Variants of Plane Diameter Completion

Clément Requilé, Dimitrios M. Thilikos, Petr A. Golovach

Variants of Plane Diameter Completion*

Petr A. Golovach[1], Clément Requilé[2], and Dimitrios M. Thilikos[3,4,5]

1 **Department of Informatics, University of Bergen, Bergen, Norway**
`Petr.Golovach@ii.uib.no`
2 **Freie Universität Berlin, Institut für Mathematik und Informatik, Berlin, Germany**
`requile@math.fu-berlin.de`
3 **AlGCo project-team, CNRS, LIRMM, Montpellier, France**
`sedthilk@thilikos.info`
4 **Department of Mathematics, University of Athens, Athens, Greece**
5 **Computer Technology Institute & Press "Diophantus", Patras, Greece**

Abstract

The Plane Diameter Completion problem asks, given a plane graph G and a positive integer d, if it is a spanning subgraph of a plane graph H that has diameter at most d. We examine two variants of this problem where the input comes with another parameter k. In the first variant, called BPDC, k upper bounds the total number of edges to be added and in the second, called BFPDC, k upper bounds the number of additional edges per face. We prove that both problems are NP-complete, the first even for 3-connected graphs of face-degree at most 4 and the second even when $k = 1$ on 3-connected graphs of face-degree at most 5. In this paper we give parameterized algorithms for both problems that run in $O(n^3) + 2^{2^{O((kd)^2 \log d)}} \cdot n$ steps.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Planar graphs, graph modification problems, parameterized algorithms, dynamic programming, branchwidth

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.30

1 Introduction

In 1987, Chung [1, Problem 5] introduced the following problem[1]: find the optimum way to add q edges to a given graph G so that the resulting graph has minimum diameter. This problem was proved to be NP-hard if the aim is to obtain a graph of diameter at most 3 [14], and later the NP-hardness was shown even for the Diameter-2 Completion problem [9]. It is also know that Diameter-2 Completion is W[2]-hard when parameterized by q [6].

For planar graphs, Dejter and Fellows introduced in [3] the Planar Diameter Completion problem that asks whether it is possible to obtain a planar graph of diameter at most

* The first author was supported by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959. The second author was supported by the FP7-PEOPLE-2013-CIG project CountGraph (ref. 630749), the collateral PROCOPE-DAAD project RanConGraph (ref. 57134837), and the Berlin Mathematical School. The research of the third author was co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF), Research Funding Program: ARISTEIA II.

[1] Notice that in all problems defined in this paper we can directly assume that G is a simple graph as loops do not contribute to the diameter of a graph and the same holds if we take simple edges instead of multiple ones.

10th International Symposium on Parameterized and Exact Computation (IPEC 2015).
Editors: Thore Husfeldt and Iyad Kanj; pp. 30–42
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

d from a given planar graph by edge additions. It is not known whether PLANAR DIAMETER COMPLETION admits a polynomial time algorithm, but Dejter and Fellows showed that, when parameterized by d, PLANAR DIAMETER COMPLETION is fixed parameter tractable [3]. The proof is based on the fact that the YES-instances of the problem are closed under taking minors. Because of the Robertson and Seymour theorem [13] and the algorithm in [11], this implies that, for each d, the set of graphs G for which (G, d) is a YES-instance can be characterized by a *finite* set of forbidden minors. This fact, along with the minor-checking algorithm in [12] implies that there exists an $O(f(d) \cdot n^3)$-step algorithm (i.e. an FPT-algorithm) deciding whether a plane graph G has a plane completion of diameter at most d. Using the parameterized complexity, this means that PLANAR DIAMETER COMPLETION is FPT, when parameterized by d. To make this result constructive, one requires the set of forbidden minors for each d, which is unknown. To find a constructive FPT-algorithm for this parameterized problem remains a major open problem in parameterized algorithm design.

Our results. We denote by $\mathbb{S}_0$ the 3-dimensional sphere. By a *plane* graph G we mean a simple planar graph G with the vertex set $V(G)$ and the edge set $E(G)$ drawn in $\mathbb{S}_0$ such that no two edges of this embedding intersect. A plane graph H is a *a plane completion* (or, simply *completion*) of another plane graph G if H is a spanning subgraph of G. A *q-edge completion* of a plane graph G is a completion H of G where $|E(H)| - |E(G)| \leq q$. A *k-face completion* of a plane graph G is a completion H of G where at most k edges are added in each face of G. We consider the following problem:

PLANE DIAMETER COMPLETION (PDC)
Input: a plane graph G and $d \in \mathbb{N}_{\geq 1}$.
Output: is there a completion of G with diameter at most d?

An important difference between PDC and the aforementioned problems is that we consider plane graphs, i.e., the aim is to reduce the diameter of a given embedding of a planar graph preserving the embedding. In particular, we are interested in the following variants:

BOUNDED BUDGET PDC (BPDC)
Input: a plane graph G and $q \in \mathbb{N}, d \in \mathbb{N}_{\geq 1}$
Question: is there a completion H of G of diameter at most d that is also a q-edge completion?

BOUNDED BUDGET/FACE PDC (BFPDC)
Input: a plane graph G and $k \in \mathbb{N}, d \in \mathbb{N}_{\geq 1}$.
Question: is there a completion H of G of diameter at most d that is also a k-face completion?

We examine the complexity of the two above problems. Our hardness results are the following.

▶ **Theorem 1.** *Both* BPDC *and* BFPDC *are* NP*-complete. Moreover,* BPDC *is* NP*-complete even for 3-connected graphs of face-degree at most 4, and* BFPDC *is* NP*-complete even for* $k = 1$ *on 3-connected graphs of face-degree at most 5.*

The hardness results are proved using a series of reductions departing from the PLANAR 3-SATISFIABILITY problem that was shown to be NP-hard by Lichtenstein in [10].

The results of Theorem 1 prompt us to examine the parameterized complexity of the above problems (for more on parameterized complexity, we refer the reader to [5]). For this, we consider the following general problem:

Bounded Budget and Budget/Face BDC (BBFPDC) *Input*: a plane graph G, $q \in \mathbb{N} \cup \{\infty\}$, $k \in \mathbb{N}$, and $d \in \mathbb{N}_{\geq 1}$. *Question*: is there a completion H of G of diameter at most d that is also a q-edge completion and a k-face completion?

Notice that when $q = \infty$ BBFPDC yields BFPDC and when $q = k$ BBFPDC yields BPDC. Our main result is that BBFPDC is fixed parameter tractable (belongs in the parameterized class FPT) when parameterized by k and d.

▶ **Theorem 2.** *It is possible to construct an $O(n^3) + 2^{2^{O((kd)\log d)}} \cdot (\alpha(q))^2 \cdot n$-step algorithm for* BBFPDC.

In the above statement and in the rest of this paper we use the function $\alpha : \mathbb{N} \cup \{\infty\} \to \mathbb{N}$ such that if $q = \infty$, then $\alpha(q) = 1$, otherwise $\alpha(q) = q$.

The main ideas of the algorithm of Theorem 2 are the following. We first observe that YES-instances of PDC and all its variants have bounded branchwidth (for the definition of branchwidth, see Section 2). The typical approach in this case is to derive an FPT-algorithm by either expressing the problem in Monadic Second Order Logic – MSOL (using Courcelle's theorem [2]) or to design a dynamic programming algorithm for this problem. However, for completion problems, this is not really plausible as this logic can quantify on *existing* edges or vertices of the graph and not on the "non-existing" completion edges. This also indicates that to design a dynamic programming algorithm for such problems is, in general, not an easy task. In this paper we show how to tackle this problem for BBFPDC (and its special cases BPDC and BFPDC). Our approach is to deal with the input G as a part of a more complicated graph with $O(k^2 \cdot n)$ additional edges, namely its *cylindrical enhancement G'* (see Section 3 for the definition). Informally, sufficiently large cylindrical grids are placed inside the faces of G and then internally vertex disjoint paths in these grids can be used to emulate the edges of a solution of the original problem placed inside the corresponding faces. Thus, by the enhancement we reduce BBFPDC to a new problem on G' certified by a suitable 3-partition of the additional edges. Roughly, this partition consists of the 1-weighted edges that should be added in the completion, the 0-weighted edges that should link these edges to the boundary of the face of G where they will be inserted, and the ∞-weighted edges that will be the (useless) rest of the additional edges. The new problem asks for such a partition that simulates a bounded diameter completion. The good news is that, as long as the number of edges per face to be added is bounded, which is the case for BBFPDC, the new graph G' has still bounded branchwidth and it is possible, in the new instance, to quantify this 3-partition of the graph G'. However, even under these circumstances, to express the new problem in Monadic Second Order Logic is not easy. For these reasons we decided to follow the more technical approach of designing a dynamic programming algorithm that leads to the (better) complexity bounds of Theorem 2. This algorithm is quite involved due to the technicalities of the translation of the BBFPDC to the new problem. It runs on a sphere-cut decomposition of the plane embedding of G' and its tables encode how a partial solution is behaving inside a closed disk whose boundary meets only (a few of) the edges of G'. We stress that this encoding takes into account the topological embedding and not just the combinatorial structure of G'. Sphere-cut decompositions as well as some necessary combinatorial structures for this encoding are presented in Section 4. The dynamic programming algorithms is presented in Section 5 and is the most technical part of this paper.

Due to space restrictions, various proofs are omitted in this extended abstract and are available in [7].

2 Definitions and preliminaries

Given a graph G, we denote by $V(G)$ (respectively $E(G)$) the *set of vertices* (respectively *edges*) of G. A graph G' is a *subgraph* of a graph G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$, and we denote this by $G' \subseteq G$. Also, in case $V(G) = V(G')$, we say that H is a *spanning subgraph* of G. If S is a set of vertices or a set of edges of a graph G, the graph $G \setminus S$ is the graph obtained from G after the removal of the elements of S. If S is a set of edges, we define $G[E]$ as the graph whose vertex set consists of the endpoints of the edges of E and whose edge set of E.

Distance and diameter. Let G be a graph and let $\mathsf{w} : E(G) \to \mathbb{N} \cup \{\infty\}$ (w is a *weighting of the edges of* G). Given two vertices $x, x' \in V(G)$ we call (x, x')*-path* every path of G with x and x' as endpoints. We also define $\mathsf{w}\text{-}\mathbf{dist}_G(x, x') = \min\{\mathsf{w}(E(P)) \mid P \text{ is an } (x, x')\text{-path in } G\}$.

Plane graphs. To simplify notations on plane graphs, we consider a plane graph G as the union of the points of $\mathbb{S}_0$ in its embedding corresponding to its vertices and edges. That way, a subgraph H of G can be seen as a graph H where $H \subseteq G$. The *faces* of a plane graph G, are the connected components of the set $\mathbb{S}_0 \setminus G$. A vertex v (an edge e resp.) of a plane graph G is *incident* to a face f and, vice-versa, f is incident to v (resp. e) if v (resp., e) lies on the boundary of f. The *degree* of a face f of G is the number of edges incident to f where bridges of G count double in this number. The *face-degree* of G is the maximum degree of a face in $F(G)$. A set $\Delta \subseteq \mathbb{S}_0$ is an open disc if it is homeomorphic to $\{(x, y) : x^2 + y^2 < 1\}$. Also, Δ is a *closed disk* of $\mathbb{S}_0$ if it is the closure of some open disk of $\mathbb{S}_0$.

Branch decomposition. Given a graph H with n vertices, a branch decomposition of H is a pair (T, μ), where T is a tree with all internal vertices of degree three and $\mu : L \to E(H)$ is a bijection from the set of leaves of T to the edges of H. For every edge e of T, we define the middle set $\mathbf{mid}(e) \subseteq V(H)$ as follows: if $T \setminus \{e\}$ has two connected components T_1 and T_2, and for $i \in \{1, 2\}$, let $H_i^e = H[\{\mu(f) : f \in L \cap V(T_i)\}]$, and set $\mathbf{mid}(e) = V(H_1^e) \cap V(H_2^e)$. The width of (T, μ) is the maximum order of the middle sets over all edges of T, i.e. $\max\{|\mathbf{mid}(e)| : e \in T\}$. The *branchwidth* of H is the minimum width of a branch decomposition of H and is denoted by $\mathbf{bw}(H)$.

We use the following lemma.

▶ **Lemma 3.** *There exists a constant c_1 such that if (G, d) is a* YES*-instance of* PDC*, then $\mathbf{bw}(G) \leq c_1 \cdot d$. The same holds for the graphs in the* YES*-instances of* BPDC*,* BFPDC*, and* BBFPDC*.*

3 The reduction

Edge colorings of new edges. Let G and H be two plane graphs such that G is a subgraph of H and let $q \in \mathbb{N} \cup \{\infty\}$, $k \in \mathbb{N}$, and $d \in \mathbb{N}_{\geq 1}$. Given a 3-partition $\mathbf{p} = \{E^0, E^1, E^\infty\}$ of $E(H) \setminus E(G)$, we define the function $\mathsf{w}_{\mathbf{p}} : E(H) \to \mathbb{N}$ such that

$$\mathsf{w}_{\mathbf{p}} \quad = \quad \{(e, 1) \mid e \in E(G)\} \cup \{(e, 0) \mid E \in E^0\} \cup \{(e, 1) \mid e \in E^1\} \cup \{(e, d+1) \mid E \in E^\infty\}.$$

We say that G has (q, k, d)*-extension* in H if there is a 3-partition $\mathbf{p} = \{E^0, E^1, E^\infty\}$ of $E(H) \setminus E(G)$ such that the following conditions hold

A. There is no path in H with endpoints in $V(G)$ that consists of edges in E^0,
B. every face F of G contains at most k edges of E^1,

C. $\forall x, y \in V(G), \mathsf{w}_{\mathbf{p}}\text{-}\mathbf{dist}_H(x, y) \leq d$, and
D. $|E^1| \leq q$.

Given a 3-partition $\mathbf{p} = \{E^0, E^1, E^\infty\}$ of $E(H) \setminus E(G)$ we refer to its elements as the *0-edges*, the *1-edges*, and the ∞*-edges* respectively. We also call the edges of G *old-edges*.

▶ **Lemma 4.** *There exists a $c_2 \in \mathbb{Z}_{\geq 1}$ and an algorithm that receives as input a planar graph G on n vertices and a positive integer k and outputs a 3-connected planar graph G_w where*
- $\mathbf{bw}(G_k) \leq c_2 \cdot k \cdot \mathbf{bw}(G))$.
- *For every $q \in \mathbb{N} \cup \{\infty\}$ and $d \in \mathbb{N}_{\geq 1}$, (G, q, k, d) is a* YES*-instance of* BBFPDC *if and only if G has a (q, k, d)*-extension *in G_k.*

Moreover, this algorithm runs in $O(k^2 \cdot n)$ steps.

4 Structures for dynamic programming

For our dynamic programming algorithm we need a variant of branchwidth for plane graphs whose middle sets have additional topological properties.

Sphere-cut decomposition. Let H be a plane graph. An arc is a subset O of the plane homeomorphic to a circle and is called a *noose of H* if it meets H only in vertices. We also set $V_O = V(H) \cap O$. An *arc* of a noose O is a connected component of $O \setminus V_O$ while in the trivial case where $V_O = \emptyset$, O does not have arcs. A *sphere-cut decomposition* or *sc-decomposition* of H is a triple (T, μ, π) where (T, μ) is a branch decomposition of H and π is a function mapping each $e \in E(T)$ to cyclic orderings of vertices of H, such that for every $e \in E(T)$ there is a noose O_e of H where the following properties are satisfied:
- O_e meets every face of H at most once,
- H_1^e is contained in one of the closed disks bounded by O_e and H_2^e is contained in the other (H_1^e and H_2^e are as in the definition of branch decomposition).
- $\pi(e)$ is a cyclic ordering of V_{O_e} defined by a clockwise traversal of O_e in the embedding of H.

We denote $X_e = V_{O_e}$ and we always assume that its vertices are clockwise enumerated according to $\pi(e)$. We denote by $\mathbf{A}_e$ the set containing the arcs of O_e. Also, if $\pi(e) = [a_1, \ldots, a_k, a_1]$, then we use the notation $\mathbf{A}_e = \{a_{1,2}, a_{2,3}, \ldots, a_{k-1,k}, a_{k,1}\}$ where the boundary of the arc $a_{i,i+1}$ consists of the vertices a_i and a_{i+1}. We also define $H_e^+ = (V(H), E(H \cup \mathbf{A}_e))$, i.e., H_e^+ is the embedding occurring if we add in H the arcs of O_e as edges. A face of H_e^+ is called *internal* if it is not incident to an arc in $\mathbf{A}_e$, i.e., it is also a face of H. A face of H_e^+ is *marginal* if it is a properly included is some face of H.

For our dynamic programming we require to have in hand an optimal sphere-cut decomposition. This is done combining the main result of [8] and [15, (5.1)] (see also [4]) and is summarized to the following.

▶ **Proposition 5.** *There exists an algorithm that, with input a 3-connected plane graph G and $w \in \mathbb{N}$, outputs a sphere-cut decomposition of G of width at most w or reports that* $\mathbf{bw}(G) > w$.

Our next step is to define a series of combinatorial structures that are necessary for our dynamic programming. Given two sets A and B we denote by A^B the set of all functions from B to A.

(d, k, q)-configurations. Given a set X and a non-negative integer t, we say that the pair $(\mathcal{X}, \chi)$ is a *t-labeled partition of* X if $\mathcal{X}$ is a collection of pairwise disjoint non-empty subsets of X and χ is a function mapping the integers in $\{1, \ldots, |\mathcal{X}|\}$ to integers in $\{0, \ldots, t\}$. In case $X = \emptyset$, a *t-labeled partition* corresponds to the pair $\{\emptyset, \varnothing\}$ where $\varnothing$ is the "empty" function, i.e. the function whose domain is empty. Let X and A be two finite sets. Given $d, k \in \mathbb{N}$ and $q \in \mathbb{N} \cup \{\infty\}$, we define a *$(d, k, q)$-configuration* of (X, A) as a quintuple $((\mathcal{X}, \chi), (\mathcal{A}, \alpha), (\mathcal{F}, \mathcal{E}), \delta, z)$ where

1. $(\mathcal{X}, \chi)$ is a 1-labeled partition of X,
2. $(\mathcal{A}, \alpha)$ is a k-labeled partition of A,
3. $(\mathcal{F}, \mathcal{E})$ is a graph (possibly with loops) where $\mathcal{F} \subseteq \{0, \ldots, d+1\}^X$,
4. $\delta \in \{0, \ldots, d+1\}^{X^2}$, and
5. if $q \in \mathbb{N}$, then $z \leq q$, otherwise $z = \infty$.

Fusions and restrictions. Let $(\mathcal{X}_1, \chi_1)$ and $(\mathcal{X}_2, \chi_2)$ be two t-labeled partitions of the sets X_1 and X_2 respectively such that $\mathcal{X}_i = \{X^i_1, \ldots, X^i_{\rho_1}\}, i \in \{1, 2\}$. We define $\mathcal{X}_1 \oplus \mathcal{X}_2$ as follows: if $x, x' \in X_1 \cup X_2$ we say that $x \sim x'$ if there is a set in $\mathcal{X}_1 \cup \mathcal{X}_2$ that contains both of them. Let $\sim_T$ be the transitive closure of $\sim$. Then $\mathcal{X}_1 \oplus \mathcal{X}_2$ contains the equivalence classes of $\sim_T$. We now define $\chi_1 \oplus \chi_2$ as follows: let $\mathcal{X}_1 \oplus \mathcal{X}_2 = \{Y_1, \ldots, Y_\rho\}$. Then for each $i \in \{1, \ldots, \rho\}$, we define $\chi_1 \oplus \chi_2(i) = \min\{t, \sum_{X^1_{i'} \subseteq Y_i} \chi_1(i') + \sum_{X^2_{i'} \subseteq Y_i} \chi_2(i')\}$.

The *fusion* of the t-labeled partitions $(\mathcal{X}_1, \chi_1)$ and $(\mathcal{X}_2, \chi_2)$ is the pair $(\mathcal{X}_1 \oplus \mathcal{X}_2, \chi_1 \oplus \chi_2)$ that is a $(t+1)$-labeled partition and is denoted by $(\mathcal{X}_1, \chi_1) \oplus (\mathcal{X}_2, \chi_2)$. Given a t-labeled partition $(\mathcal{X}, \chi)$ of a set X and given a subset X' of X we define the *restriction* of $(\mathcal{X}, \chi)$ to X' as the t-labeled partition $(\mathcal{X}', \chi')$ of X' where $\mathcal{X}' = \{X_i \cap X' \mid X_i \in \mathcal{X}\} \setminus \{\emptyset\}$ and $\chi' = \{(i, \chi(i)) \mid X_i \cap X' \neq \emptyset\}$ and we denote it by $(\mathcal{X}, \chi)|_{X'}$. We also define the intersection of $(\mathcal{X}, \chi)$ with X' as the t-labeled partition $(\mathcal{X}', \chi')$ where $\mathcal{X}' = \{X_i \in \mathcal{X} \mid X_i \cap (X \setminus X') \neq \emptyset\}$ and $\chi' = \{(i, \chi(i)) \mid X_i \cap X'' \neq \emptyset\}$ where $X'' = \cup_{X'_i \in \mathcal{X}'} X_i$ and we denote it by $(\mathcal{X}, \chi) \cap X'$. Notice that $(\mathcal{X}, \chi)|_{X'}$ and $(\mathcal{X}, \chi) \cap X'$ are not always the same.

5 Dynamic programming

The following result is the main algorithmic contribution of this paper.

▶ **Lemma 6.** *There exists an algorithm that, given (G, H, q, k, d, D, b) as input where G and H are plane graphs such that G is a subgraph of H, H is 3-connected, $q \in \mathbb{N} \cup \{\infty\}$, $k \in \mathbb{N}$, $d \in \mathbb{N}_{\geq 1}$, $b \in \mathbb{N}$, and $D = (T, \mu, \pi)$ is a sphere-cut decomposition of H with width at most b, decides whether G has* (q, k, d)-extension *in H in $(\alpha(q))^2 \cdot 2^{O(b^2 \log d) + 2^{O(b \log d)}} \cdot n$ steps.*

Proof. We use the notation $E^{\text{old}} = E(G)$ and $E^{\text{new}} = E(H) \setminus E(G)$, $V^{\text{old}} = V(G)$ and $V^{\text{new}} = V(H) \setminus V(G)$. We choose an arbitrary edge $e^* \in E(T)$, subdivide it by adding a new vertex v_{new} and update T by adding a new vertex r adjacent to v_{new}. We then root T at this vertex r and we extend μ by setting $\mu(r) = \emptyset$. In T we call *leaf-edges* all its edges that are incident to its leaves except from the edge $e_r = \{r, v_{\text{new}}\}$. An edge of T that is not a leaf-edge is called *internal*. We denote by $L(T)$ the set of the leaf-edges of T and we denote by $I(T)$ the internal edges of T. We also call e_r *root-edge*. For each $e \in E(T)$, let T_e be the tree of the forest $T \setminus \{e\}$ that does not contain r as a leaf and let E_e be the edges that are images, via μ, of the leaves of T that are also leaves of T_e. We denote $H_e = H[E_e]$ and $V_e = V(H_e)$ and observe that $H_{e_r} = H$. For each edge $e \in I(T)$, we define its children as the two edges that both belong in the connected component of $T \setminus e$ that does not contain the root r and that share a common endpoint with e. Also, for each edge $e \in E(T)$, we define Δ_e as the closed disk bounded by O_e such that $G \cap \Delta_e = H_e$. Finally, for each edge $e \in E(T)$, we set $X_e = \mathbf{mid}(e)$, $V_e^{\text{new}} = V_e \cap V^{\text{new}}$, $V_e^{\text{old}} = V_e \cap V^{\text{old}}$, $E_e^{\text{new}} = E_e \cap E^{\text{new}}$, and $E_e^{\text{old}} = E_e \cap E^{\text{old}}$.

Distance signatures and dependency graphs. Let $\mathbf{p} = \{E_e^0, E_e^1, E_e^\infty\}$ be a 3-partition of E_e^{new}. For each vertex $v \in V_e$, we define the $(X_e, \mathbf{p})$-*distance vector* of v as the function $\phi_v : X_e \to \{0, \dots, d+1\}$ such that if $x \in X_e$ then $\phi_v(x) = \min\{\mathsf{w}_{\mathbf{p}}\text{-}\mathbf{dist}_{G_e}(v, x), d+1\}$. We define the $(e, \mathbf{p})$-*dependency graph* $\mathcal{G}_{e,\mathbf{p}} = (\mathcal{F}_{e,\mathbf{p}}, \mathcal{E}_{e,\mathbf{p}})$ (that may contain loops) where $\mathcal{F}_{e,\mathbf{p}} = \{\phi_v \mid v \in V_e\}$ and such that two (not necessarily distinct) vertices ϕ and ϕ' of $\mathcal{F}_{e,\mathbf{p}}$ are connected by an edge in $\mathcal{E}_{e,\mathbf{p}}$ if and only if there exist $v, v' \in V_e$ such that $\phi = \phi_v$, $\phi' = \phi_{v'}$ and $\mathsf{w}_{\mathbf{p}}\text{-}\mathbf{dist}_{H_e}(v, v') > d$. Notice that the set $\Phi_e = \{\mathcal{G}_{e,\mathbf{p}} \mid \mathbf{p} \text{ is a 3-partition of } E_e^{\text{new}}\}$ has at most $2^{(d+2)^{|X_e|}}$ elements because $\{\mathcal{F}_{e,\mathbf{p}} \mid \mathbf{p} \text{ is a 3-partition of } E_e^{\text{new}}\} \subseteq \{0, \dots, d+1\}^{X_e}$ and, to each $\mathcal{F}_{e,\mathbf{p}}$, assign a unique edge set $\mathcal{E}_{e,\mathbf{p}}$. Intuitively, each $\mathcal{F}_{e,\mathbf{p}}$ corresponds to a partition of the elements of V_e such that vertices in the same part have the same $(X_e, \mathbf{p})$-distance signature. Moreover the existence of an edge in the $(e, \mathbf{p})$-dependency graph between two such parts implies that they contain vertices, one from each part, whose $\mathsf{w}_{\mathbf{p}}$-distance in H_e is bigger than d.

The tables. Our aim is to give a dynamic programming algorithm running on the sc-decomposition T. For this, we describe, for each $e \in E(T)$, a table $\mathfrak{T}(e)$ containing information on partial solutions of the problem for the graph G_e in a way that the table of an edge $e \in E(T)$ can be computed using the tables of the two children of e, the size of each table does not depend on G and the final answer can be derived by the table of the root-edge e_r.

We define the function $\mathfrak{T}$ mapping each $e \in E(T)$ to a collection $\mathfrak{T}(e)$ of (d, k, q)-configurations of $(X_e, \mathbf{A}_e)$. In particular, $Q = ((\mathcal{X}, \chi), (\mathcal{A}, \alpha), (\mathcal{F}, \mathcal{E}), \delta, z) \in \mathfrak{T}(e)$ iff there exists a 3-partition $\mathbf{p} = \{E_e^0, E_e^1, E_e^\infty\}$ of E_e^{new} such that the following hold:

1. $C_1, \dots, C_h$ are the connected components of $(V(H_e), E_e^0)$, then
   - $\mathcal{X} = \{V(C_1) \cap X_e, \dots, V(C_h) \cap X_e\}$ and
   - $\forall_{i \in \{1,\dots,h\}}\ \chi(i) = 1$ if C_i contains some vertex of V_e^{old}, otherwise $\chi(i) = 0$.

   (The pair $(\mathcal{X}, \chi)$ encodes the connected components of the 0-edges that contain vertices of X_e and for each of them registers the number (0 or 1) of the vertices in V_e^{old} in them. This information is important to control Condition A.)
2. $\mathcal{A}$ is a partition of $\mathbf{A}_e$ such that two arcs $A, A' \in \mathbf{A}_e$ belong in the same set, say A_i of $\mathcal{A}$ if and only if they are incident to the same marginal face f_i of H_e^+. Moreover, for each $i \in \{1, \dots, |\mathcal{A}|\}$, $\alpha(i)$ is equal to the number of edges in E_e^1 that are inside f_i.

   (Here $(\mathcal{A}, \alpha)$ encodes the "partial" faces of the embedding of G_e that are inside Δ_e. To each of them we correspond the number of 1-edges that they contain in H_e. This is useful in order to guarantee that during the algorithm, faces that stop being marginal do not contain more than k 1-edges, as required by Condition B.)
3. $(\mathcal{F}, \mathcal{E})$ is the $(e, \mathbf{p})$-dependency graph, i.e., the graph $\mathcal{G}_{e,\mathbf{p}} = (\mathcal{F}_{e,\mathbf{p}}, \mathcal{E}_{e,\mathbf{p}})$.

   (Recall that $\mathcal{F}$ is the collection of all the different distance vectors of the vertices of V_e. Notice also that there might be pairs of vertices $x, x' \in V_e$ whose $\mathbf{w}_{\mathbf{p}}$-distance in G_e is bigger than d. In order for G to have a completion of diameter d, these two vertices should become connected, at some step of the algorithm, by paths passing *outside* Δ_e. To check this possibility, it is enough to know the distance vectors of x and x' and these are encoded in the set $\mathcal{F}$. Moreover the fact that x and x' are still "far away" inside G_e is certified by the existence of an edge (or a loop) between their distance vectors in $\mathcal{F}$.)

4. For each pair $x, x' \in X_e$, $\delta(x, x') = \min\{\mathsf{w}_{\mathbf{p}}\text{-}\mathbf{dist}_{H_e}(x, x'), d+1\}$.
(This information is complementary to the one stored in $\mathcal{F}$ and registers the distances of the vertices in X_e inside H_e. As we will see, $\mathcal{F}$ and δ will be used in order to compute the distance vectors as well as their dependencies during the steps of the algorithm.)
5. There is no path in H_e with endpoints in V_e^{old} that consists of edges in E_e^0.
(This ensures that Condition A is satisfied for the current graph G_e.)
6. Every internal face of G_e^+ contains at most k edges in E_e^1.
(This ensures that Condition B holds for all the internal faces of G_e.)
7. $\forall v, v' \in V_e$, either $\mathsf{w}_{\mathbf{p}}\text{-}\mathbf{dist}_{H_e}(v, v') \leq d$ or there are two vertices $x, x' \in X_e$ such that $\phi_v(x) + \phi_{v'}(x') \leq d$.
(Here we demand that if two vertices x_1, x_2 of V_e are "far away" (have $\mathbf{w}_{\mathbf{p}}$-distance $> d$) inside H_e then they have some chance to come "close" (obtain $\mathbf{w}_{\mathbf{p}}$-distance $\leq d$) in the final graph, so that Condition C is satisfied. This fact is already stored by an edge in $\mathcal{E}$ between the two distance vectors of x and x' and the possibility that x_1 and x_2 may come close at some step of the algorithm, in what concerns the graph G_e, depends only on these distance vectors and not on the vertices x_1 and x_2 themselves.)
8. There are at most z edges of E_e^1 inside the internal faces of G_e^+ (clearly, this last condition becomes void when $q = \infty$).
(This information helps us control Condition D during the algorithm.)

Notice that in case $X_e = \emptyset$ the only graph that can correspond to the 6th step is the graph $(\{\varnothing\}, \emptyset)$ which, from now on will be denoted by $G_\varnothing$.

Bounding the set of characteristics. Our next step is to bound $\mathfrak{T}(e)$ for each $e \in E(T)$. Notice first that $|X_e| = |\mathbf{A}_e| \leq b$. This means that there are $2^{O(b \log b)}$ instantiations of $(\mathcal{X}, \chi)$ and $2^{O(k + b \log b)}$ instantiations of $(\mathcal{A}, \alpha)$. As we previously noticed, the different instantiations of $(\mathcal{F}, \mathcal{E})$ are $|\Phi_e| = 2^{2^{O(b \log d)}}$. Moreover, there are $2^{O(b^2 \log d)}$ instantiations of δ and $\alpha(q)$ instantiations of z. We conclude that there exists a function f such that for each $e \in V(T)$, $|\mathfrak{T}(e)| \leq f(k, q, b, d)$. Moreover, $f(k, q, b, d) = \alpha(q) \cdot 2^{O(b^2 \log d) + 2^{O(b \log d)}}$.

The characteristic function on the root edge. Observe that E_{new} is (k, d, q, w)-edge colorable in H if and only if $\mathfrak{T}(e_r) \neq \emptyset$, i.e., $((\emptyset, \varnothing), (\emptyset, \varnothing), G_\varnothing, \varnothing, z) \in \mathfrak{T}(e_r)$ for some $z \leq q$. Indeed, if this happens, conditions 1–4 become void while conditions 5, 6, 7, and 8 imply that $H = H_e$ satisfies the conditions A, B, C, and D respectively in the definition of the (k, d, q, w)-edge colorability of E^{new}.

The computation of the tables. We will now show how to compute $\mathfrak{T}(e)$ for each $e \in E(T)$.

We now give the definition of $\mathfrak{T}(e)$ in the case where e is a leaf of T is the following: Given a $q \in \mathbb{N} \cup \{\infty\}$, we define $A(q) = \{\infty\}$ if $q = \infty$, otherwise $A(q) = \{z \mid z \leq q\}$.

Suppose now that e_l is a leaf-edge of T where $\pi(e_l) = [a_1, a_2, a_1]$ and $\mathbf{A}_{e_l} = \{a_{1,2}, a_{2,1}\}$.

1. If $\{a_1, a_2\} \in E_e^{\text{old}}$, then

$$\begin{aligned}\mathfrak{T}(e_l) \quad = \quad \{ \big(& (\{\{a_1\}, \{a_2\}\}, \{(1,1), (2,1)\}), \\ & (\{\{a_{1,2}\}, \{a_{2,1}\}\}, \{(1,0), (2,0)\}), \\ & \big(\{\{(a_1, 0), (a_2, \mathsf{w}(\{a_1, a_2\}))\}, \{(a_1, \mathsf{w}(\{a_1, a_2\})), (a_2, 0)\}\}, \emptyset\big), \\ & \{((a_1, a_2), \mathsf{w}(\{a_1, a_2\}))\}, z\big) \mid z \in A(q)\},\end{aligned}$$

2. if $\{a_1,a_2\} \in E_e^{\text{new}}$ and $\{a_1,a_2\} \subseteq V_e^{\text{old}}$, then $\mathfrak{T}(e_l) = \mathcal{Q}^1 \cup \mathcal{Q}^\infty$ where

$$\begin{aligned}
\mathcal{Q}^1 \quad = \quad \{ \big(& (\{\{a_1\},\{a_2\}\},\{(1,1),(2,1)\}) \\
& (\{\{a_{1,2},a_{2,1}\}\},\{(1,1)\}) \\
& (\{\{(a_1,0),(a_2,1)\},\{(a_1,1),(a_2,0)\}\},\emptyset) \\
& \{((a_1,a_2),s)\},z\big) \mid z \in A(q) - \{0\}\} \\
\mathcal{Q}^\infty \quad = \quad \{ \big(& (\{\{a_1\},\{a_2\}\},\{(1,1),(2,1)\}) \\
& (\{\{a_{1,2},a_{2,1}\}\},\{(1,0)\}) \\
& (\{\{(a_1,0),(a_2,d+1)\},\{(a_1,d+1),(a_2,0)\}\},K) \\
& \{((a_1,a_2),d+1)\},z\big) \mid z \in A(q)\}
\end{aligned}$$

(the set K above contains a single edge that is not a loop), and if $\{a_1,a_2\} \in E_e^{\text{new}}$ and $\{a_1,a_2\} \not\subseteq V_e^{\text{old}}$, then $\mathfrak{T}(e_l) = \mathcal{Q}^1 \cup \mathcal{Q}^\infty \cup \mathcal{Q}^0$ where

$$\begin{aligned}
\mathcal{Q}^0 \quad = \quad \{ \big(& (\{\{a_1,a_2\}\},\{(1,1-\langle\{a_1,a_2\} \subseteq V_e^{\text{new}}\rangle)\}) \\
& (\{\{a_{1,2},a_{2,1}\}\},\{(1,0)\}) \\
& (\{\{(a_1,0),(a_2,0)\}\},\emptyset) \\
& \{((a_1,a_2),0)\},z\big) \mid z \in A(q)\}.
\end{aligned}$$

Assume now that e is a non-leaf edge of T with children e_l and e_r, the collection $\mathfrak{T}(e)$ is given by $\mathbf{join}(\mathfrak{T}(e_1),\mathfrak{T}(e_2))$ where **join** is a procedure that is depicted below. Notice that $\mathbf{A}_e$ is the symmetric difference of $\mathbf{A}_{e_l}$ and $\mathbf{A}_{e_r}$ and X_e consists of the endpoints of the arcs in $\mathcal{A}_e$. We also set $X_e^F = (X_{e_l} \cup X_{e_r}) \setminus X_e$.

Procedure **join**
Input: two collections $\mathcal{C}_{e_l}$ and $\mathcal{C}_{e_r}$ of (d,k,q)-configurations of $(X_{e_l},\mathbf{A}_{e_l})$ and $(X_{e_r},\mathbf{A}_{e_r})$.
Output: a collection $\mathcal{C}_r$ of (d,k,q)-configurations of $(X_e,\mathbf{A}_e)$
(1) set $\mathcal{C}_e = \emptyset$
(2) for every pair $(Q_{e_l},Q_{e_r}) \in \mathcal{C}_{e_l} \times \mathcal{C}_{e_r}$, if $\mathbf{merge}(Q_{e_l},Q_{e_r}) \neq \mathsf{void}$,
then let $\mathcal{C}_e \leftarrow \mathcal{C}_e \cup \{\mathbf{merge}(Q_{e_l},Q_{e_r})\}$.
(3) return $\mathcal{C}_e$

It remains to describe the routine **merge**. For this, assume that it receives as inputs the (d,k,q)-configurations $Q_l = ((\mathcal{X}_l,\chi_l),(\mathcal{A}_l,\alpha_l),(\mathcal{F}_l,\mathcal{E}_l),\delta_l,z_l)$ and $Q_r = ((\mathcal{X}_r,\chi_r),(\mathcal{A}_r,\alpha_r),(\mathcal{F}_r,\mathcal{E}_r),\delta_r,z_r)$ of $(X_{e_l},\mathbf{A}_{e_l})$ and $(X_{e_r},\mathbf{A}_{e_r})$ respectively. Procedure $\mathbf{merge}(Q_{e_l},Q_{e_r})$ returns a (d,k,q)-configuration $((\mathcal{X},\chi),(\mathcal{A},\alpha),(\mathcal{F},\mathcal{E}),\delta,z)$ of $(X_e,\mathbf{A}_e)$ constructed as follows:

1. If $z_r + z_r > q$, then return void, otherwise $z = z_l + z_r$ (This controls the number of 1-edges that are now contained in Δ_e)
2. Let $(\mathcal{X}',\chi') = (\mathcal{X}_l,\chi_l) \oplus (\mathcal{X}_r,\chi_r)$ and if $\chi'^{-1}(2) \neq \emptyset$ then return void.
(This compute the "fusion" of the connected components of $(V(H_{e_l},E^0_{e_l}))$ and $(V(H_{e_r},E^0_{e_r}))$ with vertices in V_{e_l} and V_{e_r} and makes sure that none of the created components contains 2 or more 0-vertices.)
3. Let $(\mathcal{X},\chi) = (\mathcal{X}'_l,\chi'_l)|_{V_e}$
(This computes the fusion $(\mathcal{X}'_l,\chi'_l)$ is restricted on the boundary O_e of Δ_e.)
4. Let $(\mathcal{A}',\alpha') = (\mathcal{A}_l,\alpha_l) \oplus (\mathcal{A}_r,\alpha_r)$ and if $\alpha'^{-1}(k+1) \neq \emptyset$ then return void.
5. Let $(\mathcal{A},\alpha) = (\mathcal{A}_l,\alpha_l) \oplus (\mathcal{A}_r,\alpha_r)|_{\mathbf{A}_e}$.
6. Compute the function $\gamma : (\mathcal{F}_{e_l} \cup \mathcal{F}_{e_r} \cup X_e) \times (\mathcal{F}_{e_l} \cup \mathcal{F}_{e_r} \cup X_e) \to \{0,\dots,d+1\}$, whose description is given latter.
7. Take the disjoint union of the graphs $(\mathcal{F}_l,\mathcal{E}_l)$ and $(\mathcal{F}_r,\mathcal{E}_r)$ and remove from it every edge $\{\phi_1,\phi_2\}$ for which $\gamma(\phi_1,\phi_2) \leq d$. Let $\mathcal{G}^+ = (\mathcal{F}^+,\mathcal{E}^+)$ be the obtained graph.

8. If for some edge $\{\phi_1, \phi_2\} \in \mathcal{E}^+$ it holds that for every $x_1, x_2 \in V_e$, $\gamma(\phi_1, x_1) + \gamma(\phi_2, x_2) > d$, then return void.
9. Consider the function $\lambda : \mathcal{F}_l \cup \mathcal{F}_r \to \{1, \ldots, d\}^{X_e}$ such that $\lambda(\phi) = \{(x, \gamma(\phi, x)) \mid x \in X_e\}$.
10. For every $\phi' \in \lambda(\mathcal{F}_l \cup \mathcal{F}_r)$, do the following for every set $\mathsf{F} = \lambda^{-1}(\phi')$: identify in $\mathcal{G}^+$ all vertices in F and if at least one pair of them is adjacent in $\mathcal{G}^+$, then add an loop on the vertex created after this identification. Let $\mathcal{G} = (\mathcal{F}, \mathcal{E})$ be the resulting graph (notice that $\mathcal{F} = \lambda(\mathcal{F}_l \cup \mathcal{F}_r)$).
11. $\delta = \{((x, x'), \gamma(x, x')) \mid x, x' \in V_e\}$.

The definition of function γ. We present here the definition of the function γ used in the above description of the tables of the dynamic programming procedure.

Given a non-empty set X and $q \in \{0, 1\}$ we define

$$\begin{aligned}\mathsf{ord}^q(X) \quad &= \quad \{\pi \mid \exists X' \subseteq X : X' \neq \emptyset \wedge \ |X'| \bmod 2 = q \\ &\qquad \wedge\ \pi \text{ is an ordering of } X'\}\end{aligned}$$

Given γ_l and γ_r, we define $\gamma : (\mathcal{F}_{e_l} \cup \mathcal{F}_{e_r} \cup X_e) \times (\mathcal{F}_{e_l} \cup \mathcal{F}_{e_r} \cup X_e) \to \{0, \ldots, d+1\}$ by distinguishing the following cases:

1. If $(x \in X_e \setminus X_{e_r} \wedge \phi \in \mathcal{F}_{e_l})$ or $(x \in X_e \setminus X_{e_l} \wedge \phi \in \mathcal{F}_{e_r})$, then
$$\begin{aligned}\gamma(\phi, x) \quad = \quad &\min \big\{\phi(x), \min\{\phi(p_1) + \sum_{[\![1,\rho-1]\!]} \delta_{\mathbf{s}(i)}(p_i, p_{i+1}) + \\ &\delta_{\mathbf{s}(\rho)}(p_\rho, x) \mid [p_1, \ldots, p_\rho] \in \mathsf{ord}^0(X_e^F)\}\big\},\end{aligned}$$
where $\mathbf{s}(i) = $ "l" if $\langle x \in X_e \setminus X_{e_l}\rangle = (i \bmod 2)$, otherwise $\mathbf{s}(i) = $ "r".
2. If $(x \in X_e \setminus X_{e_l} \wedge \phi \in \mathcal{F}_{e_l})$ or $(x \in X_e \setminus X_{e_r} \wedge \phi \in \mathcal{F}_{e_r})$, then
$$\begin{aligned}\gamma(\phi, x) \quad = \quad &\min \big\{\phi(p_1) + \sum_{[\![1,\rho-1]\!]} \delta_{\mathbf{t}(i)}(p_i, p_{i+1}) + \delta_{\mathbf{t}(\rho)}(p_\rho, x) \\ &\mid [p_1, \ldots, p_\rho] \in \mathsf{ord}^1(X_e^F)\}\big\},\end{aligned}$$
where $\mathbf{t}(i) = $ "l" if $\langle x \in X_e \setminus X_{e_l}\rangle \neq (i \bmod 2)$, otherwise $\mathbf{t}(i) = $ "r".
3. If x is one of the (at most two) vertices in $(X_{e_r} \cap X_{e_r}) \setminus X_e^F$ and $\phi \in \mathcal{F}_{e_l} \cup \mathcal{F}_{e_r}$, then
$$\begin{aligned}\gamma(\phi, x) \quad = \quad &\min \big\{\phi(x), \\ &\min\{\phi(p_1) + \sum_{[\![1,\rho-1]\!]} \delta_{\mathbf{u}(i)}(p_i, p_{i+1}) + \delta_{\mathbf{u}(q)}(p_\rho, x) \\ &\mid [p_1, \ldots, p_\rho] \in \mathsf{ord}^q(X_e^F)\} \mid q \in \{0, 1\}\big\}\end{aligned}$$
where $\mathbf{u}(i) = $ "r" if $\langle \phi \in \mathcal{F}_{e_l}\rangle = (i \bmod 2)$, otherwise $\mathbf{u}(i) = $ "l".
4. If $\phi, \phi' \in \mathcal{F}_l \cup \mathcal{F}_r$, then
$$\begin{aligned}\gamma(\phi, \phi') \quad = \quad &\min \big\{\phi(p_1) + \sum_{[\![1,\rho-1]\!]} \delta_{\mathbf{u}(i)}(p_i, p_{i+1}) + \phi'(p_\rho) \\ &\mid [p_1, \ldots, p_\rho] \in \mathsf{ord}^q(X_e^F)\big\}\end{aligned}$$
In this equality, $q = 1$ if ϕ and ϕ' belong in different sets in $\{\mathcal{F}_l, \mathcal{F}_r\}$, otherwise $q = 0$. The function $\mathbf{u}$ is the same as in the previous case.
5. If $x_1, x_2 \in X_e \setminus X_{e_r}$ or $x_1, x_2 \in X_e \setminus X_{e_l}$, then
$$\begin{aligned}\delta(x_1, x_2) \quad = \quad &\min \big\{\delta_{\mathbf{y}(0,x_1)}(x_1, x_2), \min\{\delta_{\mathbf{y}(0,x_1)}(x_1, p_1) + \\ &\sum_{i \in [\![1,\rho-1]\!]} \delta_{\mathbf{y}(i,x_1)}(p_i, p_{i+1}) + \\ &\delta_{\mathbf{y}(0,x_2)}(p_\rho, x_2) \mid [p_1, \ldots, p_\rho] \in \mathsf{ord}^0(X_e^F)\}\big\}\end{aligned}$$
In this equality $\mathbf{y}(i, x) = $ "l" if $\langle x \in X_e \setminus X_{e_r}\rangle = \langle i \bmod 2 = 0\rangle$ otherwise $\mathbf{y}(i, x) = $ "r".

6. If x_1, x_2 belong in different sets is $\{X_e \setminus X_{e_r}, X_e \setminus X_{e_l}\}$, then
$$\delta(x_1,x_2) = \min\{\delta_{\mathbf{y}(0,x_1)}(x_1,p_1) + \sum_{[\![1,\rho-1]\!]} \delta_{\mathbf{y}(i,x_1)}(p_i,p_{i+1}) + \delta_{\mathbf{y}(0,x_2)}(p_\rho,x_2) \mid [p_1,\ldots,p_\rho] \in \mathsf{ord}^1(X_e^F)\}$$
The function $\mathbf{y}$ is the same as in the previous case.
7. If exactly one, say x_2, of x_1, x_2 belongs in $X_{e_r} \cap X_{e_r}) \setminus X_e^F$, then
$$\delta(x_1,x_2) = \min\Big\{\delta_{\mathbf{y}(0,x_1)}(x_1,x_2), \min\{\min\{\delta_{\mathbf{y}(0,x_1)}(x_1,p_1) + \sum_{[\![1,\rho-1]\!]} \delta_{\mathbf{y}(i,x_1)}(p_i,p_{i+1}) + \delta_{\mathbf{y}(0,x_2)}(p_\rho,x_2) \,|\, [p_1,\ldots,p_\rho] \in \mathsf{ord}^q(X_e^F)\} \,|\, q \in \{0,1\}\}\Big\}$$
The function $\mathbf{y}$ is the same as in the two previous cases. In case x_1 belongs in $X_{e_r} \cap X_{e_r}) \setminus X_e^F$, then just swap the positions of x_1 and x_2 in the above equation.
8. If both x_1, x_2 belong in $X_{e_r} \cap X_{e_r}) \setminus X_e^F$, then
$$\delta(x_1,x_2) = \min\{\delta_l(x_1,x_2), \delta_r(x_1,x_2), \min\{\min\{\delta_{\mathbf{z}(0,j)}(x_1,p_1) + \sum_{[\![1,\rho-1]\!]} \delta_{\mathbf{z}(i,j)}(p_i,p_{i+1}) + \delta_{\mathbf{z}(q,j)}(p_\rho,x_2) \,|\, [p_1,\ldots,p_\rho] \in \mathsf{ord}^q(X_e^F)\} \,|\, (q,j) \in \{0,1\}^2\}\}$$
In the previous equality, $\mathbf{z}(i,j) =$ “l” if $(i + j \bmod 2) = 0$, otehrwise $\mathbf{z}(i,x) =$ “r”.

Running time analysis. It now remains to prove that procedure **join** runs in $(\alpha(q))^2 \cdot 2^{O(k^2)+2^{O(b\log d)}}$ steps. Recall that there exists a function f such that $|\mathfrak{T}(e)| \leq f(k,q,b,d)$. Therefore **merge** will be called in Step (2) at most $(f(k,q,b,d))^2$ times. The first computationally non-trivial step of **merge** is Step 5, where function γ is computed. Notice that γ has at most $((d+1)^{|X_{e_l}|} + (d+1)^{|X_{e_r}|} + |X_e|)^2 = 2^{O(b\cdot\log d)}$ entries and each of their values require running over all permutations of the subsets of X_e^F that are at most $b! = 2^{O(b\cdot\log b)}$. These facts imply that the computation of γ takes $2^{O(b\cdot\log b)}$ steps. As Steps 6–10 deal with graphs of $2^{O(b\cdot\log d)}$ vertices, the running time of **join** is the claimed one. ◀

We are now in position to prove the main algorithmic result of this paper.

Proof of Theorem 2. Given an input $I = (G,q,k,d)$ of BBFPDC, we run the algorithm of Lemma 4 with G and k as input. Let $H = G_k$ be the output of this algorithm. From the same lemma, he construction of H takes $O(k^2 n)$ steps. Then we run the algorithm of Proposition 5 with (H,w) as input, where $w = c_1 \cdot c_2 \cdot k \cdot d$. If the answer is that $\mathbf{bw}(H) > w$, then, from Lemma 4, $\mathbf{tw}(G) > c_1 \cdot d$, therefore, from Lemma 3, we can safely report that I is a NO-instance. If the algorithm of Proposition 5 outputs a sphere-cut decomposition $D = (T,\mu)$ of width at most $w = O(k \cdot d)$ then we call the dynamic programming algorithm of Lemma 6, with input (G,H,q,k,d,D,b). This, from Lemma 4, provides an answer to BBFPDC for the instance I in $(\alpha(q))^2 \cdot 2^{O((kd)^2\log d)+2^{O((kd)\log d)}} \cdot n = (\alpha(q))^2 \cdot 2^{2^{O((kd)\log d)}} \cdot n$ steps and this completes the proof of the theorem. ◀

6 Discussion

We remark that our algorithm still works for the classic PDC problem when the face-degree of the input graph is bounded. For this we define the following problem:

Bounded Face BDC (FPDC)
Input: a plane graph G with face-degree at most $k \in \mathbb{N}_{\geq 3}$, and $d \in \mathbb{N}$
Question: is it possible to add edges in G such that the resulting embedding remains plane and has diameter at most d?

We directly have the following corollary of Theorem 2.

▶ **Theorem 7.** *It is possible to construct an* $O(n^3) + 2^{2^{O((kd)\log d)}} \cdot n$*-step algorithm for* FPDC.

To construct an FPT-algorithm for PDC when parameterized by d remains an insisting open problem. The reason why our approach does not apply (at least directly) for PDC is that, as long as a completion may add an arbitrary number of edges in each face, we cannot guarantee that our dynamic programming algorithm will be applied on a graph of bounded branchwidth. We believe that our approach and, in particular, the machinery of our dynamic programming algorithm, might be useful for further investigations on this problem.

All the problems in this paper are defined on plane graphs. However, one may also consider the "non-embedded" counterparts of the problems PDC and BPDC by asking that their input is a planar combinatorial graphs (without a particular embedding). Similarly, such a counterpart can also be defined for the case of BFPDC if we ask whether the completion has an embedding with at most k new edges per face. Again, all these parameterized problems are known to be (non-constructively) in FPT, because of the results in [13, 11]. However, our approach fails to design the corresponding algorithms as it strongly requires an embedding of the input graph. For this reason we believe that even the non-embedded versions of BPDC and BFPDC are as challenging as the general Planar Diameter Completion problem.

Acknowledgement. We would like to thank the anonymous referees of an earlier version of this paper for their remarks and suggestions that improved the presentation of the paper.

References

1 F. R. K. Chung. Diameters of graphs: Old problems and new results. *Congressus Numerantium*, 60, 1987.
2 Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. *Handbook of Graph Grammars*, pages 313–400, 1997.
3 Italo J. Dejter and Michael R. Fellows. Improving the diameter of a planar graph. Manuscript, May 1993.
4 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.
5 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
6 Yong Gao, Donovan R. Hare, and James Nastos. The parametric complexity of graph diameter augmentation. *Disc. Appl. Math.*, 161(10-11):1626–1631, 2013.
7 Petr A. Golovach, Clément Requilé, and Dimitrios M. Thilikos. Variants of plane diameter completion. *CoRR*, abs/1509.00757, 2015.
8 Qian-Ping Gu and Hisao Tamaki. Optimal branch-decomposition of planar graphs in $O(n^3)$ time. *ACM Transactions on Algorithms*, 4(3), 2008.
9 Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. On the minimum-cardinality-bounded-diameter and the bounded-cardinality-minimum-diameter edge addition problems. *Oper. Res. Lett.*, 11(5):303–308, 1992.

10 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

11 Neil Robertson and Paul D. Seymour. Graph minors XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

12 Neil Robertson and Paul D. Seymour. Graph Minors. XIII. The disjoint paths problem. *J. Combin. Theory, Ser. B*, 63(1):65–110, 1995.

13 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Combin. Theory Ser. B*, 92(2):325–357, 2004.

14 Anneke A. Schoone, Hans L. Bodlaender, and Jan van Leeuwen. Diameter increase caused by edge deletion. *Journal of Graph Theory*, 11(3):409–427, 1987.

15 Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.