \in S_G} \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right) \right). \end{aligned}$$

On the other hand, from Corollary 23, we can derive:

$$\begin{aligned} COST_{\pi_{ALG}}(G) \leq & 2 \cdot D^2 \cdot \left(\sum_{L \in L_G} (|L| - 1) + \sum_{P \in P_G} \left(\sum_{S_i \in P} |S_i^-| - \max_{S_i \in P} |S_i^-| \right) \right. \\ & \left. + \sum_{S \in S_G} \left(\sum_{P_i \in S} |P_i^{\ominus}| - \max_{P_i \in S} |P_i^{\ominus}| \right) \right) \end{aligned}$$

Together, these two equations yield the claim and thus complete the proof of this theorem. □

Theorem 25. *On a series-parallel graph $G = (V, E)$, the SPGAA has a runtime of $O(|E|)$ if a minimal SP-tree of G is given as an input, and a runtime of $O(|E| \log |E|)$ otherwise.*

Proof. For this proof, we assume that the minimal SP-tree T used to define the series and parallel components of a graph $G = (V, E)$ is given as an input to the algorithm. If T is not given, the algorithm described in Appendix A can be used to compute it in time $O(|E| \log |E|)$ (thus the runtime bound is slightly higher in that case).

Let s be the sum of the sizes (w.r.t. the number of nodes in it) of all simple node sequences and let c be the total number of (series or parallel) components in G. We now determine upper bounds on s and c and then bound the runtime depending on their values.

First of all, notice that each edge in G can be part of at most one simple node sequence. Furthermore, any simple node sequence has at most twice as many nodes as edges. Thus, s can be at most $2|E|$ in total.

Second, note that the number of edges in any result of a series or parallel composition operation OP is strictly greater than the number of edges in each input to OP. Thus, c is upper bounded by $|E|$.

It is possible to implement the SPGAA in the following way: The algorithm traverses T from bottom to top and constructs an internal representation of the series-parallel graph as well as its linear arrangement during this traversal. At each node u of T, the arrangement of the subgraph corresponding to u is determined (basically by ordering the child components of u in the right way). This way, it is obvious that the runtime of the algorithm is upper bounded by the number of nodes of T (which is c) plus the time spent at each edge of T. Therefore, we now determine the latter.

At the lowest recursion level, the algorithm arranges the simple node sequences. For each simple node sequence S, determining the order of the nodes in S is possible in time linear in $|S|$. Thus, for arranging all simple node sequences, we need time linear in s only.

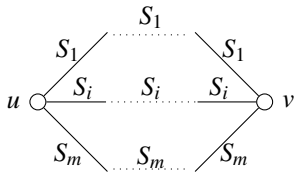
Throughout the higher recursion levels, the algorithm determines the order of the child components of all series or parallel components that are not simple node sequences. For any such component C, this is possible in time linear in the number of child components of C. Since each component can be a child component of at most one other component, this implies that the higher recursion levels require time linear in c in total.

All in all, the number of nodes in T is upper bounded by $O(|E|)$ and the total time spent at each node of T is upper bounded by $O(s + c) = O(|E|)$. Thus, the algorithm has a total runtime of $O(|E|)$. □

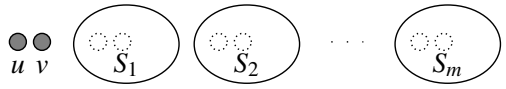
C Additional figures

(a) $SNS(6)$. (b) The resulting arrangement of $SNS(6)$.

Figure 6: SPGAA arrangement of a simple node sequence.

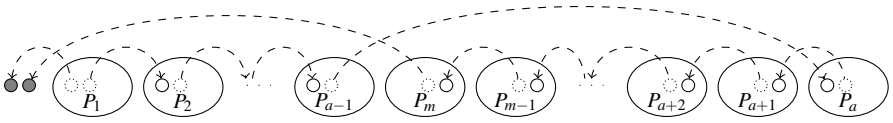


(a) P in the original graph.

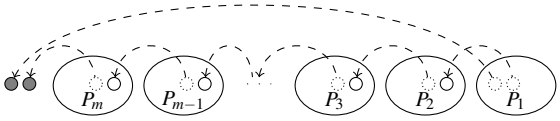


(b) The resulting arrangement of P. Dashed nodes represent former positions of u and v.

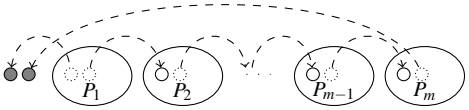
Figure 7: The order in which the SPGAA arranges a parallel component P with source u and sink v consisting of m series child components $S_1, \ldots, S_m$. Note that we assume S_m to be a biggest child component.



(a) $1 < a < m$

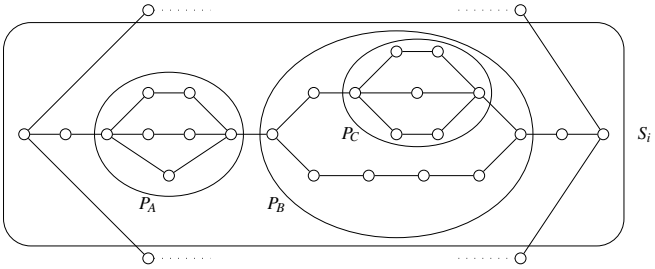


(b) $a = 1$

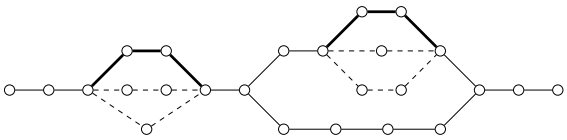


(c) $a = m$

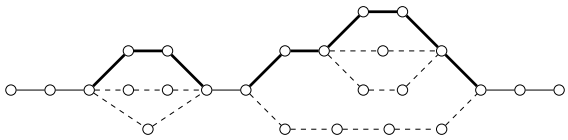
Figure 8: The order in which the SPGAA arranges a series component consisting of m child components. P_a is a biggest child component. Dashed nodes indicate the position at which a node would be placed according to the previous recursion level. Dashed arrows indicate the change in position at the current recursion level.



(a) Extract of an example graph with labels of some components.



(b) Selection of one A-path and two S-paths for each of the two components P_A and P_C.



(c) Subsequent selection of an A-path and an S-path of P_B.

Figure 9: Step-by-step illustration of an S-decomposition. After the first figure, only S_i is shown. Thick lines describe an A-path, whereas dashed lines describe S-paths.