Efficient reassembling of graphs, part 1: the linear case

Assaf Kfoury · Saber Mirzaei

Abstract The reassembling of a simple connected graph \( G = (V, E) \) is an abstraction of a problem arising in earlier studies of network analysis. Its simplest formulation is in two steps:

1. We cut every edge of \( G \) into two halves, thus obtaining a collection of \( n = |V| \) one-vertex components, such that for every \( v \in V \) the one-vertex component \( \{v\} \) has degree \( (v) \) half edges attached to it.

2. We splice the two halves of every edge together, not of all the edges at once, but in some ordering \( \Theta \) of the edges that minimizes two measures that depend on the edge-boundary degrees of assembled components.

A component \( A \) is a subset of \( V \) and its edge-boundary degree is the number of edges in \( G \) with one endpoint in \( A \) and one endpoint in \( V - A \) (which is the same as the number of half edges attached to \( A \) after all edges with both endpoints in \( A \) have been spliced together). The maximum edge-boundary degree encountered during the reassembling process is what we call the \( \alpha \)-measure of the reassembling, and the sum of all edge-boundary degrees is its \( \beta \)-measure. The \( \alpha \)-optimization (resp. \( \beta \)-optimization) of the reassembling of \( G \) is to determine an order \( \Theta \) for splicing the edges that minimizes its \( \alpha \)-measure (resp. \( \beta \)-measure). There are different forms of reassembling, depending on restrictions and variations on the ordering \( \Theta \) of the edges. We consider only cases satisfying the condition that if an edge between disjoint components \( A \) and \( B \) is spliced, then all the edges between \( A \) and \( B \) are spliced at the same time. In this report, we examine the particular case of linear reassembling, which requires that the next edge to be spliced must be adjacent to an already spliced edge. We delay other forms of reassembling to follow-up reports. We prove that \( \alpha \)-optimization of linear reassembling...
and minimum-cutwidth linear arrangement (CutWidth) are polynomially reducible to each other, and that \( \beta \)-optimization of linear reassembling and minimum-cost linear arrangement (MinArr) are polynomially reducible to each other. The known NP-hardness of CutWidth and MinArr imply the NP-hardness of \( \alpha \)-optimization and \( \beta \)-optimization.

**Keywords** Minimum cutwidth · Optimal linear arrangement · Linear graph reassembling · Polynomial reducibility · NP-hardness

### 1 Introduction

We start with a gentle presentation of our graph problem and then explain the background that motivates our examination.

**Problem Statement** Let \( G = (V, E) \) be a simple (no self-loops and no multi-edges), connected, undirected graph, with \(|V| = n \geq 1\) vertices and \(|E| = m\) edges. One version of the reassembling of \( G \) is edge-directed and can be defined by a total order \( \Theta_1 \) of the \( m \) edges of \( G \). Informally and very simply, a total order \( \Theta \) of the edges gives rise to a reassembling of \( G \) as follows:

1. We cut every edge into two halves, thus obtaining a collection of \( n \) disconnected one-vertex components, such that for every \( v \in V \) the one-vertex component \( \{v\} \) has degree \( v \) half edges attached to it.
2. We reconnect the two halves of every edge in the order specified by \( \Theta \), obtaining larger and larger components, until the original \( G \) is fully reassembled.

To distinguish this reassembling of \( G \) according to an order \( \Theta \) from a later reassembling of \( G \) more suitable for parallel computation, we call the former *sequential reassembling* and the latter *binary reassembling*.

A bridge is a yet-to-be-reconnected edge between two components, say, \( A \) and \( B \), with disjoint sets of vertices; we call such components *clusters*. The set of bridges between \( A \) and \( B \) is denoted \( \partial(A, B) \). For technical reasons, when we reconnect one of the bridges in \( \partial(A, B) \), we also reconnect all the other bridges in \( \partial(A, B) \) and cross them out from further consideration in \( \Theta \). Thus, in the reassembling of \( G \) according to \( \Theta \), there are at most \( m \) steps, rather than exactly \( m \) steps.

In the case when \( B = V - A \), the set \( \partial(A, B) \) is the same as the cut-set of edges determined by the cut \( (A, V - A) \). Instead of \( \partial(A, V - A) \), we write \( \partial(A) \). The *edge-boundary degree* of a cluster \( A \) is the number of bridges with only one endpoint in \( A \), i.e., \(|\partial(A)|\).

Several natural optimization problems can be associated with graph reassembling. Two such optimizations are the following, which we identify by the letters \( \alpha \) and \( \beta \) throughout:

---

1 These terms (bridge, cluster, and others, later in this report) are overloaded in graph-theoretical problems. We make our own use of these terms, and state it explicitly when our meaning is somewhat at variance with that elsewhere in the literature.
(α) Minimize the maximum edge-boundary degree encountered during reassembling.

(β) Minimize the sum of all edge-boundary degrees encountered during reassembling.

Initially, before we start reassembling, we always set the α-measure \(M_\alpha\) to the maximum of all the vertex degrees, i.e., \(\max \{ \text{degree}(v) \mid v \in V \}\), and we set the β-measure \(M_\beta\) to the sum of the vertex degrees, i.e., \(\sum \{ \text{degree}(v) \mid v \in V \}\), regardless of which strategy, i.e., the order \(\Theta\) of edges, is selected for the reassembling. During reassembling, after we merge disjoint nonempty clusters \(A\) and \(B\), we update the α-measure \(M_\alpha\) to: \(\max \{M_\alpha, |\partial(A \cup B)|\}\), and the β-measure \(M_\beta\) to: \((M_\beta + |\partial(A \cup B)|)\). The reassembling process terminates when only one cluster remains, which is also the set \(V\) of all the vertices.

In what we call the binary reassembling of \(G\), we reconnect bridges in several non-overlapping pairs of clusters simultaneously. That is, at every step—which we may call a parallel step for emphasis—we choose \(k \geq 1\) and choose \(k\) cluster pairs \((A_1, B_1), \ldots, (A_k, B_k)\), where \(A_1, B_1, \ldots, A_k, B_k\) are \(2k\) pairwise disjoint clusters (i.e., with pairwise disjoint subsets of vertices), and simultaneously reconnect all the bridges in \(\sum_{1 \leq i \leq k} \partial(A_i, B_i)\). The subsets \(A_1, B_1, \ldots, A_k, B_k\) may or may not include all of the vertices, i.e., in general \(\sum_{1 \leq i \leq k} (A_i \cup B_i) \subseteq V\) rather than \(= V\). (We write “∪” to denote disjoint union.)

A binary reassembling is naturally viewed as vertex-directed and described by a binary tree \(T\)—root at the top, leaves at the bottom—with \(n\) leaf nodes, one for each of the initial one-vertex clusters. Each non-leaf node in \(T\) is a cluster \((A \cup B)\) obtained by reconnecting all the bridges in \(\partial(A, B)\) between the sibling clusters \(A\) and \(B\). The first parallel step, \(i = 0\), starts at the bottom in the reassembling process, by considering the \(n\) leaf nodes of \(T\) and calculating the max (for \(\alpha\) optimization) or the sum (for \(\beta\) optimization) of all vertex degrees. If \(h\) is the height of \(T\), the last parallel step is \(i = h\), which corresponds to the root node of \(T\) (the entire set \(V\) of vertices) and produces the final α-measure and β-measure. Clearly, \(|\log n| \leq h \leq n - 1\).

Every sequential reassembling of \(G\) can be viewed as a binary reassembling of \(G\) where, at every step, only one cluster pair \((A, B)\) is selected and one nonempty set of bridges \(\partial(A, B)\) is reconnected. Conversely, by serializing (or sequentializing) parallel steps, every binary reassembling which we call strict can be re-defined as a sequential reassembling. Details of the correspondence between sequential and binary reassemblings are in Appendix 1.

A binary reassembling is strict if the merging of a cluster pair \((A, B)\) is restricted to the case \(\partial(A, B) \neq \emptyset\). If an α-optimal (resp. β-optimal) binary reassembling is strict, then its serialization is an α-optimal (resp. β-optimal) sequential reassembling.

The Linear Case A possible and natural variation (or restriction) of graph reassembling is one which we call linear. If, at every step of the reassembling process, we require that the cluster pair \((A, B)\) to be merged is such that one of the two clusters, \(A\) or \(B\) (or both at the first step), is a singleton set, then the resulting reassembling is linear. The binary tree \(T\) describing a linear reassembling of a graph \(G\) with \(n\) vertices is therefore a degenerate tree of height \(h = n - 1\).
Clearly, there can be no non-trivial parallel step which merges two (or more) cluster pairs in linear reassembling, \textit{i.e.}, no step which simultaneously merges disjoint cluster pairs \((A_1, B_1)\) and \((A_2, B_2)\) to form the non-singleton clusters \((A_1 \uplus B_1)\) and \((A_2 \uplus B_2)\), before they are merged to form the cluster \(((A_1 \uplus B_1) \uplus (A_2 \uplus B_2))\).

Another useful way of understanding a linear reassembling of graph \(G\) is in its sequential formulation (when the reassembling is strict): The order \(\Theta\) for reconnecting the edges is such that the next edge to be reconnected is always adjacent to an already re-reconnected edge, which enforces the requirement that the next cluster pair \((A, B)\) to be merged is always such that \(A\) or \(B\) is a singleton set.

There are other natural variations of graph reassembling, such as balanced reassembling, whose binary tree \(B\) maximizes the merging of cluster pairs at every parallel step and whose height \(h\) is therefore \(\lceil \log n \rceil\). We study these in follow-up reports.

\textbf{Background and Motivation} Besides questions of optimization and the variations which it naturally suggests, graph reassembling (and the related operation of graph assembling, not considered in this paper) is part of the execution by programs in a domain-specific language (DSL) for the design of flow networks (Bestavros and Kfoury 2011; Kfoury 2011, 2013; Assaf 2014; Soule et al. 2011). In network reassembling, the network is taken apart and reassembled in an order determined by the designer; in network assembling, the order in which components are put together is pre-determined, which is the order in which components become available to the designer.

A flow network is a directed graph where vertices and edges are assigned various attributes that regulate flow through the network. \(^2\) Programs for flow-network design are meant to connect network components in such a way that \textit{typings at their interfaces}, \textit{i.e.}, formally specified properties at their common boundaries, are satisfied. Network typings guarantee there are no conflicting data types when different components are connected, and insure that desirable properties of safe and secure operation are not violated by these connections, \textit{i.e.}, they are \textit{invariant properties} of the whole network construction.

A typing \(\tau\) for a network component \(A\) (or \textit{cluster} \(A\) in this report’s terminology) formally expresses a constraining relationship between the variables denoting the outer ports of \(A\) (or the edge-boundary \(\partial(A)\) in this report). The smaller the set of outer ports of \(A\) is, the easier it is to formulate the typing \(\tau\) and to test whether it is compatible with the typing \(\tau'\) of another network component \(A'\). Although every outer port of \(A\) is directed, as input port or output port, the complexity of the formulation of \(\tau\) depends only on the number of outer ports (or \(|\partial(A)|\) in this report), not on their directions.

If \(k\) is a uniform upper bound on the number of outer ports of all network components, the time complexity of reassembling the network without violating any component typing \(\tau\) can be made linear in the size \(n\) of the completed network and exponential in the bound \(k\) – not counting the preprocessing time \(f(n)\) to determine an appropriate reassembling order. Hence, the smaller are \(k\) and \(f(n)\), the more efficient is the construction of the entire network. From this follows the importance of

\(^2\) Such networks are typically more complex than the capacitated directed graphs that algorithms for max-flow (and other related quantities) and its generalizations (\textit{e.g.}, multicommodity max-flow) operate on.
minimizing the preprocessing time $f(n)$ for finding a reassembling strategy that also minimizes the bound $k$. While aforementioned measure $\alpha$ corresponds to the uniform upper bound $k$, the measure $\beta$ is a representative for the average value of $k$ in the reassembling process.

**Main Results.** In this report, we restrict attention to the linear case of graph reassembling, which is interesting and natural in its own right. We first prove that $\alpha$-optimization and $\beta$-optimization of linear reassembling are both NP-hard problems. We obtain these results by showing that:

- $\alpha$-optimization of linear reassembling and *minimum-cutwidth linear arrangement* (CutWidth)
  are polynomially-reducible to each other,
- $\beta$-optimization of linear reassembling and *minimum-cost linear arrangement* (MinArr)
  are polynomially-reducible to each other.

Both CutWidth and MinArr have been extensively studied: They are both NP-hard in general, but, in a few non-trivial special cases, amenable to low-degree polynomial-time solutions Jordi (2011). By our polynomial-reducibility results, these polynomial-time solutions are transferable to the $\alpha$-optimization and $\beta$-optimization of linear reassembling. This leaves open the problem of identifying classes of graphs, whether of practical or theoretical significance, for which there are low-degree polynomial-time solutions for our two optimization problems.

**Organization of the Report.** In Sect. 2 we give precise formal definitions of several notions underlying our entire examination. The formulation of some of these (e.g., our definition of binary trees) is not standard, but which we purposely choose in order to facilitate the subsequent analysis.

In Sects. 3 and 4 we prove our NP-hardness results about $\alpha$-optimization and $\beta$-optimization. Some of the long technical proofs for these sections are delayed to Appendix 1.

In the final short Sect. 5, we point to open problems and to further current research on $\alpha$-optimization and $\beta$-optimization of graph reassembling.

### 2 Formal definitions and preliminary lemmas

Let $G = (V, E)$ be a simple (no self-loops and no multi-edges) undirected graph, with $|V| = n \geq 1$ and $|E| = m \geq 1$. Throughout, there is no loss of generality in assuming that $G$ is connected.

As pointed out in Sect. 1, there are two distinct definitions of the reassembling of $G$. The first is easier to state informally: This is the definition of *sequential reassembling*. The second definition, *binary reassembling*, is more convenient for the optimization problems we want to study. The analysis in this report and follow-up reports is based on the formal definition of binary reassembling and its variations; we delay a formal definition of sequential reassembling to Appendix 1, where we also sketch the proof of the equivalence of the two definitions when reassembling is *strict* (see Definition 5).
2.1 Binary graph-reassembling

Our notion of a binary reassembling of graph $G$ presupposes the notion of a binary tree over a finite set $V = \{v_1, \ldots, v_n\}$. Our definition of binary trees is not standard, but is more convenient for our purposes. 3

**Definition 1** (Binary trees) An (unordered) binary tree $B$ over $V = \{v_1, \ldots, v_n\}$ is a collection of non-empty subsets of $V$ satisfying three conditions:

1. For every $v \in V$, the singleton set $\{v\}$ is in $B$.
2. The full set $V$ is in $B$.
3. For every $X \in B$, there is a unique $Y \in B$ such that: $X \cap Y = \emptyset$ and $(X \cup Y) \in B$.

The leaf nodes of $B$ are the singleton sets $\{v\}$, and the root node of $B$ is the full set $V$. Depending on the context, we may refer to the members of $B$ as its nodes or as its clusters. Several expected properties of $B$, reproducing familiar ones of a standard definition, are stated in the next two propositions.

Our definition of binary tree slightly differs from the standard definition in the sense that we are not concerned with the order of the nodes with the same parent. Formally speaking given binary tree $B$ and $X, Y, Z \in B$ where $X$ and $Y$ are direct children of $Z$ (i.e. $X \cap Y = \emptyset$ and $X \cup Y = Z$), as apposed to the standard definition, in our definition of binary tree no ordering is assigned to the nodes $X$ and $Y$.

**Proposition 2** (Properties of binary trees) Let $B$ be a binary tree as in Definition 1, let $v \in V$, and let:

$$\{v\} = X_0 \subset X_1 \subset X_2 \subset \cdots \subset X_p = V \quad (\dagger)$$

be a maximal sequence of nested clusters from $B$. We then have:

1. The sequence in $(\dagger)$ is uniquely defined, i.e., every maximal nested sequence starting from $\{v\}$ is the same.
2. For every cluster $Y \in B$, if $\{v\} \subseteq Y$, then $Y \in \{X_0, \ldots, X_p\}$.
3. There are pairwise disjoint clusters $\{Y_0, \ldots, Y_{p-1}\} \subseteq B$ such that, for every $0 \leq i < p$:

$$X_i \cap Y_i = \emptyset \quad \text{and} \quad X_i \cup Y_i = X_{i+1}.$$ 

Based on this proposition, we use the following terminology:

- We call the sequence in $(\dagger)$, which is unique by part 1, the path from the leaf node $\{v\}$ to the root node $V$.

---

3 A standard definition of a binary tree $T$ makes $T$ a subset of the set of finite binary strings $\{0, 1\}^*$ such that:

1. $T$ is prefix-closed, i.e., if $t \in T$ and $u$ is a prefix of $t$, then $u \in T$.
2. Every node has two children, i.e., $0 \in T$ iff $1 \in T$ for every $t \in \{0, 1\}^*$.

The root node of $T$ is the empty string $\varepsilon$, and a leaf node of $T$ is a string $t \in T$ without children, i.e., both $t 0 \notin T$ and $t 1 \notin T$. 

© Springer
• Every cluster containing \( v \) is one of the nodes along this unique path, according to part 2.
• In part 3, we call \( Y_i \) the unique sibling of \( X_i \), whose unique common parent is \( X_{i+1} \), for every \( 0 \leq i < p \).

**Proof** Part 1 is a consequence of the third condition in Definition 1, which prevents any cluster/node in \( B \) from having two distinct parents.

For part 2, consider the nested sequence \( \{v\} \subseteq Y \subseteq V \), and extend it to a maximal nested sequence, which is uniquely defined by part 1. This implies \( Y \) is one of the nodes along the path from \( \{v\} \) to the root \( V \).

Part 3 is another consequence of the third condition in Definition 1: For every \( 0 \leq i < p \), \( Y_i \) is the unique cluster in \( B \) such that \( X_i \cap Y_i = \emptyset \) and \( X_i \cup Y_i \in B \).

Remaining details omitted. \( \Box \)

**Proposition 3** (Properties of binary trees) Let \( B \) be a binary tree as in Definition 1. We then have:

1. For all clusters \( X, Y \in B \), if \( X \cap Y \neq \emptyset \) then \( X \subseteq Y \) or \( Y \subseteq X \).
2. For every cluster \( X \in B \), the sub-collection of clusters \( B_X := \{ Y \in B \mid Y \subseteq X \} \) is a binary tree over \( X \), with root node \( X \).
3. \( B \) is a collection of \((2n - 1)\) clusters.

**Proof** For part 1, let \( Z = X \cap Y \neq \emptyset \) and consider the maximal sequence of nested clusters which extends \( Z \subseteq X \subseteq V \) or \( Z \subseteq Y \subseteq V \). By part 1 of Proposition 2, both \( X \) and \( Y \) must occur along this nested sequence.

For part 2, we directly check that \( B_X \) satisfies the three defining properties of a binary tree. Straightforward details omitted.

Part 3 is proved by induction on \( n \geq 1 \). For the induction hypothesis, we assume the statement is true for every binary tree with less than \( n \) leaf nodes, and we then prove that the statement is true for an arbitrary binary tree over \( V \) with \( |V| = n \). Consider a largest cluster \( X \in B \) such that \( X \neq V \). There is a unique \( Y \in B \) such that \( X \cap Y = \emptyset \) and \( X \cup Y \in B \). Because \( X \) is largest in size but smaller than \( V \), it must be that \( X \cup Y = V \), which implies that \( \{X, Y\} \) is a two-block partition of \( V \). Consider now the subtrees \( B_X \) and \( B_Y \), apply the induction hypothesis to each, and draw the desired conclusion. \( \Box \)

A common measure for a standard definition of binary trees is \textit{height}, which becomes for our notion of binary trees in Definition 1:

\[
\text{height}(B) := \max \{ p \mid \text{there is } v \in V \text{ such that } \{v\} = X_0 \subsetneq X_1 \subsetneq \cdots \subsetneq X_p = V \text{ is a maximal sequence of nested clusters} \}.
\]

For a particular node/cluster \( X \in B \), the subtree of \( B \) rooted at \( X \) is \( B_X \), by part 2 in Proposition 3. The height of \( X \) in \( B \) is therefore \( \text{height}_B(X) := \text{height}(B_X) \).

If \( X, Y \subseteq V \) are disjoint sets of vertices, \( \partial_G(X, Y) \) is the subset of edges of \( G \) with one endpoint in each of \( X \) and \( Y \). If \( Y = V - X \), we write \( \partial_G(X) \) instead of \( \partial_G(X, V - X) \).
Definition 4 (Binary reassembling) A binary reassembling of the graph \( G = (V, E) \) is simply defined by a pair \((G, B)\) where \( B \) is a binary tree over \( V \), as in Definition 1.

Given a binary reassembling \((G, B)\) of \( G \), two measures are of particular interest for our later analysis, namely, for every cluster \( X \in B \), the degree of \( X \) and the height of \( X \):

\[
\text{degree}_{G,B}(X) := |\partial G(X)| \quad \text{and} \quad \text{height}_{G,B}(X) := \text{height}_B(X).
\]

If the context makes clear the binary reassembling \((G, B)\) – respectively, the binary tree \( B \) – relative to which these measures are defined, we write degree \( (X) \) and height \( (X) \) – respectively, degree \( G(X) \) and height \( G(X) \) – instead of degree \( G,B(X) \) and height \( G,B(X) \).

Definition 5 (Strict binary reassembling) We say the binary reassembling \((G, B)\) in Definition 4 is strict if it satisfies the following condition: For all clusters \( X, Y \in B \), if \( X \) and \( Y \) are sibling nodes, then \( \partial G(X, Y) \neq \emptyset \). (In Definition 4, when we merge sibling clusters \( X, Y \in B \), we do not require that \( \partial G(X, Y) \neq \emptyset \).)

2.2 Optimization problems

The following definition repeats a definition in Sect. 1 more formally.

Definition 6 (Measures on the reassembling of a graph) Let \((G, B)\) be a binary reassembling of \( G \). We define the measures \( \alpha \) and \( \beta \) on \((G, B)\) as follows:

\[
\alpha(G, B) := \max \{ \text{degree}_{G,B}(X) \mid X \in B \}, \quad \beta(G, B) := \sum \{ \text{degree}_{G,B}(X) \mid X \in B \}.
\]

An optimization problem arises with the minimization of each of these measures. For example, the optimal \( \alpha \)-measure of graph \( G \) is:

\[
\alpha(G) = \min \{ \alpha(G, B) \mid (G, B) \text{ is a binary reassembling of } G \}.
\]

We say the binary reassembling \((G, B)\) is \( \alpha \)-optimal iff \( \alpha(G, B) = \alpha(G) \). We leave to the reader the obvious similar definition of what it means for the binary reassembling \((G, B)\) to be \( \beta \)-optimal.

2.3 Linear graph-reassembling

Let \( G = (V, E) \) be a simple undirected graph with \(|V| = n\). We say the binary reassembling \((G, B)\) is linear if \( B \) is a linear binary tree over \( V \), i.e., all the clusters of size \( \geq 2 \) forms a single nested chain of length \((n - 1)\):

---

4 Our binary reassembling of \( G \) can be viewed as the “hierarchical clustering” of \( G \), similar to a method of analysis in data mining, though used for a different purpose. Our binary reassembling mimicks what is called “agglomerative, or bottom-up, hierarchical clustering” in data mining.
$X_1 \subseteq X_2 \subseteq \cdots \subseteq X_{n-1} = V$ (A)

This implies the height of $B$ is $(n - 1)$. By part 3 in Proposition 2, there are $n$ leaf nodes singleton clusters $\{Y_0, \ldots, Y_{n-1}\} \subseteq B$ such that $X_1 = Y_0 \cup Y_1$ and for every $1 \leq i \leq n - 2$:

$$X_i \cap Y_{i+1} = \emptyset \quad \text{and} \quad X_i \cup Y_{i+1} = X_{i+1}.$$ (B)

We use the letter $L$ to denote a linear binary tree, and write $(G, L)$ to denote a linear reassembling of $G$.

In Definition 7, we mostly use the notation and conventions of Jordi (2011) and the references therein. We write $vw$ to denote the edge connecting vertex $v$ and vertex $w$.

**Definition 7** (Linear arrangements and cutwidths) A linear arrangement $\phi$ of the graph $G = (V, E)$, where $|V| = n$, is a bijection $\phi : V \to \{1, \ldots, n\}$. We refer to this linear arrangement by writing $(G, \phi)$.

Following Jordi (2011), given linear arrangement $(G, \phi)$ and $i \in \{1, \ldots, n\}$, we define a two-block partition of the vertices, $V = L(G, \phi, i) \cup R(G, \phi, i)$, where:

$$L(G, \phi, i) := \{ v \in V \mid \phi(v) \leq i \} \quad \text{and} \quad R(G, \phi, i) := \{ w \in V \mid \phi(w) > i \}.$$

The edge cut at position $i$, denoted $\zeta(G, \phi, i)$, is the number of edges connecting $L(G, \phi, i)$ and $R(G, \phi, i)$:

$$\zeta(G, \phi, i) := \left| \{ vw \in E \mid v \in L(G, \phi, i) \text{ and } w \in R(G, \phi, i) \} \right|.$$

In our notation in Definition 4, we have:

$$\zeta(G, \phi, i) = \left| \partial(L(G, \phi, i), R(G, \phi, i)) \right| = \text{degree}(L(G, \phi, i)).$$

The length of $vw$ in the linear arrangement $(G, \phi)$, denoted $\xi(G, \phi, vw)$, is “1 + the number of vertices between $v$ and $w$”:

$$\xi(G, \phi, vw) := \left| \phi(v) - \phi(w) \right|.$$ 

The $\alpha$-measure and $\beta$-measure of the linear arrangement $(G, \phi)$ are defined by:

$$\alpha(G, \phi) := \max \{ \zeta(G, \phi, i) \mid 1 \leq i \leq n \} = \max \{ \text{degree}(L(G, \phi, i)) \mid 1 \leq i \leq n \},$$

$$\beta(G, \phi) := \sum \{ \zeta(G, \phi, i) \mid 1 \leq i \leq n \} = \sum \{ \text{degree}(L(G, \phi, i)) \mid 1 \leq i \leq n \}.$$ 

In the literature, $\alpha(G, \phi)$ is called the cutwidth of the linear arrangement $(G, \phi)$. The cost of the linear arrangement $(G, \phi)$ is usually defined as the total length of all the edges relative to $(G, \phi)$, i.e., the cost is the measure $\gamma(G, \phi)$ given by:

$$\gamma(G, \phi) := \sum \{ \xi(G, \phi, vw) \mid vw \in E \}.$$ 

 Springer
However, by Lemma 8 below, $\beta(G, \varphi)$ is equal to $\gamma(G, \varphi)$.

**Lemma 8** For every linear arrangement $(G, \varphi)$, we have $\beta(G, \varphi) = \gamma(G, \varphi)$.

**Proof** We have to prove that

$$\sum \{ \zeta(G, \varphi, i) \mid 1 \leq i \leq n \} = \sum \{ \xi(G, \varphi, \overline{vw}) \mid \overline{vw} \in E \}.$$ 

This equality holds whether $G$ is connected or not. So, a formal proof (omitted) can be written by induction on the number $m \geq 0$ of edges in $G$. But informally, for every edge $\overline{vw} \in E$, if $\varphi(v) = i$ and $\varphi(w) = j$ with $i < j$, then its length $\xi(G, \varphi, \overline{vw}) = j - i$. In this case, the length of edge $\overline{vw}$ contributes one unit to each of $(j - i)$ consecutive edge cuts: $\zeta(G, \varphi, i), \ldots, \zeta(G, \varphi, j - 1)$. Hence, if we delete edge $\overline{vw}$ from the graph, we decrease the two quantities:

$$\sum \{ \zeta(G, \varphi, i) \mid 1 \leq i \leq n \} \quad \text{and} \quad \sum \{ \xi(G, \varphi, \overline{vw}) \mid \overline{vw} \in E \}.$$

by exactly the same amount $(j - i)$. The desired conclusion follows.

**Definition 9** (Optimal linear arrangements) Let $G = (V, E)$ be a simple undirected graph. We say the linear arrangement $(G, \varphi)$ is $\alpha$-optimal if:

$$\alpha(G, \varphi) = \min \{ \alpha(G, \varphi') \mid (G, \varphi') \text{ is a linear arrangement} \}.$$ 

The $\alpha$-optimal linear arrangement problem is the problem of defining a bijection $\varphi : V \to \{1, \ldots, n\}$ such that $(G, \varphi)$ is $\alpha$-optimal. We define similarly the $\beta$-optimal linear arrangement problem.

**Definition 10** (Linear arrangement induced by linear reassembling) Let $G = (V, E)$ be a simple undirected graph and $(G, \mathcal{L})$ be a linear reassembling of $G$. Using the notation in (A) and (B) in the opening paragraph of Sect. 2.3, the $n$ leaf nodes (or singleton clusters) of $\mathcal{L}$ are: $Y_0, Y_1, Y_2, \ldots, Y_{n-1}$. Observe that the order of the singletons $Y_2, \ldots, Y_{n-1}$ and, therefore, the order of the $(n - 2)$ vertices in $Y_2 \cup \cdots \cup Y_{n-1}$ is uniquely determined by the chain in (A), but this is not the case for the order in which we write $Y_0$ and $Y_1$, i.e., the first cluster $X_1$ in (A) is equal to both $Y_0 \cup Y_1$ and $Y_1 \cup Y_0$.

We want to extract a linear arrangement $(G, \varphi)$ from the linear reassembling $(G, \mathcal{L})$. This is achieved by defining $\varphi : V \to \{1, \ldots, n\}$ as follows:

- Let $Y_0 = \{v\}$ and $Y_1 = \{v'\}$.
  - If $\text{degree}(v) \leq \text{degree}(v')$ then set $\varphi(v) := 1$ and $\varphi(v') := 2$.
  - If $\text{degree}(v) \geq \text{degree}(v')$ then set $\varphi(v') := 1$ and $\varphi(v) := 2$.
- For every $2 \leq i \leq n - 1$, if $Y_i = \{v\}$ then set $\varphi(v) := i + 1$.

It is possible that $\text{degree}(v) = \text{degree}(v')$, in which case $\varphi$ may place $v$ first and $v'$ second, or $v'$ first and $v$ second. This ambiguity is harmless for our analysis, in that it does not affect the $\alpha$-measure and the $\beta$-measure of the linear arrangement $(G, \varphi)$. 

\(\text{Springer}\)
Whether \( \varphi \) places \( v \) first and \( v' \) second, or \( v' \) first and \( v \) second, we call \((G, \varphi)\) a linear arrangement of \( G \) induced by the linear reassembling \((G, \mathcal{L})\).

Note that, for every \( 1 \leq i \leq n - 1 \), we have the equality \( X_i = L(G, \varphi, i) \), where \( L(G, \varphi, i) \) is the set of vertices at position \( i \) and to the left of it, as in Definition 7.

**Definition 11** (Linear reassembling induced by linear arrangement) Let \( G = (V, E) \) be a simple undirected graph and let \((G, \varphi)\) be a linear arrangement of \( G \). We extract a linear reassembling \((G, \mathcal{L})\) from \((G, \varphi)\). Then the leaf nodes-singleton clusters of \( \mathcal{L} \) are:

\[
\{ \{v\} \mid v \in V \} = \{\varphi^{-1}(i)\} \mid 1 \leq i \leq n\}.
\]

The \((n - 1)\) non-leaf nodes-clusters of \( \mathcal{L} \) are:

\[
X_1 := \{\varphi^{-1}(1), \varphi^{-1}(2)\},
X_2 := \{\varphi^{-1}(1), \varphi^{-1}(2), \varphi^{-1}(3)\},
\ldots
X_{n-1} := \{\varphi^{-1}(1), \varphi^{-1}(2), \varphi^{-1}(3), \ldots, \varphi^{-1}(n)\} = V.
\]

We call \((G, \mathcal{L})\) the linear reassembling of \( G \) induced by the linear arrangement \((G, \varphi)\).

In what follows we present an example to illustrate notions mentioned in Sect. 1 and this section. For a complete list of examples readers can refer to Kfoury and Mirzaei (2015). We first introduce a convenient notation for specifying binary trees over a set \( V \) of vertices. Let \( 2^V \) denote the power set of \( V \). We define a binary operation \( XY \) for all \( X \subseteq 2^V \) and \( Y \subseteq 2^V \) by:

\[
XY := (X \cup Y) \cup ((\bigcup X) \cup (\bigcup Y))
\]

The examples below illustrate how we use this operation “\( \overline{\cdot} \)”.

We overload the overline notation “\( \overline{\cdot} \)” to denote a nonempty subset of \( V \), i.e., \( \overline{v_1 \cdots v_k} \) means \( \{v_1, \ldots, v_k\} \). In particular, the edge connecting \( v \) and \( w \) is denoted by the two-vertex set \( \overline{vw} = \{v, w\} \).

The context will make clear whether we use “\( \overline{\cdot} \)” to refer to a set of subsets of \( V \) (in the case of binary trees) or to just a subset of \( V \).

**Example 12** The hypercube graph \( Q_3 \) is shown on the left in Fig. 1, and three of its reassemblings are shown on the right in Fig. 1. The top reassembling is neither balanced nor linear; the middle one is balanced; and the bottom one is linear.

For each of the three reassemblings on the right in Fig. 1, from top to bottom, we list the unique binary tree \( B \) over the vertices \( \{1, \ldots, 8\} \) underlying it (in its binary reassembling formulation) and one of the orderings \( \Theta \) of the edges \( \{12, \ldots, 78\} \) inducing it (in its sequential reassembling formulation):

\[
\overline{123}
\]
Fig. 1 Two different drawings of the hypercube graph $Q_3$ (on the left) and three of its reassemblings (on the right)

$$B_1 = \begin{cases} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 12 & 34 & 56 & 78 & 13 & 35 & 57 & \cdots \end{cases} \quad \Theta_1^{Q_3} = 12 \ 34 \ 56 \ 78 \ 13 \ 35 \ 57 \ \cdots$$

$$B_2 = \begin{cases} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 12 & 34 & 56 & 78 & 13 & 57 & 35 & \cdots \end{cases} \quad (balanced) \quad \Theta_2^{Q_3} = 12 \ 34 \ 56 \ 78 \ 13 \ 57 \ 35 \ \cdots$$

$$B_3 = \begin{cases} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 12 & 3 & 4 & 5 & 6 & 7 & 8 & \cdots \end{cases} \quad (linear) \quad \Theta_3^{Q_3} = 12 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ \cdots$$

where, for simplicity, we write just “v” instead of the cumbersome $\{\{v\}\} = \overline{v}$. Thus, for example, two of the subsets of $B_1$ above appear as “1 2” and “12 3 4”, and if we expand them in full, we obtain:

1 2 = \{1, 2, \{1, 2\}\} \quad and \quad 1 2 3 4 = \{1, 2, \{1, 2\}, \{1, 2, 3\}, \{2, 3\}\}. \quad (12)

The ellipsis “...” in the definition of $\Theta_i^{Q_3}$ above are the remaining edges of $Q_3$, which can be listed in any order without changing the reassembling. \footnote{We qualify $\Theta_i^{Q_3}$ with the superscript “$Q_3$” because it depends on the graph $Q_3$. The same ordering of the edges may not be valid for a sequential reassembling of another 8-vertex graph with a set of edges different from that of $Q_3$. This is not the case for the binary tree $B$ underlying the binary reassembling $(G, B)$ of a graph $G = (V, E)$; that is, regardless of the placement of edges in $G$, the tree $B$ over $V$ is valid for the binary reassembling $(G, B)$ and again for the binary reassembling $(G', B)$ of every graph $G' = (V, E')$ over the same set $V$ of vertices.}

A simple calculation of the $\alpha$-measure and $\beta$-measure of these three reassemblings of $Q_3$ produces:

$$\alpha(Q_3, B_1) = \alpha(Q_3, B_2) = 4 \quad and \quad \alpha(Q_3, B_3) = 5,$$

$$\beta(Q_3, B_1) = \beta(Q_3, B_2) = 48 \quad and \quad \beta(Q_3, B_3) = 49.$$

By exhaustive inspection (details omitted), $(Q_3, B_1)$ is both $\alpha$-optimal and $\beta$-optimal for the class of all binary reassemblings of $Q_3$. Because the $\alpha$-measure and $\beta$-measure of $(Q_3, B_1)$ and those of $(Q_3, B_2)$ are equal, $(Q_3, B_2)$ is also both $\alpha$-optimal and $\beta$-optimal for the class of all binary reassemblings and, therefore, for the smaller class of all \textit{balanced} reassemblings of which it is a member. By exhaustive
inspection again, \((Q_3, B_3)\) is both \(\alpha\)-optimal and \(\beta\)-optimal for the sub-class of all linear reassemblings, but not for the full class of all binary reassemblings of \(Q_3\).

### 3 \(\alpha\)-Optimization of linear reassembling is NP-hard

We prove that the \(\alpha\)-optimality of linear arrangements (in the literature: the minimum-cutwidth linear arrangement problem) and the \(\alpha\)-optimality of linear reassemblings are reducible to each other in polynomial time.

**Definition 13** (Chordal graph, triangulation, clique number, treewidth) Let \(G = (V, E)\) be a simple undirected graph. The following are standard notions of graph theory (Deistel 2012).

- \(G\) is a **chordal graph** if every cycle of length of 4 or more has a chord, i.e., an edge connecting two vertices that are not consecutive in the cycle.
- A **triangulation** of \(G\) is a chordal graph \(G' = (V', E')\) where \(V = V'\) and \(E \subseteq E'\). In such a case, we say that \(G\) can be **triangulated into** \(G'\), not uniquely in general.
- The **clique number** of \(G\), denoted \(\omega(G)\), is the size of a largest clique in \(G\).
- There are different equivalent definitions of the **treewidth**. We here use a definition, or a consequence of the original definition, which is more convenient for our purposes (Bronner and Ries 2006; Deistel 2012). The **treewidth** of \(G\) is:

\[
\min \{ \omega(G') \mid G' \text{ is a triangulation of } G \} - 1.
\]

In words, among all triangulations \(G'\) of \(G\), we choose a \(G'\) whose clique number is smallest: The treewidth of \(G\) is one less than the clique number of such a \(G'\).

**Lemma 14** For every positive integers \(\Delta\) and \(k\), there is an algorithm \(A\) using \(\Delta\) and \(k\) as fixed parameters, such that, given an arbitrary simple undirected graph \(G = (V, E)\) as input to \(A\), if:

1. the maximum vertex degree in \(G\) is \(\leq \Delta\), and
2. the treewidth of \(G\) is \(\leq k\),

then \(A\) computes a minimum-cutwidth linear arrangement of \(G\) in time \(O(n^{\Delta k^2})\) where \(n = |V|\).

**Proof** This is Theorem 4.2 in Thilikos et al. (2001), where the algorithm not only computes the value of a minimum cutwidth, but can be adjusted to output the corresponding minimum-cutwidth linear arrangement \(\varphi : V \to \{1, \ldots, n\}\). \(\square\)

In Lemma 15, a **cut vertex** in \(G\) is a vertex whose removal increases the number of connected components.

**Lemma 15** There is an algorithm \(A\) such that, given an arbitrary simple undirected graph \(G = (V, E)\) as input to \(A\), if:

1. every vertex in \(G\) has degree \(\leq 3\), and
2. every vertex in \(G\) of degree \(= 3\) is a cut vertex,
then \( A \) computes a minimum-cutwidth linear arrangement of \( G \) in time \( O(n^{12}) \) where \( n = |V| \).

**Proof** We show that the treewidth \( k \) of \( G \) is \( \leq 2 \). Because the maximum vertex degree \( \Delta \) of \( G \) is \( \leq 3 \), Lemma 14 implies the existence of an algorithm \( A \) which runs in time \( O(n^{2k}) = O(n^{12}) \).

To show that \( k \leq 2 \), consider a vertex \( v \) of degree \( = 3 \), which is therefore a cut vertex. The removal of \( v \) can have one of two possible outcomes:

(a) disconnect \( G \) into 3 components, or
(b) disconnect \( G \) into 2 components.

If every vertex of degree \( = 3 \) satisfies condition (a), then \( G \) is tree whose treewidth is 1, since its clique number \( \omega(G) = 2 \) in this case.

If \( C_1 \) and \( C_2 \) are cycles in \( G \), each with 3 vertices or more, then \( C_1 \) and \( C_2 \) are non-overlapping, i.e., \( C_1 \) and \( C_2 \) have no vertex in common and no edge in common.

If they have an edge \( \bar{v}_1 \bar{v}_2 \) in common, then there is an edge \( \bar{v}_2 \bar{v}_2 \in C_1 \cap C_2 \) such that \( degree(v_2) = 3 \) (or, resp., \( degree(w_2) = 3 \)) and \( v_2 \) (or, resp., \( w_2 \)) is not a cut vertex, contradicting the hypothesis. If \( C_1 \) and \( C_2 \) have no edge in common, but do have a vertex \( v \) in common, then \( degree(v_2) > 3 \), again contradicting the hypothesis.

In case one or more vertices satisfy condition (b), \( G \) can be therefore viewed as a finite collection of non-overlapping rings \( \{R_1, \ldots, R_p\} \), each ring being a cycle with at least 3 vertices, satisfying condition (c):

(c) if two distinct rings \( \{R_i, R_j\} \), with \( i \neq j \), are connected by a path \( P_{i,j} \), then the removal of all the vertices and edges of \( P_{i,j} \) (in particular the two endpoints of \( P_{i,j} \), one in \( R_i \) and one in \( R_j \), which are necessarily vertices of degree \( = 3 \)) disconnects \( G \) into 2 components.

Another way of expressing (c) is that, if all the rings \( \{R_1, \ldots, R_p\} \) are each contracted to a single vertex, then the result is a tree (where some of the internal vertices may now have degree larger than 3). Since the clique number of a ring is 3, the treewidth of a ring is 2, and the desired conclusion follows.

**Lemma 16** Let \( G = (V, E) \) be a simple undirected graph, where every vertex has degree \( \leq 3 \), and let \( (G, L) \) be a linear reassembling of \( G \). Consider the longest chain of nested clusters of size \( \geq 2 \), as in (A) in the opening paragraph of Sect. 2.3:

\[
X_1 \subsetneq X_2 \subsetneq \cdots \subsetneq X_{n-1} = V.
\]

**Conclusion:** If there is one vertex of degree \( = 3 \) in \( G \) which is not a cut vertex, then max \{\( degree(X_1), \ldots, degree(X_{n-1}) \)\} \( \geq 3 \).

**Proof** We first show there are at least two vertices of degree \( = 3 \) which are not cut vertices. Let \( v \) be a vertex of degree \( = 3 \) which is not a cut vertex, and let \( \{\bar{u}x, \bar{u}y, \bar{u}z\} \) be the three edges incident to \( v \). Because \( v \) is not a cut vertex, any two edges in \( \{\bar{u}x, \bar{u}y, \bar{u}z\} \) are consecutive edges in a cycle containing \( v \). Let \( C(v, x, y) \) be a cycle containing edges \( \{\bar{u}x, \bar{u}y\} \), and define similarly cycles \( C(v, x, z) \) and \( C(v, y, z) \). If any of these three cycles contains a chord, then the two endpoints of the chord are...
vertices of degree = 3 which are not cut vertices. If none of these three cycles contain a chord, then we can combine any two of them, because they share an edge, to form another cycle with a chord, which again implies the existence of two vertices of degree = 3 which are not cut vertices.

To conclude the proof, consider the clusters of $\mathcal{L}$ of size $\geq 2$: These are $\{X_1, \ldots, X_{n-1}\}$, and the corresponding singleton clusters are $\{Y_0, \ldots, Y_{n-1}\}$, as in (A) and (B) in Sect. 2.3. By the preceding argument, there are at least two vertices of degree = 3 which are not cut vertices. Let one of these two be $v$, with $Y_i = \{v\}$ for some $i \geq 1$.

We have $X_{i-1} \cap Y_i = \emptyset$ and $X_{i-1} \cup Y_i = X_i$. There are two cases: (1) For some vertex $w \in X_{i-1}$, there is an edge $w w' \in E$, and (2) for every vertex $w \in X_{i-1}$, there is no such edge. We consider case (1) and leave the other (easier) case (2) to the reader.

We cannot have $\deg(X_{i-1}) = 0$, otherwise $G$ is disconnected, nor can we have $\deg(X_{i-1}) = 1$, otherwise $v$ is a cut vertex. Hence, $\deg(X_{i-1}) \geq 2$. If $\deg(X_{i-1}) \geq 3$, this is already the conclusion of the lemma and there is nothing else to prove. Suppose $\deg(X_{i-1}) = 2$, the case left to consider.

Similarly, we cannot have $\deg(X_i) = 0$, otherwise $G$ is disconnected, nor $\deg(X_i) = 1$, otherwise $v$ is a cut vertex. Hence, $\deg(X_i) \geq 2$. But if $\deg(X_{i-1}) = 2$ and $\deg(X_i) \geq 2$, with $\deg(v) = 3$, then it must be that $\deg(X_i) = 3$.

Lemma 17 Let $G = (V, E)$ be a simple undirected graph, where every vertex has degree $\leq 3$ and where one vertex of degree = 3 is not a cut vertex. Let $(G, \mathcal{L})$ be a linear reassembling and $(G, \varphi)$ a linear arrangement.

Conclusion: If $(G, \mathcal{L})$ is induced by $(G, \varphi)$, or if $(G, \varphi)$ is induced by $(G, \mathcal{L})$, then:

- $\alpha(G, \mathcal{L}) = \alpha(G, \varphi)$.
- $(G, \mathcal{L})$ is $\alpha$-optimal iff $(G, \varphi)$ is $\alpha$-optimal.

Proof Straightforward consequence of Lemma 16, the definitions of $\alpha(G, \mathcal{L})$ and $\alpha(G, \varphi)$, and what it means for $(G, \mathcal{L})$ to be induced by $(G, \varphi)$ and for $(G, \varphi)$ to be induced by $(G, \mathcal{L})$. When there is at least one vertex of degree = 3 which is not a cut vertex, and therefore at least two of them by the proof of Lemma 16, we can ignore the degrees of singleton clusters in the computation of $\alpha(G, \mathcal{L})$. All details omitted. □

Theorem 18 For the class of simple undirected graphs $G$ where every vertex has degree $\leq 3$, the $\alpha$-optimality of linear arrangements $(G, \varphi)$ is polynomial-time reducible to the $\alpha$-optimality of linear reassemblings $(G, \mathcal{L})$.

More explicitly, a polynomial-time algorithm $\mathcal{A}$, which returns an $\alpha$-optimal linear reassembling $(G, \mathcal{L})$ of a graph $G$ where every vertex has degree $\leq 3$, can be used to return an $\alpha$-optimal linear arrangement $(G, \varphi)$.

Proof Consider an arbitrary $G$ where every vertex has degree $\leq 3$. If every vertex in $G$ of degree = 3 is a cut vertex, we use the algorithm in Lemma 15 to compute an $\alpha$-optimal linear arrangement $(G, \varphi)$ in time $O(n^{12})$. If there is a vertex in $G$ of degree = 3 which is not a cut vertex, we first compute an $\alpha$-optimal linear reassembling $(G, \mathcal{L})$ and then return the linear arrangement $(G, \varphi)$ induced by $(G, \mathcal{L})$. The desired conclusion follows from Lemma 17. □
Corollary 19 For the class of all simple undirected graphs $G$, the computation of $\alpha$-optimal linear reassemblings $(G, L)$ is an NP-hard problem.

Proof If there is a polynomial-time algorithm $A$ to compute, for an arbitrary simple undirected graph, an $\alpha$-optimal linear reassembling, then the same algorithm $A$ can be used to compute in polynomial-time an $\alpha$-optimal linear reassembling $(G, L)$ for a graph $G$ where every vertex has degree $\leq 3$. By Theorem 18, $A$ can be further adapted to compute an $\alpha$-optimal linear arrangement $(G, \varphi)$ for such a graph $G$ in polynomial time. But the latter problem (in the literature: the minimum-cutwidth linear arrangement problem) is known to be NP-hard (Makedon Fillia et al. 1985; Burkhard and Ivan Hal 1988).

Remark 20 To the best of our knowledge, the complexity status of the minimum-cutwidth linear arrangement problem for $k$-regular graphs for a fixed $k \geq 3$ is an open problem. If it were known to be NP-hard, we would be able to simplify our proof of Theorem 18 and its corollary considerably. In particular, we would be able to eliminate Lemmas 14 and 15 and the supporting Definition 13, as well as simplify Lemmas 16 and 17 by restricting them to $k$-regular graphs.

Theorem 18 and Corollary 19 together say the $\alpha$-optimality of linear arrangements $(G, \varphi)$ is polynomial-time reducible to the $\alpha$-optimality of linear reassemblings $(G, L)$. For completeness, we show the converse in the next theorem.

Theorem 21 For the class of simple undirected graphs $G$ in general, the $\alpha$-optimality of linear reassemblings $(G, L)$ is polynomial-time reducible to the $\alpha$-optimality of linear arrangements $(G, \varphi)$.

Proof In Appendix 1.

4 $\beta$-Optimization of linear reassembling is NP-hard

We prove that the $\beta$-optimality of linear arrangements (in the literature: the minimum-cost linear arrangement problem or also the optimal linear arrangement problem) and the $\beta$-optimality of linear reassemblings are reducible to each other in polynomial time. Towards this end, we prove an intermediate result, which is also of independent interest (Theorem 26 which presupposes Definition 22).

Definition 22 (Anchored linear reassemblings) Let $G = (V, E)$ be a simple undirected graph and let $w \in V$. Let $(G, L)$ be a linear reassembling of $G$, whose longest chain of nested clusters of size $\geq 2$, as in (A) in the opening paragraph of Sect. 2.3, is:

\[ X_1 \subsetneq X_2 \subsetneq \cdots \subsetneq X_{n-1} = V \]

and whose corresponding singleton clusters are $\{Y_0, \ldots, Y_{n-1}\}$, as determined by (B) in the opening paragraph of Sect. 2.3. We say $(G, L)$ is a linear reassembling anchored at $w \in V$ iff there is a vertex $w' \in V$ such that:

\[ Y_0 = \{w\}, \quad Y_1 = \{w'\}, \quad \text{and} \quad \text{degree}_G(w) \leq \text{degree}_G(w'). \]
Note that we require that the immediate sibling $Y_1 = \{w'\}$ of the leaf node $Y_0 = \{w\}$ satisfy the condition $\deg_G(w) \leq \deg_G(w')$. This implies that, given an arbitrary vertex $w \in V$, we cannot anchor a linear reassembling at $w$ unless we find another vertex $w' \in V$ such that $\deg_G(w) \leq \deg_G(w')$ and then make $\{w\}$ and $\{w'\}$ sibling leaf-nodes. This is a technical restriction to simplify the statement of Lemma 24.

**Definition 23** *(Anchored linear arrangements)* Let $G = (V, E)$ be a simple undirected graph and let $w \in V$. Let $(G, \varphi)$ be a linear arrangement of $G$. We say $(G, \varphi)$ is a linear arrangement anchored at $w \in V$ iff there is a vertex $w' \in V$ such that:

$$\varphi(w) = 1, \quad \varphi(w') = 2, \quad \text{and} \quad \deg_G(w) \leq \deg_G(w').$$

Again, as in Definition 22, the condition $\deg_G(w) \leq \deg_G(w')$ is imposed in order to simplify the statement of Lemma 24. It is worth noting that, if we relax this condition and allow $\deg_G(w) > \deg_G(w')$, then the new arrangement $\varphi'$ which permutes the positions of $w$ and $w'$, i.e.:

$$\varphi'(w') = 1, \quad \varphi'(w) = 2, \quad \text{and} \quad \varphi'(v) = \varphi(v) \quad \text{for all} \quad v \in V - \{w, w'\},$$

is such that $\beta(G, \varphi') < \beta(G, \varphi)$. In words, if we allowed $\deg_G(w) > \deg_G(w')$, the linear arrangement $(G, \varphi)$ would not be $\beta$-optimal. 6

Let $(G, \mathcal{L})$ be a linear reassembling anchored at vertex $w \in V$. We say $(G, \mathcal{L})$ is $\beta$-optimal relative to anchor $w$ iff:

$$\beta(G, \mathcal{L}) = \min \left\{ \beta(G, \mathcal{L}') \mid (G, \mathcal{L}') \text{ is a linear reassembling anchored at } w \right\}.$$  

Clearly, $(G, \mathcal{L})$ is a $\beta$-optimal linear reassembling, with no anchor restriction, iff:

$$\beta(G, \mathcal{L}) = \min \left\{ \beta(G, \mathcal{L}') \mid \text{there is a vertex } w \in V \text{ and} \right\}$$

$$(G, \mathcal{L}') \text{ is a linear reassembling $\beta$-optimal relative to anchor } w \right\}.$$ 

Similar obvious conditions apply to what it means for $(G, \varphi)$ to be a $\beta$-optimal linear arrangement relative to anchor $w$.

**Lemma 24** Let $G = (V, E)$ be a simple undirected graph and $w \in V$. Let $(G, \mathcal{L})$ be a linear reassembling of $G$ anchored at $w$, and $(G, \varphi)$ be a linear arrangement of $G$ anchored at $w$, such that:

- $(G, \varphi)$ is induced by $(G, \mathcal{L})$ or $(G, \mathcal{L})$ is induced by $(G, \varphi)$.

**Conclusion:** $(G, \mathcal{L})$ is $\beta$-optimal relative to anchor $w$ iff $(G, \varphi)$ is $\beta$-optimal relative to anchor $w$.

---

6 A similar statement applies to the $\alpha$-measure: If we allowed $\deg_G(w) > \deg_G(w')$, then the new arrangement $\varphi'$ would be such that $\alpha(G, \varphi') \leq \alpha(G, \varphi)$, but not necessarily $\alpha(G, \varphi') < \alpha(G, \varphi)$. 
Proof Let \( d = \text{degree}(w) \geq 1 \) and \( \Delta = \sum \{ \text{degree}(v) \mid v \in V \text{ and } v \neq w \} \). Consider the case when arrangement \((G, \varphi)\) is induced by reassembling \((G, \mathcal{L})\). (We omit the case when reassembling \((G, \mathcal{L})\) is induced by arrangement \((G, \varphi)\), which is treated similarly.) From Definition 6,

\[
\beta(G, \mathcal{L}) = d + \Delta + \sum \{ \text{degree}(X_i) \mid 1 \leq i \leq n - 1 \}
\]

where \( X_1, \ldots, X_{n-1} \) are all the clusters of size \( \geq 2 \) in \( \mathcal{L} \). From Definitions 7 and 10,

\[
\beta(G, \varphi) = d + \sum \{ \text{degree}(X_i) \mid 1 \leq i \leq n - 1 \}
\]

Hence, both \( \beta(G, \mathcal{L}) \) and \( \beta(G, \varphi) \) are minimized when the same quantity \( \sum \{ \text{degree}(X_i) \mid 1 \leq i \leq n - 1 \} \) is minimized. The desired conclusion follows. \( \blacksquare \)

Remark 25 There is an obvious definition of anchored \( \alpha \)-optimality, similar to that of anchored \( \beta \)-optimality above. However, results for the latter do not necessarily have counterparts for the former. In particular, the conclusion of Lemma 24 does not hold for \( \alpha \)-optimality. Specifically, there are simple counter-examples showing the existence of a simple undirected graph \( G(V, E) \) with a distinguished vertex \( w \in V \) such that:

- there is a linear reassembling \((G, \mathcal{L})\) which is \( \alpha \)-optimal relative to anchor \( w \),
- but the linear arrangement \((G, \varphi)\) induced by \((G, \mathcal{L})\) is not \( \alpha \)-optimal relative to anchor \( w \).

There is an examination, yet to be undertaken, of the relation between linear reassemblings \((G, \mathcal{L})\) and linear arrangements \((G, \varphi)\) that are \( \alpha \)-optimal relative to the same anchor, similar to our study of anchored \( \beta \)-optimality below. This examination we do not pursue in this report.

Theorem 26 For the class of all simple undirected graphs \( G = (V, E) \), each with a distinguished vertex \( w \in V \), the two following problems are polynomial-time reducible to each other:

- the \( \beta \)-optimality of linear arrangements \((G, \varphi)\) anchored at \( w \),
- the \( \beta \)-optimality of linear reassemblings \((G, \mathcal{L})\) anchored at \( w \).

More explicitly, a polynomial-time algorithm \( A \), which returns a linear reassembling \((G, \mathcal{L})\) [resp. a linear arrangement \((G, \varphi)\)] which is \( \beta \)-optimal relative to anchor \( w \) can be used to return in polynomial time a linear arrangement \((G, \varphi)\) [resp. a linear reassembling \((G, \mathcal{L})\)] which is \( \beta \)-optimal relative to anchor \( w \).

Proof This is an immediate consequence of Lemma 24. \( \blacksquare \)

Definition 27 (Auxiliary graphs) Let \( G = (V, E) \) be a simple undirected graph, with \( |V| = n \) and \( |E| = m \). For every \( w \in V \) we define what we call an auxiliary graph \( G_w = (V_w, E_w) \) as follows:
Thus, the subgraph of $G_w$ induced by the set $V$ is simply the original graph $G$, and the subgraph of $G_w$ induced by the set $U \cup \{w\}$ is the complete graph $K_{p+1}$ over $p+1$ vertices.

Informally, $G_w$ is constructed from $G$ and the complete graph $K_{p+1}$ by identifying vertex $w \in V$ with one of the vertices of $K_{p+1}$. In particular, $w$ is a cut vertex of the auxiliary graph $G_w$. We call $w$, which is the common vertex of $G$ and $K_{p+1}$, the distinguished vertex of $G_w$.

Lemma 28 If $G_w = (V_w, E_w)$ is the auxiliary graph for vertex $w \in V$, as constructed in Definition 27, then $|V_w| \leq n^2$ and $|E_w| \leq (n^4 - 2n^3 + 3n^2 - 2n)/2$.

Proof The number $m$ of edges in $G$ is bounded by $(n^2 - n)/2$. Hence, $p = \sum \{\deg(v) \mid v \in V\} \leq (n^2 - n)$, implying that the total number of vertices $p+n$ in $G_w$ is $\leq (n^2 - n) + n = n^2$. The number of edges in $K_p$ is $(p^2 - p)/2$, and in $K_{p+1}$ it is $(p^2 + p)/2$, which is $\leq ((n^2 - n)^2 + (n^2 - n))/2 = (n^4 - 2n^3 + 3n^2 - 2n)/2$. Hence, the total number of edges in $G_w$ is $m + (p^2 + p)/2 \leq (n^4 - 2n^3 + 3n^2 - 2n)/2$. $\square$

Let $L$ be a linear binary tree over $V$ where, as in (A) in the opening paragraph of Sect. 2.3, the longest chain of nested clusters of size $\geq 2$ is:

$$X_1 \subset Y_1 \subset \cdots \subset X_{n-1} = V,$$

and let the corresponding singleton clusters be $\{Y_0, \ldots, Y_{n-1}\}$ as determined by (B).

The linear tree $L$ is uniquely determined by a sequence of vertices written in the form:

$$[v_1 \cdots v_n]$$

where $Y_0 = \{v_1\}, Y_1 = \{v_2\}, \ldots, Y_{n-1} = \{v_n\}$. We say $[v_1 \cdots v_n]$ is the vertex sequence induced by $L$, and $L$ the linear reassembling (or the linear binary tree) induced by the vertex sequence $[v_1 \cdots v_n]$.

Similarly, if $\varphi : V \to \{1, \ldots, n\}$ is a linear arrangement of $V$, then $\varphi$ is uniquely determined by a sequence of vertices in the same form:

$$[v_1 \cdots v_n]$$

where $\varphi^{-1}(1) = v_1, \varphi^{-1}(2) = v_2, \ldots, \varphi^{-1}(n) = v_n$. We say $[v_1 \cdots v_n]$ is the vertex sequence induced by $\varphi$, and $\varphi$ the linear arrangement induced by the vertex sequence $[v_1 \cdots v_n]$.

For the auxiliary graph $G_w$, whether we deal with a linear reassembling $(G_w, L)$ or a linear arrangement $(G_w, \varphi)$, it is convenient to consider sequences of the following form, which interleaves vertices and cutwidths:
\[ S := \left[ x_1 (r_1, s_1) \ x_2 (r_2, s_2) \ \cdots \ x_{n+p-1} (r_{n+p-1}, s_{n+p-1}) \ x_{n+p} \right] \]

where \( \{x_1, \ldots, x_{n+p}\} = V_w = \{v_1, \ldots, v_n\} \cup \{u_1, \ldots, u_p\} \), and for every \( 1 \leq i \leq n + p - 1 \):

\[ r_i := \text{degree}_G([x_1, \ldots, x_i]) \quad \text{and} \quad s_i := \text{degree}_{K_{p+1}}([x_1, \ldots, x_i]). \]

We say the sequence \( S \) in (♦) is the sequence of vertices and cutwidths induced by \( (G_w, \mathcal{L}) \) or by \( (G_w, \varphi) \), whichever of the two is the case. The measure \( \beta \) on \( S \) is:

\[ \beta(S) := \sum_{1 \leq i \leq n+p-1} (r_i + s_i). \]

**Lemma 29** Consider the sequence of vertices and cutwidths induced by \( (G_w, \mathcal{L}) \) or by \( (G_w, \varphi) \), as just defined. **Conclusion:**

- For every \( 1 \leq i \leq n + p - 1 \), it holds that \( r_i + s_i = \text{degree}_{G_w}([x_1, \ldots, x_i]). \)
- If the sequence in (♦) is induced by the linear arrangement \( (G_w, \varphi) \), then

\[ \beta(G_w, \varphi) = \beta = \sum_{1 \leq i \leq n+p-1} (r_i + s_i). \]

- If the sequence in (♦) is induced by the linear reassembling \( (G_w, \mathcal{L}) \), then

\[ \beta(G_w, \mathcal{L}) = \Delta + \beta(S) = \Delta + \sum_{1 \leq i \leq n+p-1} (r_i + s_i), \]

where \( \Delta = \sum \{ \text{degree}_{G_w}(v) \mid v \in V_w \text{ and } v \neq x_1 \}. \)

**Proof** Straightforward consequence of the definitions. All details omitted. \( \square \)

We say that the sequence \( S \) is **scattered** if the vertices of \( K_{p+1} \) do not occur consecutively, i.e., the vertices of \( K_{p+1} \) are interspersed with vertices of \( V - \{w\} \).

**Lemma 30** Let \( G_w = (V_w, E_w) \) be the auxiliary graph for vertex \( w \in V \), as constructed in Definition 27. Let \( S \) be the sequence of vertices and cutwidths, as in (♦), induced by a \( \beta \)-optimal linear reassembling \( (G_w, \mathcal{L}) \) or by a \( \beta \)-optimal linear arrangement \( (G_w, \varphi) \). **Conclusion:** \( S \) is not scattered.

In words, in a \( \beta \)-optimal linear reassembling \( (G_w, \mathcal{L}) \) [or in a \( \beta \)-optimal linear arrangement \( (G_w, \varphi) \), resp.] all the vertices of \( K_{p+1} \) are reassembled consecutively [or arranged consecutively, resp.] without intervening vertices from \( V - \{w\} \).

**Proof** In Appendix 1. \( \square \)
Consider again the sequence $S$ of vertices and cutwidths in (♦). Suppose $S$ is not scattered. This means that the $p + 1$ vertices of $K_{p+1}$ occur consecutively in $S$. We say $S$ is balanced iff one of two conditions holds:

(1) $\{x_1, \ldots, x_{n-1}\} = V - \{w\}$, $\{x_n\} = \{w\}$, $\{x_{n+1}, \ldots, x_{n+p}\} = U$.
(2) $\{x_1, \ldots, x_p\} = U$, $\{x_{p+1}\} = \{w\}$, $\{x_{p+2}, \ldots, x_{n+p}\} = V - \{w\}$.

In words, $S$ is balanced if all the vertices of $V - \{w\}$ are on the same side (on the left in (1), or on the right in (2)) of the distinguished vertex $w$ and all the vertices of $U$ are on the other side (on the right in (1), or on the left in (2)) of $w$. Put differently still, $S$ is balanced if all the vertices of $V - \{w\}$ are together, all the vertices of $U$ are together, and $w$ is between the two sets of vertices. The following is a refinement of the preceding lemma and its proof is based on a similar argument.

**Lemma 31** Let $G_w = (V_w, E_w)$ be the auxiliary graph for vertex $w \in V$, as constructed in Definition 27. Let $S$ be the sequence of vertices and cutwidths, as in (♦), induced by a $\beta$-optimal linear reassembling $(G_w, \mathcal{L})$ or by a $\beta$-optimal linear arrangement $(G_w, \varphi)$. Conclusion: $S$ is balanced.

**Proof** In Appendix 1. □

By the preceding lemma, if the sequence $S$ in (♦) is induced by a $\beta$-optimal linear reassembling $(G_w, \mathcal{L})$, or by a $\beta$-optimal linear arrangement $(G_w, \varphi)$, then $S$ is balanced, either on the left or on the right. For the rest of the analysis below, we assume that $S$ is balanced on the right, i.e., all the vertices in $U$ occur first, then $w$, and then all the vertices of $V - \{w\}$.

**Definition 32** Restrictions of linear reassemblings and linear arrangements Let $\mathcal{L}$ be a linear binary tree over the set $V$. If $V' \subseteq V$, the restriction of $\mathcal{L}$ to $V'$, denoted $(\mathcal{L} \mid V')$, consists of the following clusters:

$$(\mathcal{L} \mid V') := \{X \cap V' \mid X \in \mathcal{L}\}.$$

It is a straightforward exercise to show that $(\mathcal{L} \mid V')$ is a linear binary tree over $V'$.

Let $\varphi : V \to \{1, \ldots, n\}$ be a linear arrangement of $V$. The restriction of $\varphi$ to $V'$, denoted $(\varphi \mid V')$, is defined as follows. For every $1 \leq i \leq n' = |V'|$, let:

$$(\varphi \mid V')(v) := i \text{ where } v = \varphi^{-1}(j) \text{ and } j \in \{1, \ldots, n\} \text{ is the largest integer such that } |\{\varphi^{-1}(1), \ldots, \varphi^{-1}(j-1)\} \cap V'| = i - 1.$$

Again here, it is straightforward to show that $(\varphi \mid V')$ is a linear arrangement of $V'$ such that:

$$(\varphi \mid V')^{-1}(1), \ldots, (\varphi \mid V')^{-1}(n') \text{ is a subsequence of } \varphi^{-1}(1), \ldots, \varphi^{-1}(n).$$

Moreover, if $(G, \mathcal{L})$ is a linear reassembling [resp. $(G, \varphi)$ is a linear arrangement] of the simple undirected graph $G = (V, E)$ and $G' = (V', E')$ is the subgraph of $G$
induced by \( V' \subseteq V \), then \((G', (L \mid V'))\) is a linear reassembling [resp. \((G', (\varphi \mid V'))\) is a linear arrangement] of \( G' \).

**Lemma 33** Let \( G_w = (V_w, E_w) \) be the auxiliary graph for vertex \( w \in V \), as constructed in Definition 27.

1. If \((G_w, L)\) is a \( \beta \)-optimal linear reassembling of \( G_w \) with no anchor restriction, then \((G, (L \mid V))\) is a \( \beta \)-optimal linear reassembling relative to anchor \( w \).
2. If \((G_w, \varphi)\) is a \( \beta \)-optimal linear arrangement of \( G_w \) with no anchor restriction, then \((G, (\varphi \mid V))\) is a \( \beta \)-optimal linear arrangement relative to anchor \( w \).

**Proof** We prove part 1 only, the proof of part 2 is similar. By Lemma 31, the sequence \( S \) induced by a \( \beta \)-optimal linear reassembling \((G_w, L)\) is balanced. By our assumption preceding Definition 22, we take \( S \) to be balanced on the right, i.e., all the vertices in \( U \) occur first, then \( w \), and then all the vertices of \( V - \{w\} \). There are no edges connecting vertices in \( U \) on the left to vertices in \( V - \{w\} \) on the right, with \( w \) a cut vertex in the middle. The \( \beta \)-optimality of \((G_w, L)\) implies the \( \beta \)-optimality of the linear reassembling \((G, (L \mid V))\) of the subgraph \( G = (V, E) \) of \( G_w = (V_w, E_w) \). We omit all formal details. \( \square \)

**Theorem 34** For the class of all simple undirected graphs \( G \), the two following problems are polynomial-time reducible to each other:

- the \( \beta \)-optimality of linear arrangements \((G, \varphi)\),
- the \( \beta \)-optimality of linear reassemblings \((G, L)\).

More explicitly, a polynomial-time algorithm \( A \), which returns a \( \beta \)-optimal linear reassembling \((G, L)\) [resp. a \( \beta \)-optimal linear arrangement \((G, \varphi)\)] of an arbitrary graph \( G \), can be used to return a \( \beta \)-optimal linear arrangement \((G, \varphi)\) [resp. a \( \beta \)-optimal linear reassembling \((G, L)\)] in polynomial time.

**Proof** We compute a \( \beta \)-optimal linear reassembling \((G_{v_i}, L_i)\) [resp. a \( \beta \)-optimal linear arrangement \((G_{v_i}, \varphi_i)\)] of the auxiliary graph \( G_{v_i} \) one for each vertex \( v_i \in V = \{v_1, \ldots, v_n\} \). We next consider the linear reassembling \((G, (L_i \mid V))\) [resp. the linear arrangement \((G, (\varphi_i \mid V))\)] which, by Lemma 33, is a \( \beta \)-optimal linear reassembling relative to anchor \( v_i \) [resp. a \( \beta \)-optimal linear arrangement relative to anchor \( v_i \)], for every \( 1 \leq i \leq n \). Let \((G, \varphi_i)\) be the linear arrangement induced by the linear reassembling \((G, (L_i \mid V))\) [resp. let \((G, L_i)\) be the linear reassembling induced by the linear arrangement \((G, (\varphi_i \mid V))\)]. By Lemma 24, \((G, \varphi_i)\) is a \( \beta \)-optimal linear arrangement relative to anchor \( v_i \) [resp. \((G, L_i)\) is a \( \beta \)-optimal linear reassembling relative to anchor \( v_i \)], for every \( 1 \leq i \leq n \). Among these \( n \) linear arrangements [resp. \( n \) linear reassemblings], we choose one such that \( \beta(G, \varphi_i) \) is minimized [resp. \( \beta(G, L_i) \) is minimized]. \( \square \)

**Corollary 35** For the class of all simple undirected graphs \( G \), the computation of \( \beta \)-optimal linear reassemblings \((G, L)\) is an NP-hard problem.

**Proof** This follows from the NP-hardness of the minimum-cost linear arrangement problem (also called the optimal linear arrangement problem in the literature) (Garey Michael et al. 1976). This problem is the same as what we call, in this report, the problem of computing a \( \beta \)-optimal linear arrangement. \( \square \)
Remark 36  To the best of our knowledge, the complexity status of the minimum-cost linear arrangement problem (or optimal linear arrangement problem) for $k$-regular graphs for a fixed $k \geq 3$ is an open problem. If it were known to be NP-hard, we would be able to simplify our proof of Theorem 34 and its corollary considerably.

5 Related and future work

We mentioned several open problems from the literature, still unresolved to the best of our knowledge, in Remarks 20, 25, and 36. If these open problems were solved, partially or optimally, they would permit various simplifications in our proofs. In particular, even though one of our reductions can be carried out in polynomial time by invoking an earlier result on cutwidths (Lemmas 14 and 15), its $O(n^{12})$ complexity is prohibitive (see the proof of Theorem 18); this is the reduction that reduces the $\alpha$-optimality of linear arrangements to the $\alpha$-optimality of linear reassemblies.

Beyond open problems whose resolutions would simplify and/or strengthen some of this report’s results and their proofs, our wider research agenda is to tackle forms of graph reassembling other than linear—in particular, balanced reassembling and binary reassembling in general, both strict and non-strict, all alluded to in Sects. 1 and 2. For each form of reassembling, both $\alpha$-optimization and $\beta$-optimization will have to be addressed; as suggested by the examination in this report, these two optimizations seem to call for different proof methods, despite their closely related definitions.

We also need to study classes of graphs for which $\alpha$-optimization and/or $\beta$-optimization of their reassembling, in any of the forms mentioned above, can be carried out in low-degree polynomial times. Finally, there is the question of whether, by allowing approximate solutions, we can turn the NP-hardness of any of the preceding optimizations into polynomially-solvable optimizations. The literature on approximation algorithms dealing with graph layout problems is likely to be an important resource to draw from (among many other papers, the older (Arora et al. 1996; Tom and Satish 1999; Rao and Richa 1998) the more recent (Charikar et al. 2006), and the survey (Jordi 2011).

Acknowledgements  Partially supported by NSF awards CCF-0820138 and CNS-1135722.

Appendix 1: sequential graph reassembling

Let $\mathcal{P}$ be the set of all the partitions of the set $V = \{v_1, \ldots, v_n\}$ of vertices in the graph $G = (V, E)$. There are two special partitions in $\mathcal{P}$:

$$P_0 := \{ \{v\} \mid v \in V \} \quad \text{and} \quad P_\infty := \{V\}.$$

Given two partitions $X, Y \in \mathcal{P}$, we write $X \sqsubseteq Y$ if $X$ is finer than $Y$ or, equivalently, $Y$ is coarser than $X$, i.e., for every block $A \in X$ there is a block $B \in Y$ such that $A \subseteq B$. We write $X \sqsubset Y$ iff $X \subseteq Y$ and $X \neq Y$. The relation “$\sqsubseteq$” is a (non-strict) partial order on $\mathcal{P}$, with a least element (the finest partition $P_0$) and a largest element (the coarsest partition $P_\infty$). We need the following simple fact.
Lemma 37 In the poset \((\mathcal{P}, \sqsubseteq)\) of all partitions of \(n\) elements, a maximal chain (linearly ordered with \(\sqsubseteq\)) is a sequence of \(n\) partitions, always starting with \(P_0\) and ending with \(P_\infty\).

Proof If \(X_1 \sqsubseteq X_2 \sqsubseteq \cdots \sqsubseteq X_k\) is a maximal chain of partitions, necessarily with \(X_1 = P_0\) and \(X_k = P_\infty\) because the chain is maximal, then \(X_1\) has \(n\) blocks, \(X_2\) has \(n - 1\) blocks, in general \(X_p\) has \(n - p + 1\) blocks, and \(X_k\) has one block. The length \(k\) of the chain is therefore exactly \(n\). \(\square\)

Definition 38 (Sequential graph reassembling, i.e., according to an ordering of the edges) Let \(\Theta\) be an ordering of the edges in \(E\). We use \(\Theta\) to select \(n\) partitions in \(\mathcal{P}\) forming a maximal chain (linearly ordered with \(\sqsubseteq\)), which starts with the finest partition \(P_0\) and ends with the coarsest partition \(P_\infty = \{V\}\), say:
\[
X_1 \sqsubseteq X_2 \sqsubseteq \cdots \sqsubseteq X_n \quad \text{where} \quad X_1 = P_0 \quad \text{and} \quad X_n = P_\infty,
\]
as we explain next. To define \(X_{p+1}\) from \(X_p\), we associate each \(X_p\) with a subsequence \(\Theta_p\) of the initial sequence \(\Theta_1 = \Theta\), for every \(p \geq 1\). The subsequence \(\Theta_p\) keeps track of all the edges that have not yet been reconnected. We obtain the next pair \((X_{p+1}, \Theta_{p+1})\) from the preceding pair \((X_p, \Theta_p)\) as follows:

1. Take the first edge \(e\) in the sequence \(\Theta_p\), i.e., let \(\Theta_p = e \Theta_p'\) for some \(\Theta_p'\), with \(e = vw\) for some \(v, w \in V\), and let \(A\) and \(B\) be the unique blocks in \(X_p\) containing \(v\) and \(w\), respectively.
2. Merge the two blocks \(A\) and \(B\) to obtain \(X_{p+1}\), i.e., let:
\[
X_{p+1} := \left( X_p - \{ A, B \} \right) \cup \{ A \cup B \}.
\]
3. Delete every edge \(e'\) whose two endpoints are in the new block \(A \cup B\) to obtain \(\Theta_{p+1}\), i.e., let:
\[
\Theta_{p+1} := \Theta_p \setminus \{ e' \in E \mid e' = v'w' \quad \text{and} \quad \{ v', w' \} \subseteq A \cup B \}.
\]

In words, we go from \((X_p, \Theta_p)\) to \((X_{p+1}, \Theta_{p+1})\) by merging the two blocks \(A\) and \(B\) in \(X_p\) that are connected by the first edge \(e\) in \(\Theta_p\), and then removing from further consideration all edges whose endpoints are in \(A \cup B\).

We refer to the sequential reassembling of \(G\) according to the ordering \(\Theta\) by writing \((G, \Theta)\), the result of which is the chain of partitions \(X = X_1 \sqsubseteq \cdots \sqsubseteq X_n\), more succinctly written also as \(X = X_1 \cdots X_n\).

Remark 39 In Definition 38, when we merge the two blocks \(A\) and \(B\) because the edge \(e\) has its two endpoints in \(A\) and \(B\), not only do we reconnect the two halves of \(e\), but we additionally reconnect every other edge \(e'\) whose two endpoints are also in \(A\) and \(B\). Thus, in general, we may reconnect several edges simultaneously – all the edges between \(A\) and \(B\) in the original graph – rather than one at a time by strictly following the order specified by \(\Theta\). The same happens with binary graph-reassembling (Definition 4).
Definition 38 describes the process of going from an ordering \( \Theta \) of edges to a maximal chain \( \mathcal{X} \) of partitions. If \( \mathcal{X} = X_1 \cdots X_n \) is a maximal chain of partitions, we say that \( \mathcal{X} \) is strict if, for every consecutive pair \((X_p, X_{p+1})\) with \( A, B \in X_p \) and \( A \cup B \in X_{p+1} \), where \( 1 \leq p < n \), it is the case that \( \partial G(A, B) \neq \emptyset \). The result of a sequential reassembling \((G, \Theta)\) is always a strict maximal chain \( \mathcal{X} \) of partitions. We can also carry out the process in reverse, as asserted by the next lemma.

Lemma 40 (From a maximal chain \( \mathcal{X} \) of partitions to an ordering \( \Theta \) that induces it)

Let \( G = (V, E) \) be a graph, and \( \mathcal{P} \) the set of all partitions of \( V \), as in Definition 38. For every maximal chain of partitions \( \mathcal{X} = X_1 \sqsubset \cdots \sqsubset X_n \), with \( X_1, \ldots, X_n \in \mathcal{P} \), if \( \mathcal{X} \) is strict, then there is an ordering (not necessarily unique) \( \Theta \) of \( E \) such that \((G, \Theta) = \mathcal{X} \).

Proof This is a consequence of Definition 38. Details omitted. \( \square \)

We want to relate the two notions: sequential reassembling \((G, \Theta)\) in Definition 38 and binary reassembling \((G, B)\) in Definition 4. The discussion to follow uses the following facts.

Lemma 41 Let \( B \) be a binary tree over \( V = \{v_1, \ldots, v_n\} \), given in the formulation of Definition 1.

1. If \( S = \{X_1, \ldots, X_k\} \subseteq B \) is a maximal collection of \( k \geq 2 \) pairwise disjoint sets in \( B \), then \( S \) is a partition of \( V \). We call such a maximal collection \( S \) a cross-section of the binary tree \( B \).
2. If \( \mathcal{I} \) is the set of all cross-sections of \( B \), then \( (\mathcal{I}, \sqsubseteq) \) is a proper sub-poset of the poset \( (\mathcal{P}, \sqsubseteq) \) in Lemma 37, with the same bottom element \( P_0 \) and top element \( P_\infty \).
3. In the poset \( (\mathcal{I}, \sqsubseteq) \), a maximal chain has exactly \( n \) entries, always starting with \( P_0 \) and ending with \( P_\infty \).

Proof All three parts can be proved by induction on \( n \geq 1 \), using the same reasoning as in the proofs of Propositions 2 and 3. All details omitted. \( \square \)

In the preceding lemma, it is worth noting that the size of \( \mathcal{P} \) is fixed as a function of \( n \), the so-called Bell number \( B(n) \), which counts the partitions of an \( n \)-element set and grows exponentially in \( n \). \(^7\) By contrast, the size of \( \mathcal{I} \) is much smaller, depends on both \( n \) and the shape of the binary tree \( B \), and can be as small as \( n \) (the case when \( B \) is a linear, i.e., a degenerate binary tree).

Consider a sequential reassembling \((G, \Theta)\) of the graph \( G = (V, E) \), the result of which is a maximal chain of \( n \) partitions \( \mathcal{X} = X_1 \sqsubset \cdots \sqsubset X_n \), as in Definition 38, where \( X_1 = P_0 \) and \( X_n = P_\infty \). Since every successive partition \( X_{p+1} \) in \( \mathcal{X} \) is obtained from the previous \( X_p \) by merging two blocks in \( X_p \), there is a natural way of organizing \( \mathcal{X} \) in the form of a binary tree \( B \), with \( n \) (not all) of the cross-sections of \( B \) being exactly \( \{X_1, \ldots, X_n\} \). Let \( \text{binary}(G, \Theta) \) denote the binary reassembling thus obtained.

\(^7\) There is no known simple expression for the exponential growth of \( B(n) \) as a function of \( n \), though there are various ways of estimating tight lower bounds and tight upper bounds on its asymptotic growth (Odlyzko 1995).
Consider next a binary reassembling \((G, \mathcal{B})\) of the graph \(G = (V, E)\). The set \(\mathcal{S}\) of all cross-sections in \(\mathcal{B}\) is uniquely defined. We want to extract from \(\mathcal{S}\) a maximal chain \(\mathcal{X}\) of cross-sections/partitions, ordered by \(\sqsubseteq\), which, by Lemma 40, will in turn induce an ordering of \(\Theta\) of the edges. The problem here is that there are generally many such maximal chains \(\mathcal{X}\). We need therefore a method to canonically extract a unique maximal chain \(\mathcal{X}\) from \(\mathcal{S}\) and a unique edge-ordering \(\Theta\) from \(\mathcal{X}\). We propose such a method in the next paragraph.

We assume that the binary reassembling \((G, \mathcal{B})\) is strict and that there is a fixed ordering of the vertices, say, \(v_1 < v_2 < \cdots < v_n\). The vertex ordering “\(<\)” is extended to edges, and to sets of edges, as follows:

- If \(e = \overline{vw}\) is the edge joining vertices \(v\) and \(w\), we assume \(v < w\).
- If \(e = \overline{vw}\) and \(e' = \overline{v'w'}\), then \(e < e'\) iff either \(v < v'\) or \(v = v'\) and \(w < w'\).
- If \(A \subseteq E\), then \(\text{canon}(A)\) is the canonical ordering of \(A\) w.r.t. “\(<\)”, i.e., \(\text{canon}(A) = e_1 e_2 \cdots e_k\) where \(A = \{e_1, e_2, \ldots, e_k\}\) and \(e_1 < e_2 < \cdots < e_k\).
- If \(A\) and \(B\) are non-empty disjoint set of edges, with \(\text{canon}(A) = e_1 e_2 \cdots\) and \(\text{canon}(B) = f_1 f_2 \cdots\), then \(\text{canon}(A) < \text{canon}(B)\) iff \(e_1 < f_1\).

If \(W \in \mathcal{B}\), then \(\mathcal{B}_W\) is the subtree of \(\mathcal{B}\) rooted at \(W\) (see Proposition 3). We write \((G, \mathcal{B}_W)\) for a partial binary reassembling of the graph \(G = (V, E)\), the result being the subgraph of \(G\) induced by \(W\) together with all the edges in \(\partial_G(W)\) as dangling edges, i.e., edges with only one endpoint in \(W\). We define a canonical ordering of all the edges already in place in the partial reassembling \((G, \mathcal{B}_W)\), denoted \(\text{canon}(G, \mathcal{B}_W)\), as follows:

\[
\text{canon}(G, \mathcal{B}_W) := \begin{cases} 
\varepsilon \text{ (the empty string)} & \text{if } W \text{ is a singleton set,} \\
\text{canon}(G, \mathcal{B}_T) \text{ canon}(G, \mathcal{B}_U) \text{ canon}(\partial(T, U)) & \text{if } W = T \sqcup U \text{ and,} \\
\text{canon}(G, \mathcal{B}_T) \text{ canon}(G, \mathcal{B}_U) \text{ canon}(\partial(T, U)) & \text{if } W = T \sqcup U \text{ and,} \\
\text{canon}(G, \mathcal{B}_U) \text{ canon}(G, \mathcal{B}_T) \text{ canon}(\partial(T, U)) & \text{if } W = T \sqcup U \text{ and,} \\
\text{canon}(G, \mathcal{B}_T) \text{ canon}(G, \mathcal{B}_U) \text{ canon}(\partial(T, U)) & \text{if } W = T \sqcup U \text{ and,} \\
\end{cases}
\]

Because the binary reassembling \((G, \mathcal{B})\) is strict, \(\partial(T, U) \neq \emptyset\) in the second and third cases above, which implies \(\text{canon}(\partial(T, U)) \neq \varepsilon\). If \(W = V\), then \((G, \mathcal{B}) = (G, \mathcal{B}_W)\) and \(\text{canon}(G, \mathcal{B}) = \text{canon}(G, \mathcal{B}_W)\).

**Proposition 42** (Relating sequential reassembling and binary reassembling) *Let \(G = (V, E)\) be a simple undirected graph. We have the following facts:

1. For every sequential reassembling \((G, \Theta)\), there is a binary tree \(\mathcal{B}\) over \(V\) such that:

   \[
   \text{binary}(G, \Theta) = (G, \mathcal{B}).
   \]

2. For every strict binary reassembling \((G, \mathcal{B})\), there is an ordering \(\Theta\) of \(E\) such that:

   \[
   \text{canon}(G, \mathcal{B}) = (G, \Theta).
   \]
3. For every strict binary reassembling \((G, B)\), it holds that:

\[
\text{binary}(\text{canon}(G, B)) = (G, B).
\]

**Proof** Parts 1 and 2 follow from the definitions and discussion that precede the proposition. All details omitted. Part 3 can be proved by structural induction on the subtrees \(B_W\) of \(B\), where the induction hypothesis is \(\text{binary}(\text{canon}(G, B_W)) = (G, B_W)\). All details omitted again. \(\square\)

It is possible to refine the notion of “canonical ordering” on the set of edges \(E\), so that the equality \(\text{canon}(\text{binary}(G, \Theta)) = (G, \Theta)\) holds which, together with the equality in part 3 of the preceding proposition, will mean that the functions \(\text{binary}()\) and \(\text{canon}()\) are inverses of each other. We omit this refinement as it will go further afield from our main concerns.

**Appendix 2: Remaining proofs for Sects. 3 and 4**

We supply the details of several long straightforward and/or highly technical proofs which we omitted in Sects. 3 and 4 in order to facilitate the grasp of the different concepts and their mutual dependence.

**Proof of Theorem 21** Let \(G = (V, E)\) be an arbitrary simple undirected graph, with \(|V| = n\). It suffices to show that if \((G, \varphi)\) is an \(\alpha\)-optimal linear arrangement, then the linear reassembling \((G, L)\) induced by \((G, \varphi)\) is also \(\alpha\)-optimal, using Definition 11.

In the notation of Definition 11, the clusters of \(L\) of size \(\geq 2\) are \(\{X_1, \ldots, X_{n-1}\}\). For the singleton clusters of \(L\), we pose \(Y_{i-1} := \{\varphi^{-1}(i)\}\), where \(1 \leq i \leq n\). From Definition 7:

\[
\alpha(G, \varphi) = \max \{\text{degree}(Y_0), \max \{\text{degree}(X_j) \mid 1 \leq j \leq n - 1\}\},
\]

\[
\alpha(G, L) = \max \{\max \{\text{degree}(Y_i) \mid 0 \leq i \leq n - 1\}, \max \{\text{degree}(X_j) \mid 1 \leq j \leq n - 1\}\}.\]

By way of getting a contradiction, assume that \((G, \varphi)\) is \(\alpha\)-optimal but that the induced \((G, L)\) is not \(\alpha\)-optimal. Hence, there is another linear reassembling \((G, L')\) which is \(\alpha\)-optimal such that \(\alpha(G, L') < \alpha(G, L)\). Using the same notation for both \((G, L)\) and \((G, L')\), where every name related to the latter is decorated with a prime, the inequality \(\alpha(G, L') < \alpha(G, L)\) implies the inequality:

\[
\max \{\max \{\text{degree}(Y_i') \mid 0 \leq i \leq n - 1\}, \max \{\text{degree}(X_j') \mid 1 \leq j \leq n - 1\}\} < \max \{\max \{\text{degree}(Y_i) \mid 0 \leq i \leq n - 1\}, \max \{\text{degree}(X_j) \mid 1 \leq j \leq n - 1\}\}.
\]

But \(\max \{\text{degree}(Y_i') \mid 0 \leq i \leq n - 1\} = \max \{\text{degree}(Y_i) \mid 0 \leq i \leq n - 1\}\), which implies two inequalities:

\[
\max \{\text{degree}(Y_i) \mid 1 \leq i \leq n - 1\} < \max \{\text{degree}(X_j) \mid 1 \leq j \leq n - 1\},
\]

\(1\)
max \{ \text{degree}(X_j) \mid 1 \leq j \leq n - 1 \} < \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n - 1 \}. \quad (2)

Hence, by inequality (1), we have:

\[ \alpha(G, \varphi) = \alpha(G, \mathcal{L}) = \max \{ \text{degree}(X_j) \mid 1 \leq j \leq n - 1 \}. \]

Consider now the linear arrangement \((G, \varphi')\) induced by the linear reassembling \((G, \mathcal{L}')\), using Definition 10. We have:

\[ \alpha(G, \varphi') = \max \{ \text{degree}(Y'_0), \max \{ \text{degree}(X'_j) \mid 1 \leq j \leq n - 1 \} \}. \]

If \(\text{degree}(Y'_0) \geq \max \{ \text{degree}(X'_j) \mid 1 \leq j \leq n - 1 \}\), then inequality (1) implies \(\alpha(G, \varphi') < \alpha(G, \varphi)\), else inequality (2) implies again \(\alpha(G, \varphi') < \alpha(G, \varphi)\). In both cases, the \(\alpha\)-optimality of \((G, \varphi)\) is contradicted. \(\square\)

For the proofs of Lemmas 30 and 31, we take a closer look at how the vertices of \(K_{p+1}\) are positioned in the sequence \(\mathcal{S}\) in \((\Diamond)\) in Sect. 4. From the fact that \(p\) is the sum of all the vertex degrees in \(G\), it follows that \(p\) is even and \(p + 1\) odd. From the sequence \(\mathcal{S}\), we can extract the subsequence \((\mathcal{S} \mid K_{p+1})\) consisting of all the vertices of \(K_{p+1}\) and corresponding cutwidths:

\[(\mathcal{S} \mid K_{p+1}) = [x_{i_1} \ s_{i_1} \ x_{i_2} \ s_{i_2} \ \cdots \ \cdots \ x_{i_p} \ s_{i_p} \ x_{i_{p+1}}]\]

where \(\{i_1, \ldots, i_{p+1}\} \subseteq \{1, \ldots, n + p\}\) and \(\{x_{i_1}, \ldots, x_{i_{p+1}}\} = \{u_1, \ldots, u_p\} \cup \{w\}\). In the preceding sequence, every vertex has the same degree \(p\) in the subgraph \(K_{p+1}\). In the full graph \(G_w\), every vertex from \(K_{p+1}\) has again the same degree \(p\), except for the distinguished vertex \(w\) which has degree \(p + d\) where \(d = \text{degree}_G(w)\). In particular, we have:

\[s_{i_1} = p, \quad s_{i_2} = 2 \cdot (p - 1), \quad s_{i_3} = 3 \cdot (p - 2), \ldots, \quad s_{i_{p-1}} = (p - 1) \cdot 2, \quad s_{i_p} = p.\]

The mid-point of \((\mathcal{S} \mid K_{p+1})\) is \(x_{i_{(p/2)+1}}\). The two adjacent cutwidths of the mid-point \(x_{i_{(p/2)+1}}\) are:

\[s_{i_{(p/2)}} = \frac{p}{2} \cdot \left(\frac{p}{2} + 1\right) \quad \text{and} \quad s_{i_{(p/2)+1}} = \left(\frac{p}{2} + 1\right) \cdot \frac{p}{2},\]

so that also, as one can readily check:

\[s_{i_{(p/2)}} = s_{i_{(p/2)+1}} = \frac{p^2 + 2p}{4} = \max \{s_{i_1}, s_{i_2}, \ldots, s_{i_p}\},\]

and the sequence of cutwidths \((s_{i_1}, \ldots, s_{i_p})\) is equal to its own reverse \((s_{i_p}, \ldots, s_{i_1})\). Moreover, for every \(j\) such that \(1 \leq j < i_1\) or \(i_{p+1} < j \leq n + p\), we have \(s_j = 0\).
Also, it is intuitively useful for the argument in the proof of Lemma 30 to keep in mind that:

\[(s_{i_1} - s_j) = p, \quad \text{for every} \quad 1 \leq j < i_1,\]
\[(s_{i_2} - s_j) = p - 2, \quad \text{for every} \quad i_1 \leq j < i_2,\]
\[(s_{i_3} - s_j) = p - 4, \quad \text{for every} \quad i_2 \leq j < i_3,\]
\[
\ldots \ldots \ldots
\]
\[(s_{i_{(p/2)}} - s_j) = 2, \quad \text{for every} \quad i_{(p/2)-1} \leq j < i_{p/2},\]
\[(s_{i_{(p/2)+1}} - s_j) = 0, \quad \text{for every} \quad i_{p/2} \leq j < i_{(p/2)+1}.\]

**Proof of Lemma 30** In the sequence \(S\) in \(\Diamond\) in Sect. 4, suppose:

- \(x_i\) is the leftmost vertex in \(U \cup \{w\}\),
- \(x_j\) is the leftmost vertex in \(V - \{w\}\) to the right of \(x_i\),
- \(x_k\) is the rightmost vertex in \(U \cup \{w\}\),
- \(x_\ell\) is the rightmost vertex in \(V - \{w\}\) to the left of \(x_\ell\),

where \(1 \leq i \leq j \leq k \leq \ell \leq p + n\). Graphically, \(S\) can be represented by:

\[
\begin{array}{cccccccc}
 x_1 & \cdots & x_{i-1} & x_i & \cdots & x_{j-1} & \circled{x_j} & x_{j+1} \\
 \text{all in } V - \{w\} & & & & & & & \text{all in } U \cup \{w\} \\
 \cdots & x_k & x_{k+1} & \cdots & x_\ell & \cdots & x_{\ell+1} & \cdots & x_{p+n} \\
 & \text{all in } U \cup \{w\} & & & & & & \text{all in } V - \{w\}
\end{array}
\]

The circled vertices, \(x_j\) and \(x_k\), are in \(V - \{w\}\). If \(S\) is scattered, then \(1 \leq i < j\) and/or \(k < \ell \leq n + p\), with the possibility that \(j = k\) in which case there is only one vertex in \(V - \{w\}\) inserted between all the vertices of \(U \cup \{w\}\). We define:

\[scatter(S) := \min \{j - i, \ell - k\} \geq 1,\]

when \(S\) is scattered. If \(S\) is not scattered, we set \(scatter(S) := 0\), so that \(S\) is scattered iff \(scatter(S) \geq 1\). Moreover, with \(p\) even and \(p + 1\) odd, it is always the case that \(scatter(S) \leq p/2\), so that if \(S\) is scattered, then:

\[1 \leq scatter(S) \leq \frac{p}{2}.
\]

To complete the proof, it suffices to show that if \(S\) is scattered, then we can define another sequence \(S'\) from \(S\) such that:

\[\beta(S') < \beta(S) \quad \text{and} \quad scatter(S') < scatter(S).
\]

We obtain \(S'\) from \(S\) as follows:

1. if \(j - i \leq \ell - k\), remove \(x_j\) from the \(j\)-th position and insert it between \(x_{i-1}\) and \(x_i\),
2. if $j - i > \ell - k$, remove $x_k$ from the $k$-th position and insert it between $x_\ell$ and $x_{\ell+1}$.

With no loss of generality, let $j - i \leq \ell - k$. The portion of $S$ under consideration is therefore:

$$
(r_{i-1}, s_{i-1}) \underbrace{x_i (r_{i}, s_{i}) \cdots (r_{j-2}, s_{j-2}) x_{j-1}}_{\text{all in } U \cup \{w\}} (r_{j-1}, s_{j-1}) \overbrace{x_j (r_j, s_j)}^{\text{all in } U \cup \{w\}}
$$

and the order of all the vertices in the new $S'$ is:

$$
\underbrace{x_1 \cdots x_{i-1}}_{\text{all in } V \setminus \{w\}} \underbrace{x_j}_{\text{all in } U \cup \{w\}} \underbrace{x_i \cdots x_{j-1}}_{\text{all in } U \cup \{w\}} \underbrace{x_{j+1}}_{\text{all in } V \setminus \{w\}} \cdots \underbrace{x_{k-1}}_{\text{all in } V \setminus \{w\}} \underbrace{x_k}_{\text{all in } U \cup \{w\}} \underbrace{x_{k+1} \cdots x_{\ell}}_{\text{all in } U \cup \{w\}} \overbrace{x_{\ell+1} \cdots x_{p+n}}^{\text{all in } V \setminus \{w\}}
$$

Let $x_j = v \in V \setminus \{w\}$ and partition $d = \text{degree}_G(v)$ into $d = d^L + d^R$, where:

- $d^L$ is the number of vertices in $\{x_1, \ldots, x_{j-1}\} \cap V$ which are connected to $v$,
- $d^R$ is the number of vertices in $\{x_{j+1}, \ldots, x_{n+p}\} \cap V$ which are connected to $v$.

There are different cases, depending on:

- the value of $j - i$ between 1 and $p/2$,
- the value of $d^R - d^L$ between $-d$ and $+d$,
- whether the distinguished vertex $w$ is in $\{x_i, \ldots, x_{j-1}\}$ or in $\{x_{j+1}, \ldots, x_{\ell}\}$,
- whether $v$ is connected to $w$ or not.

We consider only one of the cases, which is also a “worst case” to explain, and leave to the reader all the other cases, which are simple variations of this “worst case”. For the “worst case” which we choose to consider, let:

1. $j = i + p/2$ so that $j - i = p/2$,
2. $d^L = 0$ so that $d^R - d^L = d$,
3. $w$ is in $\{x_{j+1}, \ldots, x_{\ell}\}$,
4. there is an edge $\overline{v \cdot w}$ connecting $v$ and $w$.

With assumptions (1) to (4), as well as after:

- substituting $i + (p/2)$ for $j$,
- replacing the sequence of cutwidths $s_{i-1}, s_i, \ldots, s_{i+(p/2)-1}, s_{i+(p/2)}, s_{i+(p/2)+1}$ by their actual values $0, p, \ldots, (p^2 + 2p - 8)/4, (p^2 + 2p)/4, (p^2 + 2p)/4$, respectively,
- and posing $r := r_{i-1}$,

the portion of $S$ under consideration becomes:

$$
\underbrace{(r, 0) x_i (r, p) \cdots (r, (p^2 + 2p - 8)/4)}_{\text{all in } U \cup \{w\}} x_{i+(p/2)} \underbrace{(r, (p^2 + 2p)/4)}_{\text{all in } U \cup \{w\}} \overbrace{w}^{\text{all in } U \cup \{w\}} (r + d, (p^2 + 2p)/4)
$$
The corresponding portion in the new $S'$ is:

$$(r, 0) \quad \bigcirc \quad (r + d, 0)$$

$$x_i \quad (r + d, p) \quad \cdots \quad (r + d, (p^2 + 2p - 8)/4) \quad x_i + (p/2) \quad (r + d, (p^2 + 2p)/4)$$

with all the cutwidths to the left and to the right of the shown portion being identical in $S$ and $S'$. It is now readily seen that the value of $\beta(S')$ is:

$$\beta(S') = \beta(S) - (p^2 + 2p)/4 + (p/2) d = \beta(S) - \frac{p^2 - 2(d - 1)p}{4}$$

Let $\Delta := \sum \{ \text{degree}_G(x) \mid x \in V \}$. By the construction of the auxiliary graph $G_w$, we have $p = \Delta$. The value of $d = \text{degree}_G(v) \leq \Delta/2$, the upper bound $\Delta/2$ being the extreme case when $G$ is a star graph with: $v$ at its center, $\Delta/2$ leaf vertices among $\{x_{j+1}, \ldots, x_{n+p}\} \cap V$, and all other vertices of $V$ being isolated. Hence,

$$p^2 - 2(d - 1)p \leq \Delta^2 - 2 \left( \frac{\Delta}{2} - 1 \right) \Delta = \Delta^2 - \Delta^2 + 2\Delta = 2\Delta.$$ 

Hence, $\beta(S') \leq \beta(S) - \Delta/2$, so that $\beta(S') < \beta(S)$ which is the desired conclusion.

For precision in the next proof, we introduce the measure of unbalance.

**Definition 43** (*Unbalance*) Consider the sequence $S$ in (♦) in Sect. 4.

- Let $a^L \geq 0$ be the number of vertices from $V - \{w\}$ to the left of $w$, and $a^R \geq 0$ be the number of vertices from $V - \{w\}$ to the right of $w$.
- Let $b^L \geq 0$ be the number of vertices from $U$ to the left of $w$, and $b^R \geq 0$ be the number of vertices from $U$ to the right of $w$.

The unbalance of $S$ is measured by:

$$\text{unbal}(S) := \min \{ (n - a^L - 1) + (p - b^R), (n - a^R - 1) + (p - b^L) \}. $$

The quantity $(n - a^L - 1) + (p - b^R)$ measures $S'$'s unbalance on the left, and similarly $(n - a^R - 1) + (p - b^L)$ measures $S'$'s unbalance on the right. It is useful to keep in mind that:

$$(p - b^R) + (p - b^L) = p \quad \text{and} \quad (n - a^L - 1) + (n - a^R - 1) = n - 1,$$

so that, if the quantity $(n - a^L - 1) + (p - b^R)$ or the quantity $(n - a^R - 1) + (p - b^L)$ is reduced to 0, then the other of these two quantities is increased to $n - 1 + p$. 

$\blacksquare$
Proof of Lemma 31  By Lemma 30, we can assume that $S$ is not scattered. It suffices to show that if $\text{unbal}(S) \geq 1$, we can define another sequence $S'$ from $S$ such that $\beta(S') < \beta(S)$ and $\text{unbal}(S') < \text{unbal}(S)$. We use the notation in the proof of Lemma 30. The portion of $S$ that we examine closely is:

$$
(r_{i-1}, s_{i-1}) \ x_i (r_{i}, s_i) \ x_{i+1} (r_{i+1}, s_{i+1}) \ldots (r_{i+p-1}, s_{i+p-1}) \ x_{i+p} (r_{i+p}, s_{i+p})
$$

where:

$$V - \{w\} = \{x_1, \ldots, x_{i-1}\} \cup \{x_{i+p+1}, \ldots, x_n\}, \quad \text{with} \quad 1 \leq i \leq n,$n$$

$$U \cup \{w\} = \{x_i, \ldots, x_{i+p}\}, \quad \text{with} \quad w = x_{i+k} \quad \text{and} \quad 0 \leq k \leq p.$$

We partition $d = \deg_G(x_{i+k}) = \deg_G(w)$ into $d = d^L + d^R$, where:

- $d^L \geq 0$ is the number of vertices in $\{x_1, \ldots, x_{i-1}\}$ which are connected to $w = x_{i+k}$ in $G$,
- $d^R \geq 0$ is the number of vertices in $\{x_{i+p+1}, \ldots, x_n\}$ which are connected to $w = x_{i+k}$ in $G$.

And we partition $e = \deg_{K_p+1}(x_{i+k}) = \deg_{K_p+1}(w)$ into $e = e^L + e^R = p$, where:

- $e^L = k$ is the number of vertices in $\{x_i, \ldots, x_{i+k-1}\}$ which are connected to $w = x_{i+k}$ in $K_{p+1}$,
- $e^R = (p - k)$ is the number of vertices in $\{x_{i+k+1}, \ldots, x_{i+p}\}$ which are connected to $w = x_{i+k}$ in $K_{p+1}$.

Though not explicitly used below, it is worth noting that $e^L$ and $e^R$ here are the same as $b^L$ and $b^R$ in Definition 43 because $K_{p+1}$ is a complete graph (but $d^L$ and $d^R$ are not the same as $a^L$ and $a^R$). We consider 5 separate cases, $\{(a), (b), (c), (d), (e)\}$:

(a) $k = 0$, which implies $e^L = 0$ and $e^R = p$.

In case (a), because $\text{unbal}(S) \neq 0$ by hypothesis, it must be that $\{x_{i+p+1}, \ldots, x_n\} \neq \emptyset$. It suffices to move the vertices in $\{x_{i+p+1}, \ldots, x_n\}$ to the left of $w = x_i$, also preserving their order

$$x_1, \ldots, x_{i-1}, x_{i+p+1}, \ldots, x_n.$$

Using a reasoning similar to that in the proof of Lemma 30, we leave it to the reader to show that $\text{unbal}(S') = 0$ and $\beta(S') < \beta(S)$ for the resulting sequence $S'$.

(b) $k = p$, which implies $e^L = p$ and $e^R = 0$.

Case (b) is similar to case (a). Because $\text{unbal}(S) \neq 0$ by hypothesis, it must be that $\{x_1, \ldots, x_{i-1}\} \neq \emptyset$. In this case, we move the vertices in $\{x_1, \ldots, x_{i-1}\}$ to the right of $w = x_{i+p}$. Again, we leave it to the reader to show that $\text{unbal}(S') = 0$ and $\beta(S') < \beta(S)$ for the resulting sequence $S'$.

For the three remaining cases, we can assume that neither $k = 0$ nor $k = p$, i.e., both $w \neq x_i$ and $w \neq x_{i+p}$. Two cases of these three are:
(c) \( d_L > d_R \), in which case we transpose \( w = x_{i+k} \) and \( x_i \).

(d) \( d_L < d_R \), in which case we transpose \( w = x_{i+k} \) and \( x_{i+p} \).

By a reasoning similar to that in the proof of Lemma 30, we leave to the reader the straightforward details showing that \( \beta(S') < \beta(S) \) in both case (c) and case (d).

(e) \( d_L = d_R \), in which case the value of \( \beta(S) \) remains unchanged by tranposing \( w = x_{i+k} \) and \( x_i \), or by tranposing \( w = x_{i+k} \) and \( x_{i+p} \), and so we need an additional argument.

The additional argument for case (e), is to first transpose \( w = x_{i+k} \) and \( x_i \), or alternatively transpose \( w = x_{i+k} \) and \( x_{i+p} \), thus reducing case (e) to case (a), or alternatively reducing case (e) to case (b). \( \square \)

References

Arora S, Frieze A, Kaplan H (1996) A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. In: IEEE Proceedings of the 37th Annual Symposium on Foundations of Computer Science, pp 21–30

Assaf K (2014) The syntax and semantics of a domain-specific language for flow-network design. Sci Comput Progr 93((part A)):19–38

Bestavros A, Kfoury A (2011) A domain-specific language for incremental and modular design of large-scale verifiably-safe flow networks. In: Proceeding of IFIP Working Conference on Domain-Specific Languages (DSL 2011), EPTCS volume 66, pp 24–47

Bronner D, Ries B (2006) An introduction to treewidth. Technical Report ROSE-REPORT-2006-004, Ecôle Polytechnique Fédérale de Lausanne

Burkhard M, Ivan Hal S (1988) Min cut is NP-complete for edge weighted trees. Theor Comput Sci 58(1–3):209–229

Charikar M, Hajiaghayi MT, Karloff H, Rao S (2006) \( \ell^2 \) Spreading metrics for vertex ordering problems. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, pp 1018–1027. Society for Industrial and Applied Mathematics

Deistel R (2012) Graph theory. Springer, Berlin

Garey Michael R, Johnson David S, Larry Stockmeyer (1976) Some simplified NP-complete graph problems. Theor Comput Sci 1:237–267

Jordi P (2011) Addenda to the survey of layout problems. Bull EATCS 105:177–201

Kfoury A (2011) The denotational, operational, and static semantics of a domain-specific language for the design of flow networks. In: Proceedings of SBLP 2011: Brazilian Symposium on Programming Languages

Kfoury A, Mirzaei S (2013) A different approach to the design and analysis of network algorithms. Technical Report BUCS-TR-2012-019, CS Dept, Boston University

Kfoury A, Mirzaei S (2015) Efficient reassembling of graphs, part 1: the linear case. arXiv:1509.06434

Makedon Fillia S, Papadimitriou Christos H, Hal SI (1985) Topological bandwidth. SIAM J Algebraic Discret Methods 6(3):418–444

Odlyzko AM (1995) Asymptotic enumeration methods. In: Graham RL, Grotschel M, Lovast L (eds) Handbook of combinatorics, vol 2, pp 1063–1229. Cambridge, MA, MIT Press

Rao S, Richa AW (1998) New approximation techniques for some ordering problems. In: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete algorithms, pp 211–218. Philadelphia, PA, Society for Industrial and Applied Mathematics

Soule N, Bestavros A, Kfoury A, Lapets A (2011) Safe compositional equation-based modeling of constrained flow networks. In: Proceedings of 4th Int’l Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Zürich

Thilikos DM, Serna MJ, Bodlaender HL (2001) A polynomial time algorithm for the cutwidth of bounded degree graphs with small treewidth. In: Meyer F, Auf der H (eds), Algorithms, ESA 2001. LNCS 2161, pp 380–390. Springer, Berlin

Tom L, Satish R (1999) Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. J ACM (JACM) 46(6):787–832