Correlation Clustering and Two-edge-connected Augmentation for Planar Graphs

Philip N. Klein*†1, Claire Mathieu†2, and Hang Zhou†3

1 Brown University, United States
klein@brown.edu
2 CNRS, École Normale Supérieure, France
cmathieu@di.ens.fr
3 École Normale Supérieure, France
hangzhou@di.ens.fr

Abstract

In correlation clustering, the input is a graph with edge-weights, where every edge is labelled either + or − according to similarity of its endpoints. The goal is to produce a partition of the vertices that disagrees with the edge labels as little as possible.

In two-edge-connected augmentation, the input is a graph with edge-weights and a subset \( R \) of edges of the graph. The goal is to produce a minimum weight subset \( S \) of edges of the graph, such that for every edge in \( R \), its endpoints are two-edge-connected in \( R \cup S \).

For planar graphs, we prove that correlation clustering reduces to two-edge-connected augmentation, and that both problems have a polynomial-time approximation scheme.

1 Introduction

1.1 Correlation Clustering

Correlation clustering takes as input a graph whose edges are labelled either (+) or (−). A (+) edge represents evidence that its endpoints belong in the same cluster, and a (−) edge represents evidence that its endpoints belong in different clusters. Each edge has a non-negative weight reflecting the strength of the evidence. The goal is to find a clustering minimizing the total weight of edges inconsistent with that evidence. This formulation, previously from computational biology [11], was introduced by Bansal, Blum, and Chawla [8]. They suggested as an application the clustering of documents into topics.

In this paper, we study the case when the graph is planar. The motivation comes from image segmentation. The goal is to partition the image into regions representing different image components. An image is represented by a grid of pixels. For each pair of neighboring pixels, comparing the pixels’ values yields an assessment of how likely the pixels are to belong
Correlation Clustering and Two-edge-connected Augmentation for Planar Graphs

Figure 1 In this unweighted grid, every solid (resp. dashed) edge represents a pair of similar (resp. dissimilar) pixels. Dotted lines indicate an optimal partition with inconsistent edges $e_1$, $e_2$, $e_3$.

...to the same region. There can be spurious assessments. So global optimization is needed to find a good segmentation. See Figure 1. When an image is large, it is common for a visual task to first coalesce coherent uniform neighborhoods of pixels into superpixels, using preprocessing based on local properties such as brightness, color, and texture, see [3, 35]. We then extract a local similarity measure on pairs of adjacent superpixels, and the goal is to find a good segmentation of the superpixel graph under that measure. To achieve this, researchers used correlation clustering as the formulation [4, 5, 6, 30, 41]. They gave experimental results based on techniques such as integer linear programming or linear programming relaxation.

Note that the superpixel graph is planar. However, correlation clustering is NP-hard for planar graphs [7]. Prior to this work, the best result with theoretical guarantee was a constant-factor approximation for minor-excluded graphs by Demaine, Emanuel, Fiat, and Immorlica [17]. In this paper, we give a polynomial-time approximation scheme (PTAS).

Theorem 1. For any $\epsilon > 0$, there is a polynomial-time $(1 + \epsilon)$-approximation algorithm for correlation clustering in weighted planar graphs.

Related work

Why do we restrict ourselves to planar graphs? Because the general (weighted) problem is APX-hard [8]. Charikar, Guruswami, and Wirth [16] and independently Demaine, Emanuel, Fiat, and Immorlica [17] gave logarithmic-factor approximation algorithms. There have been improved approximation algorithms when the graph is complete [1, 8, 16]; or when, for each edge, the agreement weight and disagreement weight of that edge sum to one [1, 8]; or when, in addition, the weights satisfy the triangle inequality [25]. When the number of clusters is limited to a constant, Giotis and Guruswami [26] gave a PTAS. The problem was also studied in a planted model [36] and from the viewpoint of fixed-parameter tractability [10].

We discussed the problem of minimizing weight of disagreement; maximizing weight of agreement is equivalent at optimality but easier to approximate [8, 16, 40]. Researchers have also considered other objective functions [2].

1.2 Two-edge-connected Augmentation

In the field of telecommunications, an important task is to ensure that the network is resilient against single-link failures [39]. The two-edge-connected augmentation problem takes as input a graph $G$ with non-negative edge-weights and a subset $R$ of edges of the graph. The goal is...
to find a minimum-weight subset $S$ of edges of the graph such that for every edge $uv \in R$, $u$ and $v$ are two-edge-connected in the subgraph $R \cup S$.

This problem is a generalization of the well-studied tree augmentation problem: given a graph $G$ with edge-weights and given a spanning tree $T$ of $G$, find a minimum-weight set $S$ of edges such that for each edge $uv$ of $T$, $u$ and $v$ are 2-edge-connected in $T \cup S$. The condition is equivalent to requiring that for each edge $uv$ of $T$, $u$ and $v$ are 2-edge-connected in $T \cup S$. The tree augmentation problem is NP-hard [20, 23] and APX-hard [33]. Frederickson and Ja’Ja’ [23] gave a polynomial-time 2-approximation algorithm. The running time of that algorithm was improved by Khuller and Thurimella [28], and further by Galluccio and Proietti [24]. We give a PTAS for the generalized problem when the input is restricted to planar graphs.

**Theorem 2.** For any $\epsilon > 0$, there is a polynomial-time $(1 + \epsilon)$-approximation algorithm for two-edge-connected augmentation in weighted planar graphs.

### Related work

A closely related problem is two-edge-connected spanning subgraph, for which a constant-factor approximation algorithm was known [29]. When the graph is planar, Berger and Grigni [12] gave a PTAS. One might think that this would lead to a PTAS for our problem, but it is not the case because the weight of a two-edge-connected augmentation can be much smaller than the minimum weight of a two-edge-connected spanning subgraph. For the Steiner-type generalization of the two-edge-connected subgraph problem, there was a constant-factor approximation algorithms [32]. When the graph is planar, Borradaile and Klein [14] gave a PTAS.\(^1\)

There is a variety of other related work, see [34] for a survey. Some studied the special case when the weights are all one [21, 22, 37], or when, in addition, the graph is complete [20]. There was a 2-approximation algorithm for the related problem of augmenting a connected subgraph to achieve two-edge-connectivity among a pre-specified set of terminal vertices [38]. Edge-connectivity augmentation problems were subsumed by the work of Jain [27] on survivable network design.

### Techniques and Notations

Our techniques for proving Theorem 1 and Theorem 2 include planar duality, prize-collecting clustering, brick decomposition, sphere-cut decomposition, and dynamic programming.

Throughout the paper, we allow graphs to have parallel edges. For a graph $G$, we note $V[G]$ as its vertex set and $E[G]$ as its edge set. For a subset $H \subseteq E[G]$, we identify $H$ with the subgraph induced by edges from $H$. The weight of $H$ is defined by $\sum_{e \in H} \text{weight}(e)$. The boundary $\partial(H)$ is the set of vertices $u$ that are incident to some edge of $H$ and to some edge of $E[G] \setminus H$. Similarly, for a subset $U \subseteq V[G]$, its boundary $\partial(U)$ is the set of edges $uv$ such that $u \in U$ and $v \in V[G] \setminus U$. A plane graph is a planar graph together with a planar embedding. We use the phrases plane graph and planar graph interchangeably. We use $\text{OPT}(G, R)$ to denote the weight of the optimal two-edge-connected augmentation for $(G, R)$. The parameters $G$ and $R$ are omitted when they are clear from the context.

---

\(^1\) In their problem, a solution is allowed to include multiple copies of edges of the input graph.
3 Theorem 2 Implies Theorem 1

We address correlation clustering and two-edge-connected augmentation in one paper because of the reduction in Theorem 3, which shows that Theorem 2 implies Theorem 1.

**Theorem 3.** There is an approximation-preserving reduction from correlation clustering in a weighted planar graph to two-edge-connected augmentation in a weighted planar graph.

**Remark.** In practice, we may use an approximation algorithm for two-edge-connected augmentation that is different from the algorithm in Theorem 2, and then from the reduction (Theorem 3), we obtain an algorithm for planar correlation clustering with the same approximation factor.

**Lemma 4 (Bridge-Deletion Lemma).** Let $G$ be a plane graph. Let $R$ be a subset of $E[G]$. Let $S$ be a minimal two-edge-connected augmentation for $(G, R)$. Then every connected component in the subgraph $R \cup S$ is two-edge-connected.

**Proof.** Suppose there is a bridge edge $e$ in the subgraph $R \cup S$. Since $S$ is a two-edge-connected augmentation for $(G, R)$, $e$ cannot belong to $R$. Thus $S \setminus \{e\}$ remains a two-edge-connected augmentation, which contradicts with the minimality of $S$. ◁

**Proof of Theorem 3.** Let $G_0$ be a plane graph in the correlation clustering problem. Define $G_1$ to be the abstract dual of $G_0$. $G_0$ is also the abstract dual of $G_1$ and $E[G_0] = E[G_1].$\(^2\)

Define $R$ to be the duals of the $\langle - \rangle$ edges in $G_0$. Next, we construct the graph $G_2$ from $G_1$ by duplicating of the edges of $R$. The weights are preserved. The theorem follows from Lemma 5 and Lemma 6.

**Lemma 5.** A subset of edges $S$ is the set of disagreements of some clustering in $G_0$ if and only if the subgraph $R \oplus S$ of $G_1$ is a collection of two-edge-connected components.

**Proof.** ($\implies$) Let $\{V_i\}_i$ be a clustering of $G_0$ with disagreements $S$. It is easy to see that $R \oplus S = \bigcup_i \partial(V_i)$\(^3\). Since every $\partial(V_i)$ corresponds to a union of cycles in $G_1$, the subgraph $R \oplus S$ of $G_1$ corresponds to a collection of two-edge-connected components. ($\impliedby$) Suppose the subgraph $R \oplus S$ of $G_1$ is a collection of two-edge-connected components. For every face $F$ in that subgraph, its boundary is a cycle, thus corresponds to a cut $\partial(V_F)$ in $G_0$, where $V_F$ is the set of vertices of $G_0$ in the interior of this face. Thus $\{V_F\}$ for all faces $F$ is a clustering of $G_0$ with disagreements $S$. ◁

**Lemma 6.** Let $S$ be a minimal two-edge-connected augmentation for $(G_2, R)$. Then the subgraph $R \oplus S$ of $G_1$ is a collection of two-edge-connected components.

**Proof.** By the Bridge-Deletion Lemma (Lemma 4), every connected component in the subgraph $R \cup S$ of $G_2$ is two-edge-connected. Suppose, for the sake of contradiction, that there is a bridge edge $uv$ in the subgraph $R \oplus S$ of $G_1$. There is a $u$-to-$v$ path in the subgraph $R \cup S \setminus \{uv\}$ of $G_2$, and some edge $u'v'$ of this path does not belong to the subgraph $R \oplus S$ of $G_1$. We have $u'v' \in (R \cup S) \setminus (R \oplus S) = R \cap S$. In addition, $u'$ and $v'$ are connected in

---

\(^2\) See for example [18] for the definition of abstract dual and its properties.

\(^3\) This is because, for any edge $e$ belonging to some $\partial(V_i)$, since its endpoints are in different clusters, either $e$ is a $\langle - \rangle$ edge and has not been swapped (i.e., $e \in R$ and $e \notin S$), or $e$ is a $\langle + \rangle$ edge and has been swapped (i.e., $e \notin R$ and $e \in S$); and for any edge $e$ not belonging to any $\partial(V_i)$, since its endpoints are in the same cluster, either $e$ is a $\langle - \rangle$ edge and has been swapped (i.e., $e \in R$ and $e \in S$), or $e$ is a $\langle + \rangle$ edge and has not been swapped (i.e., $e \notin R$ and $e \notin S$).
In the example, $R = \{e_1, e_2, e_3, e_4\}$. The optimal two-edge-connected augmentation consists of edges $e_5$ and $e_6$. However, any Steiner tree connecting the edges of $R$ must include one of the edges $e_7$ and $e_8$, whose weight may be much higher than weight ($\{e_5, e_6\}$).

$R \cup S \setminus \{u'v'\}$ through the edge $uv$. However, since $S$ is minimal, $S$ cannot include the edge $u'v'$. Contradiction.

Thus we complete the proof of Theorem 3.

\section{Reduction to Instance with a Connected Skeleton}

Without loss of generality, we assume for the rest of the paper that the edges of $R$ have weight zero.

To prove Theorem 2, we focus on a related version (Theorem 7), where we are given in addition a connected subgraph $T$ that contains every edge of $R$. We defer the proof of Theorem 7 to later sections.

\textbf{Theorem 7 (Augmentation Theorem).} Let $G$ be a plane graph with edge-weights. Let $R$ be a subset of $E[G]$. Let $T$ be a connected subgraph of $G$ that contains every edge of $R$. For every $\epsilon > 0$, there is a polynomial-time algorithm Augment-Connected($G, R, T, \epsilon$) that computes a two-edge-connected augmentation $S$ for $(G, R)$ such that weight ($S$) $\leq (1 + \epsilon)OPT(G, R) + \epsilon^2 \cdot$ weight ($T$).

\textbf{Remark.} The connected subgraph $T$ is needed for the \textit{brick decomposition}, which is the first step to achieve the algorithm of the Augmentation Theorem.

In the rest of this section, we prove Theorem 2 using the Augmentation Theorem. One might consider connecting all edges of $R$ with a Steiner tree $T$, and then applying the Augmentation Theorem. However, $OPT$ could be much smaller than the minimum weight of a Steiner tree when the solution is not connected (see Figure 2). In that case, the upper bound given by the Augmentation Theorem would not imply an approximation scheme.

Fortunately, there is an algorithmic tool, called \textit{prize-collecting clustering}, due to Bateni, Hajiaghayi, and Marx [9], that addresses exactly this kind of obstacle. They used it in addressing the Steiner forest problem. They started with a 2-approximate solution, and used prize-collecting clustering to decompose the instance into subinstances. We use the same approach for two-edge-connected augmentation, see Algorithm 1.

First, find a two-approximate solution $Y$ and contract the edges of $Y$ and of $R$. Next, assign potentials to use prize-collecting clustering to find a forest $F$. Consider the subgraph of the uncontracted graph consisting of the edges of the forest $F$ together with the edges of the 2-approximate solution and the edges of $R$. This subgraph defines the connected components $T_1, \ldots, T_k$ as output.

Line 1 computes a 2-approximate solution using Jain’s algorithm [27] (which solves a much more general problem).
Correlation Clustering and Two-edge-connected Augmentation for Planar Graphs

Algorithm 1 REDUCE-TO-CONNECTED

Input: a weighted planar graph $G$ and a subset $R$ of edges, $\epsilon > 0$
Output: connected subgraphs $T_1, \ldots, T_k$
1: $Y \leftarrow$ two-edge-connected augmentation with weight at most $2 \cdot \OPT$
2: $(U_1, \ldots, U_k) \leftarrow$ two-edge-connected components of $R \cup Y$
3: Contract each component $U_i$ to build a new graph $\hat{G}$
4: For every $v \in \hat{G}$, let $\phi_v$ be $\epsilon^{-1}$ times the weight of the component corresponding to $v$
5: Do prize-collecting clustering on $\hat{G}$ and $\phi$, obtaining a forest $F$
6: Return the connected components $T_1, \ldots, T_k$ of the subgraph $F \cup R \cup Y$ of $G$

Lemma 8 (corollary from [27]). There is an algorithm that computes in polynomial time a two-edge-connected augmentation $Y$ for $(G, R)$ such that weight $(Y) \leq 2 \cdot \OPT$ and that every connected component in $R \cup Y$ is two-edge-connected.

Proof. By [27], there is polynomial time algorithm that gives a 2-approximation for the following problem: Given a multi-graph $G'$ with non-negative edge-costs, and requirements $r_{u,v} \in \mathcal{Z}$ for each pair $(u, v)$ of vertices, find a minimum-cost subgraph of $G'$ such that, for each pair $(u, v)$, the network has flow $r_{u,v}$ between $u$ and $v$.

We reduce the two-edge-connected augmentation problem with instance $(G, R)$ to this problem: for every edge $uv$ of $G$ that is in $R$, set $r_{u,v} = 2$ and create two edges between $u$ and $v$ in $G'$ with costs 0 and weight $(uv)$ respectively; and for every edge $uv$ of $G$ that is not in $R$, set $r_{u,v} = 0$ and create one edge between $u$ and $v$ in $G'$ with cost weight $(uv)$. It is easy to see that the two problems are equivalent. Thus we obtain a two-edge-connected augmentation $S$ for $(G, R)$ satisfying Property 1. We then apply Bridge-Deletion Lemma (Lemma 4) to get $S' \subseteq S$ and let $Y = S'$. Then $Y$ is a two-edge-connected augmentation satisfying both properties.

Line 5 uses prize-collecting clustering, which receives a graph with vertex-potentials $\phi_v$, and returns a forest $F$ of edges of weight at most $2 \sum_v \phi_v$. Since the sum of vertex-potentials is at most $2 \epsilon^{-1} \cdot \OPT$, the weight of $F$ is at most $4 \epsilon^{-1} \cdot \OPT$. Using essentially the same arguments as in [9], we obtain the following.

Theorem 9 (variant of Theorem 1.3 in [9]). Let $G$ be a plane graph with edge-weights. Let $R$ be a subset of $E(G)$. For fixed $\epsilon$, Algorithm 1 computes in polynomial time a set of connected subgraphs $T_1, \ldots, T_k$ with the following properties:

- $\bigcup_i T_i$ contains every edge of $R$.
- $\sum_i \text{weight}(T_i) \leq (4/\epsilon + 2)\OPT(G, R)$.
- $\sum_i \OPT(G_i, R_i) \leq (1 + \epsilon)\OPT(G, R \cap T_i)$

Proof of Theorem 2. The top-level algorithm of Theorem 2 is given in Algorithm 2. By the Augmentation Theorem (Theorem 7) and Property 1 of Theorem 9, the output is a two-edge-connected augmentation for $(G, R)$.

For each $i$, the weight of $S_i$ is at most $(1 + \epsilon/7)\OPT(G, R \cap T_i) + (\epsilon/7)^2 \cdot \text{weight}(T_i)$. Summing over $i$ and combining Properties 2 and 3 of Theorem 9, we infer that the weight of the output solution is at most $(1 + \epsilon)\OPT(G, R)$.


Algorithm 2 \texttt{Augment}

\textbf{Input:} a planar graph $G$, a subset of edges $R$, and $\epsilon > 0$

\textbf{Output:} two-edge-connected augmentation $S$ for $(G, R)$

1. $(T_1, \ldots, T_k) \leftarrow \texttt{Reduce-to-Connected}(G, R, \epsilon/7)$ \hfill $\triangleright$ Theorem 9
2. for $i \leftarrow 1$ to $k$
3. \hfill $S_i \leftarrow \texttt{Augment-Connected}(G, R \setminus T_i, T_i, \epsilon/7)$ \hfill $\triangleright$ Theorem 7
4. return $(\bigcup_i S_i) \setminus R$

5 Techniques for Proving the Augmentation Theorem

5.1 New Use of Brick Decomposition

For non-local problems in weighted planar graphs in which the weight of the optimal solution can be much smaller than the weight of the graph, the \textit{brick decomposition} technique of [15] has proved to be quite versatile: a planar embedded subgraph $M$ (called the \textit{mortar graph}) is selected, and the \textit{bricks} are the subgraphs of $G$ embedded in the faces of $M$ (see Section 6.1). The key is the following properties of $M$.

\textbf{Property 1:} $M$ has weight $O(OPT)$;

\textbf{Property 2:} There exists a near-optimal solution that crosses the boundary of each brick only a constant number of times.

Both properties are achievable for problems such as \textit{Steiner tree} [15], \textit{Steiner forest} [9], \textit{TSP} [13], and \textit{two-edge-connected survivability} [14] for the variant in which the solution is allowed to include multiple copies of edges of the input graph.

The main obstacle in applying this approach to \textit{two-edge-connected augmentation} is that Property 2 seems unachievable using the known brick-decomposition construction. We therefore use the mortar graph in a new way. We take additional care in the construction of the mortar graph because of the edges of $R$. As a consequence, instead of Property 2, we can show that, after a transformation\(^4\) of the instance, we have:

\textbf{Property 2’ (Structure Theorem):} There exists a near-optimal solution such that, for any brick and any two vertices $u, v$ on the boundary of the brick, there exists a $u$-to-$v$ Jordan curve inside the brick that intersects the near-optimal solution at only a constant number of points.\(^5\) See Figure 3.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{brick_decomposition.png}
\caption{The rectangle is a brick. The solid curves are parts of a near-optimal solution. The dashed curves illustrate the $u_1$-to-$v_1$ path and the $u_2$-to-$v_2$ path inside the brick.}
\end{figure}

\(^4\) The transformation is to add artificial copies of the brick boundaries. See Figure 4 in Section 6.

\(^5\) The constant depends on $\epsilon$.
5.2 Outline of Algorithm \textsc{Augment-Connected}

We use ideas from [15] which we now summarize:

1. Build a mortar graph of $G$ based on the connected skeleton $T$.
2. Do Breadth-First Search (BFS) on the dual of the mortar graph, and select a mod-$k$ residue $j$ such that edges whose levels are congruent to $j$ have total weight at most $1/k$ times the weight of the mortar graph.
3. Commit to including these edges in the ultimate solution; this decomposes the graph into subinstances each consisting of at most $k$ levels of bricks.
4. A planar graph consisting of only $k$ BFS levels has branchwidth $2k$, i.e., can be recursively decomposed into clusters of edges such that each cluster is bounded by at most $2k$ vertices. However, here we must diverge. Note that the branch decomposition obtained above has a special form: it is a sphere-cut decomposition, which means that each cluster of edges is precisely the set of edges enclosed by a Jordan curve $J$ that intersects no edges (and intersects a constant number of vertices) of the mortar graph. This is where Property 2' comes in: each segment of $J$ traversing a brick can be replaced with a curve that intersects a constant number of points of the near-optimal solution. This yields a new Jordan curve $J'$ that passes through a constant number of points of the near-optimal solution. Such structure enables us to design a dynamic program (DP), given in Section 8.

For each cluster of the sphere-cut decomposition, the DP enumerates all possibilities of the intersection points of the unknown near-optimal solution with the partially unknown Jordan curve $J'$. The DP also enumerates all possibilities of the connected structure of the part of the solution inside $J'$. See Section 8.2. Note that there may be some edges of the graph that are in the parent cluster but not in the child clusters (Figure 9), so the DP must do a bit of extra work to go from tables for the children to the table for the parent. See Section 8.3.

6 Structure Theorem

The Structure Theorem (Theorem 13) is the key to the polynomial-time performance of the dynamic program (Section 8). Before stating the theorem, we recall the definition and properties of brick decomposition from [15] in Section 6.1, and we illustrate the transformation of doubling brick boundaries in Section 6.2.

6.1 Mortar Graph and Brick Decomposition

► Definition 10 (Mortar Graph and Bricks, slight adaptation from [15]). Let $G$ be a plane graph with edge-weights. Let $R$ be a subset of $E(G)$. Let $M$ be a subgraph of $G$. For each face $F$ of $M$, we define a brick $B$ as the planar subgraph of $G$ embedded inside the face, including the boundary edges of $F$. We denote the interior of $B$ as the brick without the boundary edges of $F$. We call $M$ a mortar graph of $G$ if the boundary of every brick $B$, in counter-clockwise order, is the concatenation of four paths North$_B$, South$_B$, East$_B$, West$_B$ (the subscript $B$ is omitted when it is clear from the context), such that:

1. No edge of $R$ is in the interior of $B$, or on South$_B$, East$_B$, or West$_B$.  

Property 2' is proved by reducing nesting and adding boundary cycles. See Sections 6 and 7.
Doubling Brick Boundaries

The proof of the Structure Theorem applies to a modified version of the graph in which artificial copies of the South, East, and West brick boundaries are added (Figure 4), and zero-weight edges are added between corresponding vertices. We call this doubling these boundaries. Note that no edges of $R$ are duplicated (according to Property 1 of Definition 10). Let $H$ be the resulting graph.

Let $G$ be a plane graph with edge-weights. Let $M$ be its mortar graph. Let $P = p_0 \cdots p_\ell (\ell \geq 1)$ be any boundary of any brick $B$. The operation of doubling the boundary $P$ is defined as follows. For every $i \in [1, \ell - 1]$, create a copy $p_i'$ of $p_i$, and add the edge $p_ip_i'$ of

2. South$_B$ is a shortest path in $B$, and every proper subpath of North$_B$ is an almost shortest path in $B$, i.e., its weight is at most $(1 + \epsilon)$ times the weight of the shortest path between its endpoints in $B$;
3. There exists an integer $k = O(1/\epsilon^4)$ and vertices $s_0, s_1, \ldots, s_k$ ordered from west to east along South$_B$ such that, for any vertex $x$ on the segment $[s_i, s_{i+1})$ of South$_B$, the weight of the segment between $x$ and $s_i$ along South$_B$ is less than $\epsilon$ times the weight of the shortest path between $x$ and North$_B$ in $B$.

Lemma 11 (Brick-Decomposition Lemma, slight adaptation from [15]). Let $G$ be a planar graph with edge-weights. Let $R$ be a subset of $E[G]$. Let $T$ be a connected subgraph of $G$ that contains every edge of $R$. There is a polynomial-time algorithm that computes a mortar graph $M$ of $G$ such that:
1. $\text{weight}(M) = O(\text{weight}(T)/\epsilon)$;
2. $\sum_{\text{brick } B} \text{weight}(\text{East}_B \cup \text{West}_B) = O(\epsilon^2 \cdot \text{weight}(T))$.

Remark. There are slight differences between [15] and here:
- Property 1 of Definition 10 is added. It can be achieved by requiring that edges in $R$ may only appear on the North boundaries during the strip decomposition in [15].
- North and South are swapped in Property 2 of Definition 10. In fact, the distinction between North and South is not so important in the mortar graph construction in [15].
- The constant $k$ in Property 3 of Definition 10 is $O(1/\epsilon^4)$ instead of $O(1/\epsilon^3)$ in [15]. This is because Lemma 11 requires that the total weight of East and West boundaries is bounded by $O(\epsilon^2 \cdot \text{weight}(T))$ instead of by $O(\epsilon \cdot \text{weight}(T))$ in [15]. Thus we choose a supercolumn from every $O(1/\epsilon^4)$ columns, instead of from every $O(1/\epsilon^3)$ columns in [15].
Figure 5 The dashed path from \( u \) to \( v \) has few crossings with the modified solution (solid).

weight 0;\(^6\) Add the edge \( p_i p_{i+1} \) of the same weight as \( p_i p_{i+1} \) for every \( i \in [0, \ell - 1] \); Finally, for every edge \( uv \in (B \setminus P) \times P \), replace the edge \( uv \) by \( uv' \) of the same weight. We denote by \( P' \) the path \( p_0', p_1', \ldots, p_\ell' \). For the result of doubling the West, South, and East boundaries of the brick \( B \), see Figure 4. Note \( \text{West}' \), \( \text{South}' \), and \( \text{East}' \) as their corresponding copies.

Let \( H \) be the graph by doubling the West, South, and East boundaries of every brick in \( G \). From the construction, the mortar graph of \( H \) is inherited from that of \( G \).

\textbf{Lemma 12 (Boundary-Doubling Lemma).} A two-edge-connected augmentation for \((G, R)\) can be transformed into a two-edge-connected augmentation for \((H, R)\) in linear time without increasing the weight, and vice versa.

\textbf{Proof.} A solution of \( G \) can be transformed into a solution of \( H \) by including all edges \( vv' \) of weight 0. Reversely, a solution of \( H \) can be transformed into a solution of \( G \) by including the boundary edge \( p_i p_{i+1} \) of a brick if at least one of \( p_i p_{i+1} \) or \( p_i' p_{i+1}' \) is in the solution of \( H \). \( \blacksquare \)

\textbf{6.3 Theorem Statement}

\textbf{Theorem 13 (Structure Theorem).} Let \( G \) be a plane graph with edge-weights. Let \( R \) be a subset of \( E[G] \). Let \( M \) be the mortar graph of \( G \). Let \( H \) be the graph obtained from \( G \) by doubling the South, East, and West boundaries of every brick.

For any two-edge-connected augmentation \( S_0 \) for \((H, R)\), there exists a two-edge-connected augmentation \( S \) for \((H, R)\) such that:

- \( \text{weight}(S) \leq (1 + \varepsilon)\text{weight}(S_0) + 3 \sum_{\text{brick } B} \text{weight}(\text{East}_B \cup \text{West}_B) \);

- For any brick and any two vertices \( u, v \) on the boundary of the brick, there exists a \( u \)-to-\( v \) Jordan curve inside the brick that has \( O(1/\epsilon^4) \) crossings with \( S \), all occurring at vertices.

\textbf{6.4 Proof Sketch}

\textbf{Lemma 14 (Empty Cycle Lemma, [14]).} Let \( S \) be a two-edge-connected augmentation for \((H, R)\). Let \( C \) be a non-self-crossing cycle of \( S \) strictly enclosing no edge of \( R \). Deleting the edges of \( S \) that are strictly enclosed by \( C \) remains a two-edge-connected augmentation.

The proof of the Structure Theorem consists in modifying the initial solution so that any pair of vertices on the boundary of a brick can be connected by a curve that has few crossings.

\(^6\) Note \( p_0' := p_0 \) and \( p_\ell' := p_\ell \) to simplify the notation.
Figure 6 Reducing nesting of the solution: if there are more than $\lceil 1/\epsilon \rceil$ nested paths (left figure), add a piece of the South boundary (the bold segment in the right figure) and empty the cycle thus created. The same operation is applied to nested paths connected to the North boundary, with the caveat that edges of $R$ need not be added to the solution (since the solution is supposed to be an augmentation of $R$).

Figure 7 Adding South cycles (the bold cycles in the right figure) into the solution. The number of South cycles needed is $O(1/\epsilon^4)$ due to Property 3 in the definition of bricks.

with the modified solution. Figure 5 shows the kind of curve we use. It starts at a given vertex $u$ on the brick boundary, traverses nested paths to reach the South boundary, then bypasses South-to-North paths using cycles formed by the duplicated edges of the South boundary, and finally again traverses nested paths to reach the given vertex $v$ on the brick boundary. In order to have a small number of crossings, we must ensure that the number of nested paths is small and that only a small number of South cycles are used to bypass the South-to-North paths. This is illustrated in Figures 6 and 7.

The construction of the solution $S$ works on each brick in turn, modifying the initial solution $S_0$ inside that brick: adding the East and West cycles (i.e., the East and West boundaries together with their duplicates), reducing nesting as in Figure 6, and adding South cycles (i.e., parts of South together with their duplicates) as in Figure 7.

7 Complete Proof of Structure Theorem

The proof is mainly based on the following lemma.

Lemma 15. Let $S$ be any two-edge-connected augmentation for $(H, R)$. Let $B$ be a brick in $H$. Let $F$ be the set of edges of $S$ that are in the interior of $B$. Then there exists a set of edges $F_3$ in $B$, such that
1. $(S \setminus F) \cup F_3$ is a two-edge-connected augmentation for $(H, R)$;
2. weight $(F_3) \leq (1 + 5\epsilon)(\text{weight } F) + 2\text{weight } (\text{East } \cup \text{West})$;
3. For any two vertices $u, v$ on the boundary of $B$, there exists a $u$-to-$v$ Jordan curve inside the brick that has at most $O(1/\epsilon^4)$ crossings with $F_3$, all occurring at vertices.

Proof of Theorem 13 using Lemma 15. Let $S$ be initialized as $S_0$. For each brick $B$ in turn, let $F$ be the set of edges of $S$ that are in the interior of $B$. We apply Lemma 15 and
Correlation Clustering and Two-edge-connected Augmentation for Planar Graphs

update $S$ by $(S \setminus F) \cup F_3$. At each step, $S$ remains a two-edge-connected augmentation. The final weight of $S$ is increased from the weight of $S_0$ by $\sum_{\text{brick } B} (\text{weight } (F_3) - \text{weight } (F))$, which is at most $5\epsilon \cdot \text{weight } (S_0) + 2(1 + 5\epsilon) \sum_{\text{brick } B} \text{weight } (\text{East } B \cup \text{West } B)$. Theorem 13 follows by replacing $\epsilon$ by $\epsilon' = \epsilon/5$.

The rest of the section of to prove Lemma 15.

### 7.1 Construction of $F_3$

There are three steps to modify $F$ into $F_3$.

**Step 1: Modify $F$ into $F_1$ by adding East and West cycles.** We add to $F$ the edges of the two cycles East $\circ$ East' and West $\circ$ West' (see Figure 4) and remove from $F$ all edges inside the two cycles. Prune the result by removing from $F$ the edges in the interior of $B$ that are unnecessary, thus obtaining a forest.

Let $F_1$ be the result.

**Step 2: Modify $F_1$ into $F_2$ by reducing nesting.** A South arch $A$ is a path whose endpoints $u$ and $v$ are on South and whose other vertices are all strictly in the interior of the brick. The $u$-to-$v$ path along South is called the base of $A$. For a South arch $A$ from $F_1$, the South arch-emptying operation is to add to $F_1$ the base of $A$ (thus closing a cycle), and to remove from $F_1$ the edges strictly inside that cycle.

Consider every maximally enclosing arch and define its South depth to be 0. Let $\kappa = [1/\epsilon]$. For each $d = 0, 1, \ldots, \kappa - 1$, for each arch $A$ of depth $d$, consider every maximally enclosing arch whose edges are strictly inside $A$, and define its depth to be $d + 1$. Apply the arch-emptying operation to every South arch at depth $\kappa$. See Figure 6.

We similarly define North arches, North depths, and North arch-emptying operation, except that the North boundary may contain edges from $R$, so in the North arch-emptying operation, instead of adding the $u$-to-$v$ path along North, we add the edges of the $u$-to-$v$ path along North that are not in $R$.

Let $F_2$ be the result of South and North arch-emptying operations.

**Step 3: Modify $F_2$ into $F_3$ by adding South cycles.** First, we greedily define a collection of disjoint South-to-North paths $P_0, \ldots, P_t$ (with associated indexes $k_0, \ldots, k_t$), see Figure 8. We follow the approach in [15]: Let $s_0, \ldots, s_k$ be the vertices of South in Definition 10. Let $P_0$ be the East boundary of the brick. For each $i \geq 0$, let $k_i$ be the integer such that the start vertex of $P_i$ belongs to $[s_{k_i}, s_{k_i+1})$, and let $P_{i+1}$ be the easternmost path in $F_2$ from $[s_0, s_{k_i})$ to North that does not go through any vertices of South or North or $P_i$. Let $t$ be the last index for which $P_t$ is defined. Note that $t \leq k = O(1/\epsilon^4)$.

For each $i \geq 0$, let $x_i$ be the start vertex of $P_i$. Add to $F_2$ the cycle $C_i = [s_{k_i}, x_i] \circ x_i x'_i \circ [x'_i, s'_i] \circ s'_i s_{k_i}$, which is called a South cycle. See Figure 7. We remove from $F_2$ all edges inside $C_i$, thus all vertices of $C_i$ are on the same face. Let $F_3$ be the result.

### 7.2 Feasibility (Property 1 of Lemma 15)

By Lemma 14 (empty cycle lemma), the removals of edges inside cycles preserve feasibility, so $(S \setminus F) \cup F_3$ is a two-edge-connected augmentation for $(H, R)$.

### 7.3 Bounding the weight (Property 2 of Lemma 15)

The following lemma bounds the cost of arch-emptying operations with respect to one maximally enclosing arch, and is the key to bounding the weight.
Then weight (by Lemma 6.10 in [15]) and enclosing arch in Thus we have: which is at most Combining Equations (1), (2), (3), we obtain the bound on weight ($F'$) as in Lemma 15.

\[\sum_{F \in A} \text{weight}(F) \leq (1 + 2\epsilon) \cdot \text{weight}(F_1(A)).\]

Thus $\sum_{A \in A}$ weight $(A) \leq 2\epsilon \cdot \text{weight}(F_1(A))$ as required. ▶

Now we bound the weight of $F_3$. Since $F_1$ is obtained by adding the East and West boundaries and their copies East’ and West’, we have:

\[\text{weight}(F_1) \leq \text{weight}(F) + 2\text{weight} (\text{East}_B \cup \text{West}_B).\] (1)

Note that every edge of $F_1$ belongs to at most one $F_1(A)$ among all maximally enclosing arches $A$. Applying Lemma 16 to each maximally enclosing arches, and summing them up, we have:

\[\text{weight}(F_2) \leq (1 + 2\epsilon) \cdot \text{weight}(F_1).\] (2)

Finally, since the weight of every South cycle $C_i$ is $2 \cdot \text{weight} ([s_k, x_i]) \leq 2\epsilon \cdot \text{weight}(P_i)$ (by Lemma 6.10 in [15]) and $\sum_i \text{weight}(P_i) \leq \text{weight}(F_2)$, we have:

\[\text{weight}(F_3) \leq (1 + 2\epsilon)\text{weight}(F_2).\] (3)

Combining Equations (1), (2), (3), we obtain the bound on weight ($F'$) as in Lemma 15.

\[\sum_{A \in A} \text{weight}(A) \leq 2\epsilon \cdot \text{weight}(F_1(A)).\]
7.4 Bounding the Number of Crossings (Property 3 of Lemma 15).

Definition 17. For any two vertices \( u, v \) of the brick, consider the \( u \)-to-\( v \) Jordan curve inside the brick that has minimum number of crossings with \( F_3 \), all occurring at vertices. Define the distance measure \( \text{dist}(u, v) \) as the number of vertices (including \( u, v \)) on this curve minus 1.\(^7\) Let \( X, Y \) be subsets of vertices of the brick. Define \( \text{dist}(u, X) = \min_{v \in X} \text{dist}(u, v) \) and \( \text{dist}(X, Y) = \min_{u \in X, v \in Y} \text{dist}(u, v) \).

We use the notation \( \text{dist}_{F_2}(u, v) \) to denote the distance with respect to \( F_2 \) instead of \( F_3 \).

The following lemma is the key to prove Property 3 of Lemma 15.

Lemma 18. For any vertices \( u, v \) on South \( \cup \) North, \( \text{dist}(u, v) = O(1/\epsilon^4) \).

Proof of Property 3 of Lemma 15 using Lemma 18. The property can be rewritten as follows: for every vertices \( u, v \) on the boundary of the brick, \( \text{dist}(u, v) = O(1/\epsilon^4) \).

Let \( u, v \) be any boundary vertices. Let \( u_0 \) (resp. \( v_0 \)) be the vertex on South \( \cup \) North which minimizes \( \text{dist}(u, u_0) \) (resp. \( \text{dist}(v, v_0) \)).\(^8\) Then \( \text{dist}(u, v) \leq \text{dist}(u, u_0) + \text{dist}(u_0, v_0) + \text{dist}(v_0, v) \). From Step 1 of the construction, \( u \) and \( u_0 \) (resp. \( v \) and \( v_0 \)) are on the same face, thus within distance 1 from each other. By Lemma 18, \( \text{dist}(u_0, v_0) = O(1/\epsilon^4) \). So \( \text{dist}(u, v) = O(1/\epsilon^4) \).

To prove Lemma 18, we need Fact 19, Lemmas 21 and 22. Lemma 20 is used to prove Lemmas 21 and 22.

Fact 19. For every \( u, v \), \( \text{dist}(u, v) \leq \text{dist}_{F_2}(u, v) + 2 \).

Lemma 20. Let \( A \) be an arch in \( F_2 \). Let \( u \) be a vertex on the base of \( A \). Then \( \text{dist}_{F_2}(u, A) = O(1/\epsilon) \).

Proof. Consider a set of arches \( \{A_i\}_{1 \leq i \leq \ell} \) in \( F_2 \), where \( A_0 = A \), and every \( A_i \) \((i \geq 1)\) is the maximal enclosing arch whose edges are strictly inside \( A_{i-1} \) and whose base contains \( u \). Let \( \ell \) be the last index for which \( A_\ell \) is defined. From Step 2 of the construction, we know \( \ell = O(1/\epsilon) \). Let \( u_\ell \in A_\ell \) be the vertex that is on the same face with \( u \) in \( F_2 \). For every \( 1 \leq i \leq \ell \), let \( u_{i-1} \in A_{i-1} \) be the vertex that is on the same face with \( u_i \) in \( F_2 \) (such vertex exists by the definition of \( \{A_i\} \)). Thus we obtain \( u_0 \in A \) with \( \text{dist}_{F_2}(u, u_0) \leq \ell = O(1/\epsilon) \).

Lemma 21. For every vertex \( u \in \text{North} \), \( \text{dist}_{F_2}(u, \text{South}) = O(1/\epsilon) \).

Proof. If \( u \) is not on the base of any North arch, it is easy to see that there exists some vertex on South that is on the same face with \( u \) in \( F_2 \), i.e., \( \text{dist}_{F_2}(u, \text{South}) = 1 \).

If \( u \) is on the base of some North arch, let \( A \) be the maximally enclosing North arch in \( F_2 \) whose base contains \( u \). By Lemma 20, there exists some vertex \( v \) on the arch \( A \) such that \( \text{dist}_{F_2}(u, v) = O(1/\epsilon) \). On the other hand, it is easy to see that there exists some vertex on South that is on the same face with \( v \) in \( F_2 \). Thus \( \text{dist}_{F_2}(u, \text{South}) = O(1/\epsilon) \).

Lemma 22. For every \( i \leq t \), \( \text{dist}(C_{i-1}, C_i) \leq 3 \); and for every vertex \( u \in [s_{k_i}, s_{k_{i-1}}) \), \( \text{dist}(u, C_i) = O(1/\epsilon) \).

---

\(^7\) When \( u = v \), \( \text{dist}(u, v) = 0 \).

\(^8\) If \( u \) (resp. \( v \)) is on South \( \cup \) North, then \( u_0 \) (resp. \( v_0 \)) equals \( u \) (resp. \( v \)).
Proof. First, we show that for every $i$, $\text{dist}(C_{i-1}, C_i) \leq 3$. The non-trivial case is when $F_2$ has a non-trivial path $Q$ from $P_i$ to the South segment $(x_i, x_{i-1})$. See Figure 8. Let $q_i$ and $r_i$ be the start and end vertices of $Q$. Let $A$ be the arc in $F_2$ consisting of the part of $P_i$ between $x_i$ and $q_i$ and the path $Q$. Then $\text{dist}(C_i, C_{i-1}) \leq \text{dist}(C_i, x_i) + \text{dist}(x_i, r_i) + \text{dist}(r_i, C_{i-1})$. By definition, $x_i \in C_i$. Since $s_{k_{i-1}}$ is on the base of $A$, $r_i \in C_{i-1}$. So $\text{dist}(C_i, C_{i-1}) \leq \text{dist}(x_i, r_i) \leq \text{dist}_{F_2}(x_i, r_i) + 2$ by Fact 19. Note that $x_i$ and $r_i$ are on the same face in $F_2$, because $A$ does not have non-trivial paths connecting to South. So $\text{dist}(C_i, C_{i-1}) \leq 3$.

For every vertex $u \in [s_{k_i}, x_i]$, $\text{dist}(u, C_i) = 0$. Every vertex $u \in (x_i, s_{k_{i-1}})$ is on the base of $A$. Thus by Lemma 20, $\text{dist}_{F_2}(u, A) = O(1/\epsilon)$. Since $A$ and $C_i$ share the vertex $x_i$ in $F_3$, we have $\text{dist}(u, C_i) \leq \text{dist}(u, A) + 1 \leq \text{dist}_{F_2}(u, A) + 3 = O(1/\epsilon)$.

**Proof of Lemma 18.** Let $u, v$ be any vertices on South $\cup$ North. Let $u_1$ (resp. $v_1$) be the vertex on South which minimizes $\text{dist}_{F_2}(u, u_1)$ (resp. $\text{dist}_{F_2}(v, v_1)$). Then $\text{dist}(u, v) \leq \text{dist}(u, u_1) + \text{dist}(u_1, v_1) + \text{dist}(v_1, v)$. By Fact 19, $\text{dist}(u, u_1) \leq \text{dist}_{F_2}(u, u_1) + 2$, $\text{dist}(v, v_1) \leq \text{dist}_{F_2}(v, v_1) + 2$. By Lemma 21, $\text{dist}_{F_2}(u, u_1)$ and $\text{dist}_{F_2}(v, v_1)$ are $O(1/\epsilon)$. So we only need to show that $\text{dist}(u_1, v_1) = O(1/\epsilon^4)$.

Let $i \leq t$ be such that $u_1 \in [s_{k_i}, s_{k_{i-1}}]$, and let $j \leq t$ be such that $v_1 \in [s_{k_j}, s_{k_{j-1}}]$. Since any two vertices of $C_i$ (resp. $C_j$) are on the same face (i.e., at distance at most 1 from each other), $\text{dist}(u_1, v_1) \leq \text{dist}(u_1, C_i) + \text{dist}(C_i, C_j) + \text{dist}(C_j, v_1) + 2$. By Lemma 22, $\text{dist}(u_1, C_i) = O(1/\epsilon)$, and $\text{dist}(v_1, C_j) = O(1/\epsilon)$. Using Lemma 22, we have:

$$\text{dist}(C_i, C_j) \leq \sum_{\ell = \max(i,j)+1}^{\max(i,j)} \text{dist}(C_{\ell-1}, C_{\ell}) + |j - i| \leq 4|j - i| = O(1/\epsilon^4).$$

Thus $\text{dist}(u_1, v_1) = O(1/\epsilon^4)$.

### 8 Dynamic Programming

In this section, we design a dynamic program (Theorem 23) to solve the two-edge-connected augmentation problem for $(H, R)$ in the special case where the dual of the mortar graph has bounded diameter. From the Structure Theorem, in order to get a near-optimal solution, we may restrict attention to solutions that satisfy the property defined there. A dynamic program computes the best among all such solutions.

**Theorem 23 (Dynamic Programming Theorem).** Let $R, M, H$ be defined as in the Structure Theorem (Theorem 13). Assume, in addition, that the dual graph of $M$ has diameter $O(1/\epsilon^2)$. There is an algorithm that computes in polynomial time a two-edge-connected augmentation $S$ for $(H, R)$ such that weight $(S) \leq (1 + \epsilon)OPT(H, R) + 3 \sum_{\text{brick} B} \text{weight} (\text{East}_B \cup \text{West}_B)$.

#### 8.1 Sphere-Cut Decomposition

Our DP is based on a special kind of branch-decomposition of plane graphs, called a sphere-cut decomposition (see [19]): A noose of a plane graph is a Jordan curve that intersects only vertices of the graph and not edges. A sphere-cut decomposition of width $w$ is a family of non-crossing nooses each intersecting at most $w$ vertices; the nooses form a binary tree by the enclosure relation, each leaf noose encloses exactly one edge, and each edge is enclosed by a leaf noose. For each noose in the sphere-cut decomposition, we refer to the set of edges enclosed as a cluster.

---

9 If $u$ (resp. $v$) is on South, then $u_1$ (resp. $v_1$) equals $u$ (resp. $v$).
Lemma 24 (trivial adaptation from [31]). Let $G$ be a plane graph whose dual graph has diameter $k$. Then $G$ has a sphere-cut decomposition of width $2k$, and it can be computed in linear time.

8.2 Specification of DP Table

In this section, we define the index of the DP table and the value at an index.

By Lemma 24, $M$ has a sphere-cut decomposition $SC$ of width $O(1/\epsilon^3)$. The first index of the DP table is a cluster $E$ of $SC$.

Let $S_0$ be the optimal two-edge-connected augmentation for $(H,R)$, and let $S$ be the solution obtained in the Structure Theorem (Theorem 13). By the Bridge-Deletion Lemma (Lemma 4), we can modify $S_0$ so that every connected component in $R \cup S$ is two-edge-connected, without increasing the weight of $S$. For every cluster $E$ of $SC$, let $J_E$ be the noose enclosing $E$ and of minimum number of crossings with $R \cup S$ (all occurring at vertices), breaking ties by choosing the minimally enclosing one.$^{10}$ It is easy to show that the family of nooses $\{J_E\}_{E \in SC}$ is non-crossing.

Lemma 25. For every cluster $E$ of $SC$, $J_E$ intersects $O(1/\epsilon^7)$ vertices of $R \cup S$.

Proof. Since $SC$ has width $O(1/\epsilon^3)$, there is a noose enclosing $E$ that has $O(1/\epsilon^3)$ intersections with $M$. From one intersection to the next, it goes across a single brick, and by the Structure Theorem (Theorem 13), the part inside this brick can be chosen so as to have $O(1/\epsilon^3)$ intersections with $S$. This results in a noose enclosing $E$ that has $O(1/\epsilon^7)$ intersections with $R \cup S$.

Let $Q^* \subseteq V[H]$ denote the (unknown) set of $O(1/\epsilon^7)$ intersection vertices of $J_E$ with $S \cup R$. The second index of the DP table is a subset $Q \subseteq V[H]$ of size $O(1/\epsilon^7)$.

Next, we encode the connectivity structure of the part of $R \cup S$ inside $J_E$. Let $R_E$ (resp. $\Gamma^*$) denote the set of edges of $R$ (resp. $S$) that are inside $J_E$. Define a forest $F_0^*$ from $R_E \cup \Gamma^*$ by contracting every two-edge-connected component into a node. A node of $F_0^*$ is called internal if its corresponding two-edge-connected component in $R_E \cup \Gamma^*$ does not contain any node from $Q^*$, i.e., the component is strictly inside $J_E$. We then define a forest $F^*$ from $F_0^*$ by splicing internal nodes of degree 2 and removing internal nodes that are singletons. By the construction, $F^*$ has at most $|Q^*|$ non-internal nodes, and it does not contain internal nodes of degree 0, 1, or 2. So $F^*$ has at most $2|Q^*| - 2$ nodes. The third index of the DP table is a forest $F$ of at most $2|Q| - 2$ nodes. Moreover, there is a map $\psi^*$ giving the natural many-to-one map from $Q^*$ to nodes of $F^*$. The fourth index of the DP table is a map $\psi$ from $Q$ to $V[F]$. To summarize:

Definition 26 (DP index). An index of the DP table, also called a DP index, contains the following:

- $E$: a cluster of the sphere-cut decomposition $SC$
- $Q$: a subset of $V[H]$ of size $O(1/\epsilon^7)$
- $F$: a forest of size at most $2|Q| - 2$
- $\psi$: a map from $Q$ to $V[F]$, such that every node of degree 0, 1, or 2 in the forest $F$ belongs to the image of $\psi$.

$^{10}$Since the noose is a geometric object, it is not uniquely defined, but a discrete formulation can be given using the face-vertex incidence graph.
In addition, the triple \((Q, F, \psi)\) as defined above is called a partial DP index.\(^\text{11}\)

Before defining the value at a DP index, we need the concept of consistency to relate a solution with a DP index.

\begin{definition}[consistency] Let \((E, Q, F, \psi)\) be an index of the DP table. We say that a subset \(\Gamma\) of \(E[H]\) is consistent with \((E, Q, F, \psi)\) if, for every node \(a \in V[F]\), there exists a subgraph \(H_a\) of \(\Gamma \cup R_E\) such that the endpoints of every edge in \(R_E \cap H_a\) is two-edge-connected in \(H_a\); and for every edge \(ab \in E[F]\), there exists a path \(I_{ab}\) of \(\Gamma \cup R_E\) connecting \(H_a\) and \(H_b\), such that the following holds:
\begin{enumerate}
  \item Every vertex \(u \in Q\) belongs to \(H_{\psi(u)}\);
  \item For every edge \(uv \in R_E\) such that \(u\) and \(v\) are not two-edge-connected in \(\Gamma \cup R_E\), there exists exactly one edge \(ab\) from \(F\) such that \(uv \in I_{ab}\).
\end{enumerate}
We note \(H = \{H_a\}\) and \(I = \{I_{ab}\}\).
\end{definition}

By definition, \(\Gamma^*\) is consistent with \((E, Q^*, F^*, \psi^*)\). When \(E\) is the root cluster \(M\) of \(SC\), any \(\Gamma\) that is consistent with \((M, \emptyset, \emptyset, \emptyset)\) is a two-edge-connected augmentation for \((H, R)\).

For every DP index \((E, Q, F, \psi)\), define its value \(DP(E, Q, F, \psi)\) as the minimum weight among a collection of \(\Gamma\)'s, such that:
\begin{itemize}
  \item **Correctness:** Every \(\Gamma\) in this collection is consistent with \((E, Q, F, \psi)\);
  \item **Optimality:** If \((Q, F, \psi) = (Q^*, F^*, \psi^*)\), then \(\Gamma^*\) is in this collection.
\end{itemize}

In order to prove the Dynamic-Programming Theorem (Theorem 23), we only need to find a polynomial-time algorithm to fill in the DP table and to output the value \(DP(M, \emptyset, \emptyset, \emptyset)\).\(^\text{12}\)

8.3 Hole Region between Parent and Children

Let \(E\) be a cluster of \(SC\) and let \(E_1\) and \(E_2\) be its child clusters. Let \(Q^*, Q_1^*, Q_2^* \subseteq V[H]\) be the sets of intersections of \(R \cup S\) with \(J_E, J_{E_1}, J_{E_2}\). The hole region is the area inside \(J_E\) but outside \(J_{E_1}\) and \(J_{E_2}\) in the plane.\(^\text{13}\) See Figure 9. We remark that the hole region cannot contain edges from \(R\).

Let \(\hat{\Gamma}\) denote the set of edges of \(S\) in the hole region. Let \(\hat{Q}\) denote the set of intersections of \(S\) with the boundary of the hole region. We have \(\hat{Q} \subseteq Q^* \cup Q_1^* \cup Q_2^*\), thus \(|\hat{Q}| = O(1/\epsilon^4)\). From \(\hat{\Gamma}\) and \(\hat{Q}\), we encode the connectivity structure of the part of \(S\) in the hole region as a forest \(\hat{F}\) of at most \(2|\hat{Q}| - 2\) nodes and a map \(\hat{\psi} : \hat{Q} \rightarrow V[\hat{F}]\). This is similar to the encoding in Section 8.2.

We use a side table \(T\) for the computation at hole regions. The table is indexed by a partial DP index \((\hat{Q}, \hat{F}, \hat{\psi})\). The value \(T(\hat{Q}, \hat{F}, \hat{\psi})\) is defined as the minimum weight of any \(\hat{\Gamma}\) that is consistent with \((\hat{Q}, \hat{F}, \hat{\psi})\) and contains no cycles.

\begin{definition}[compatibility] Let \((Q, F, \psi), (Q_1, F_1, \psi_1), (Q_2, F_2, \psi_2), (\hat{Q}, \hat{F}, \hat{\psi})\) be partial DP indexes. Define a graph \(F_0\) as follows: starting from \(F_1 \cup F_2 \cup \hat{F}\), for every vertex \(u \in Q_1 \cap Q_2\), merge the nodes \(\psi_1(u)\) and \(\psi_2(u)\) (ideem for every \(u \in Q_1 \cap Q\) and every \(u \in Q_2 \cap \hat{Q}\)). Define a forest \(F_0\) by contracting two-edge-connected components in \(F_0\). We say that \((Q_1, F_1, \psi_1), (Q_2, F_2, \psi_2), (\hat{Q}, \hat{F}, \hat{\psi})\) are compatible with \((Q, F, \psi)\) if:
\begin{itemize}
  \item \(Q \subseteq Q_1 \cup Q_2 \cup \hat{Q}\).
\end{itemize}
\end{definition}

\(^{11}\)Note that the description of \(Q, F, \psi\) is independent of \(E\).

\(^{12}\)The DP outputs the value of a solution, not the solution itself; but it is easy to enrich the DP in the standard manner so that it also outputs the solution achieving the value.

\(^{13}\)Note that \(J_E, J_{E_1}\), and \(J_{E_2}\) are non-crossing.
Figure 9: $J_E$ is the outermost boundary. It encloses 4 areas that are separated by the solid curves. $J_{E_1}$ (resp. $J_{E_2}$) is the boundary of the left (resp. right) area. The hole region contains the top and bottom areas. The dashed paths represent $R \cup S$ inside $J_E$. The points represent vertices from $Q^* \cup Q_1 \cup Q_2$.

- $\psi: Q \to F_0$ is the natural extension of $\psi_1, \psi_2, \hat{\psi}$;
- $F$ is obtained from $F_0$ by deleting singletons $u \notin \psi(Q)$ and splicing nodes $u \notin \psi(Q)$ of degree 2.

\textbf{Fact 29.} $(Q_1^*, F_1^*, \psi_1^*), (Q_2^*, F_2^*, \psi_2^*)$ are compatible with $(Q^*, F^*, \psi^*)$.

\textbf{Lemma 30.} Let $(Q_i, F_i, \psi_i)$, $(Q_2, F_2, \psi_2)$, $(\hat{Q}, \hat{F}, \hat{\psi})$ be compatible with $(Q, F, \psi)$. Let $\Gamma_i$ be consistent with $(E_i, Q_i, F_i, \psi_i)$, for $i = 1, 2$, and let $\hat{\Gamma}$ be consistent with $(\hat{Q}, \hat{F}, \hat{\psi})$. Then $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \hat{\Gamma}$ is consistent with $(E, Q, F, \psi)$.

\textbf{Proof.} Let $(H_1, I_1), (H_2, I_2), (\hat{H}, \hat{I})$ be defined in the definition of consistency with respect to $\Gamma_1, \Gamma_2, \hat{\Gamma}$. The proof follows the lines of the construction of $F_0, F_0'$, and $F$ in the definition of compatibility. We initialize $\mathcal{H} := H_1 \cup H_2 \cup \hat{H}$ and $I := I_1 \cup I_2 \cup \hat{I}$, and do the following steps.

1. For every vertex $u \in Q_1 \cap Q_2$, let $H'$ be the concatenation of $H_{\psi_1(u)}$ and $H_{\psi_2(u)}$ at vertex $u$. For every edge $uv$ in $R_E \cap H'$, it is trivial to see that $u$ and $v$ are two-edge-connected in $H'$. Update $\mathcal{H}$ as $\mathcal{H} \cup \{H'\} \setminus \{H_{\psi_1(u)}, H_{\psi_2(u)}\}$. Idem for every $u \in Q_1 \cap \hat{Q}$ and every $u \in Q_2 \cap \hat{Q}$.

2. For every two-edge-connected component $C$ in $F_0'$, let $H'$ be the union of the subgraphs of $\{H_a\}_{a \in C}$ and of $\{I_{ab}\}_{ab \in C}$. To show that every edge $uv$ in $R_E \cap H'$ is such that $u$ and $v$ are two-edge-connected in $H'$, we only need to consider the non-trivial case when $uv$ belongs to a path $I_{ab}$ for some $ab \in C$. Define $H''$ as the union of the subgraphs of $\{H_a\}_{a \in C}$ and of $\{I_{ab}\}_{ab \in C}$, with edge repetition. Obviously, $u$ and $v$ are two-edge-connected in $H''$. Since $uv$ appears in none of $\{H_a\}_{a \in C}$ and in exactly one of $\{I_{ab}\}_{ab \in C}$, its multiplicity in $H''$ is 1. Thus there is a $u$-to-$v$ path in $H'' \setminus \{uv\}$, which can be transformed into a $u$-to-$v$ path in $H' \setminus \{uv\}$. So $u$ and $v$ are two-edge-connected in $H'$. Update $\mathcal{H}$ as $\mathcal{H} \cup \{H'\} \setminus \{H_a\}_{a \in C}$ and update $I$ as $I \setminus \{I_{ab}\}_{ab \in C}$.

3. For every internal node $a$ of $F_0$ of degree 0, delete $H_a$ from $\mathcal{H}$. For every internal node $a$ of $F_0$ of degree 2, let $b$ and $c$ be its neighbors. Let $u$ be the end vertex of the path $I_{ba}$ and let $v$ be the start vertex of the path $I_{ac}$. Since $u$ and $v$ are both in $H_a$, there is a path $I'$ in $H_a$ connecting $u$ and $v$. We set $I_{bc} = I_{ba} \circ I' \circ I_{ac}$. Delete $H_a$ from $\mathcal{H}$, and update $I$ as $I \cup \{I_{bc}\} \setminus \{I_{ba}, I_{ac}\}$.
Next, we show that $\Gamma$ is consistent with $(E, Q, F, \psi)$ using the above $H$ and $T$. It is easy to see that Properties 1 of consistency holds. Now we prove Property 2 of consistency. Let $uv$ be any edge in $R_E$, and assume without loss of generality that $uv \in R_{E_1}$. The interesting case is when $u$ and $v$ are not two-edge-connected in $\Gamma_1 \cup R_{E_1}$. Thus $uv \in I_{ab}$ for some edge $ab$ from $F_1$. There are three possibilities of the edge $ab$ in $F$: (1) contracted in some two-edge-connected component $C$; (2) merged during the splicing operation; (3) remains an edge in $F$. Property 2 holds in all the three cases.

\subsection{Implementation of DP Table}

First, the algorithm fills in the side table $T$ during the preprocessing. Notice that any $\hat{\Gamma} \subseteq E[H]$ that is consistent with $(\hat{Q}, \hat{F}, \hat{\psi})$ and contains no cycles is such that, every node $a$ in $\hat{F}$ corresponds to a vertex $u_a$ in the graph $H$, and every edge $ab$ in $\hat{F}$ corresponds to a path between $u_a$ and $u_b$ in $\hat{\Gamma}$. Therefore, to compute the value $T(\hat{Q}, \hat{F}, \hat{\psi})$, the algorithm enumerates, for every $a \in \hat{F}$, the vertex $u_a$ among $V[H]$. For every $ab \in \hat{F}$, it then computes the shortest path between $u_a$ and $u_b$ in $H$. The union of all these shortest paths defines the current $\hat{\Gamma}$. The value $T(\hat{Q}, \hat{F}, \hat{\psi})$ is the minimum weight of all $\hat{\Gamma}$’s during the enumeration. The overall running time of the preprocessing is thus polynomial.

Next, the algorithm fills in the DP table in the order of the index $E$ from bottom up in $SC$. Consider a DP index $(E, Q, F, \psi)$. Let $E_1$ and $E_2$ be the child clusters of $E$. The algorithm enumerates every combination of $(E_1, Q_1, F_1, \psi_1)$, $(E_2, Q_2, F_2, \psi_2)$, and $(\hat{Q}, \hat{F}, \hat{\psi})$ that are \textit{compatible} with $(E, Q, F, \psi)$, and the current weight is the sum of the three entries. $DP(E, Q, F, \psi)$ is assigned with the minimum weight during the enumeration.

At the beginning, the value of every entry is initialized as $\infty$.

\textbf{Base cases.}

$E = \{uv\}$. There are two possibilities for $(Q^*, F^*, \psi^*)$:

\[
\begin{cases}
(Q^A, F^A, \psi^A), & \text{if edge } uv \text{ belongs to exactly one of } S \text{ and } R, \\
(Q^B, F^B, \psi^B), & \text{otherwise}.
\end{cases}
\]

Here $Q^A = \{u, v\}$; $F^A$ is a forest containing two nodes $a$ and $b$ and an edge $ab$; $\psi^A$ maps $u$ to $a$ and $v$ to $b$; $Q^B = \emptyset$; $F^B = \emptyset$; and $\psi^B = \emptyset$. \footnote{When edge $uv$ belongs to neither $S$ nor $R$, the DP state $(Q^*, F^*, \psi^*) = (Q^B, F^B, \psi^B)$. However, when edge $uv$ belongs to both $S$ and $R$, strictly speaking, we have: $Q^* = \{u, v\}$, $F^*$ contains a singleton, and $\psi^*$ maps both $u$ and $v$ to that singleton. But since $S$ is minimal, $S \setminus \{uv\}$ is a two-edge-connected augmentation for $(H, R \setminus \{uv\})$. So we identify this case (and its DP state) with the case that $uv$ belongs to neither $S$ nor $R$ (and its DP state).}

If $uv \in R$, we set $DP(E, Q^A, F^A, \psi^A) = 0$ and set $DP(E, Q^B, F^B, \psi^B) = weight(uv)$.

If $uv \notin R$, we set $DP(E, Q^B, F^B, \psi^B) = 0$ and set $DP(E, Q^A, F^A, \psi^A) = weight(uv)$.

By inspection, both properties of the DP value are satisfied.

\textbf{Recurrence.}

$E = E_1 \cup E_2$, where $E_1, E_2 \in SC$.

\[
DP(E, Q, F, \psi) = \min \left\{ DP(E_1, Q_1, F_1, \psi_1) + DP(E_2, Q_2, F_2, \psi_2) + T(\hat{Q}, \hat{F}, \hat{\psi}) \right\}
\]
where the minimum is taken over all partial indexes \((Q_1, F_1, \psi_1), (Q_2, F_2, \psi_2), (\hat{Q}, \hat{F}, \hat{\psi})\) that are compatible with \((Q, F, \psi)\).

By recurrence, both properties of the DP value are satisfied: the Correctness Property follows from Lemma 30 and the Optimality Property follows from Fact 29.

## 9 Proof of Augmentation Theorem

The algorithm for the Augmentation Theorem is as follows.

**Algorithm 3 Augment-Connected**

**Input:** a planar graph \(G\), and a subset of edges \(R\), and a connected subgraph \(T\)

**Output:** two-edge-connected augmentation for \((G, R)\)

1. \(M \leftarrow\) mortar graph of \(G\) based on \(R\) and \(T\) \(\triangleright\) Lemma 11
2. \(M^* \leftarrow\) dual graph of \(M\)
3. BFS on \(M^*\) from an arbitrary brick, thus every brick has a level between 0 and \(\ell\)
4. Let \(k = \Theta(1/\epsilon^3)\)
5. Let \(j \in [0, k - 1]\) minimizes \(\text{weight (boundary of bricks of level } \equiv j \mod k)\)
6. \(S_0 \leftarrow\) boundary of bricks of level \(\equiv j \mod k\)
7. for every \(i \in [0, \ell/k]\) do
   8. \(G_i \leftarrow\) subgraph of \(G\) of bricks with level between \((i - 1)k + j\) and \(ik + j\)
   9. \(H_i \leftarrow G_i\) by doubling East, South, and West boundaries of every brick \(\triangleright\) Section 6.2
10. \(M_i \leftarrow M\) restricted to \(G_i\)
11. \(R_i \leftarrow R \cap G_i\)
12. \(\text{DP}_i \leftarrow \text{Dynamic-Program}(H_i, M_i, R_i)\) \(\triangleright\) Section 8
13. return \((\bigcup_i \text{DP}_i) \cup S_0\)

For every subinstance \(G_i\), the mortar graph \(M_i\) of \(G_i\) consists of at most \(k\) levels of bricks. The mortar graph structure of \(H_i\) is inherited from that of \(G_i\). By the Boundary Doubling Lemma (Lemma 12), the two-edge-connected augmentation problem for \((G_i, R_i)\) and for \((H_i, R_i)\) are equivalent, where \(R_i = R \cap G_i\). Using the Dynamic Programming Theorem (Theorem 23), a two-edge-connected augmentation \(\text{DP}_i\) for \((H_i, R_i)\) can be computed in polynomial time such that

\[
\text{weight (DP}_i) \leq (1 + \epsilon)OPT(H_i, R_i) + 3 \sum_{\text{brick } B \in H_i} \text{weight (East}_B \cup \text{West}_B).
\]

So weight \((\bigcup_i \text{DP}_i)\) is at most \((1 + \epsilon)OPT(G, R) + O(\epsilon^2 \cdot \text{weight (T)})\) using Property 3 of Lemma 11. The weight of \(S_0\) is at most \(1/k\) times the weight of the mortar graph \(M\), which is \(O(\epsilon^2 \cdot \text{weight (T)})\) by Lemma 11. Thus weight \((\bigcup_i \text{DP}_i) \cup S_0\) \(\leq (1 + \epsilon)OPT(G, R) + O(\epsilon^2 \cdot \text{weight (T)})\).

Theorem 7 follows by replacing \(\epsilon\) by \(\epsilon' = \epsilon/K\) for some appropriate \(K\), which is the hidden constant in \(O(\epsilon^2 \cdot \text{weight (T)})\).

**Acknowledgements** We would like to thank Howard J. Karloff for helping make the connection between correlation clustering and two-edge-connected augmentation in planar graphs; Nabil Mustafa for numerous discussions; and Grigory Yaroslavtsev.
References


