Electronic Colloquium on Computational Complexity, Report No. 91 (2019)

A Sublinear-space and Polynomial-time Separator Algorithm for Planar Graphs

Ryo Ashida[*], Tatsuya Imai[†], Kotaro Nakagawa[‡], A. Pavan[§], N. V. Vinodchandran[¶], and Osamu Watanabe[‖]

Abstract

In [12], the authors presented an algorithm for the reachability problem over directed planar graphs that runs in polynomial-time and uses $O(n^{1/2+\epsilon})$ space. A critical ingredient of their algorithm is a polynomial-time, $\widetilde{O}(\sqrt{n})$-space algorithm to compute a separator of a planar graph. The conference version provided a sketch of the algorithm and many nontrivial details were left unexplained. In this work, we provide a detailed construction of their algorithm.

1 Introduction

The purpose of this paper is to give a detail construction of an $\widetilde{O}(\sqrt{n})$-space and polynomial-time algorithm for computing an $O(\sqrt{n})$-size separator for a given planar graph with n vertices. We prove the following theorem.

Theorem 1.1. *For some constant $\alpha > 0$, there exists an algorithm that takes an undirected planar graph with n vertices as input and outputs its α-separator of size $O(\sqrt{n})$ in polynomial-time and $\widetilde{O}(\sqrt{n})$-space.*

Remark. *In this paper, we have not optimized the constant α and prove the theorem with $\alpha = 1/31$. Throughout this paper, we will use n to denote the number of vertices of an input graph, which will be used as a size parameter when we discuss graph problems. For measuring space complexity, we will use, for simplicity, the number of words used as working memory, where we assume that each word is large enough to contain an integer up to n, for specifying the index of a vertex of an input graph. By "$\widetilde{O}(\sqrt{n})$-space" we mean that the computation that uses $O(\sqrt{n})$ words as working memory, which is $O(\sqrt{n}\log n)$-space in the standard Turing machine model.*

The *Separator Theorem* states that any planar graph G with n vertices has a *separator* of size $O(\sqrt{n})$, that is, a set S of $O(\sqrt{n})$ vertices of G such that by removing S, G is split into disconnected subgraphs of almost equal size, say, each having more than $n/3$ vertices. In fact, in their seminal work that first proved the Separator Theorem, Lipton and Tarjan [14] gave an efficient separator algorithm, an algorithm for computing an $O(\sqrt{n})$-size separator for planar graphs. The notion of planar separator and the algorithm of Lipton and Tarjan have found several applications in designing efficient algorithms for planar graphs.

Since the work of Lipton and Tarjan, several versions of separator algorithms have been proposed [8, 13, 10, 9, 11], and they have been applied to design various algorithms for planar graphs. The purpose of this paper is to give yet another type of separator algorithm. The motivation for the new separator algorithm stems from its relation to the graph reachability problem.

The *graph reachability* or the *st-connectivity* problem asks, for a given graph G and two of its vertices s and t, whether there is a path in G from s to t or not. (Unless specified otherwise, we assume here that an input of the graph reachability problem is a *directed* graph.) This problem is one of the fundamental computational problems in both algorithm design and in complexity theory. In

[*]HEROZ Inc, `rashida2810@gmail.com`
[†]HEROZ Inc, `tatsuya.imai.30100041@gmail.com`
[‡]JMA SYSTEMS Co. `kootaroonakagawa@gmail.com`
[§]Iowa State University, `pavan@cs.iastate.edu`
[¶]University of Nebraska-Lincoln (part of the work done when the author was visiting Johns Hopkins University), `vinod@cse.unl.edu`
[‖]Tokyo Institute of Technology, `watanabe@c.titech.ac.jp`

ISSN 1433-8092

particular, this problem is known to be complete (under logspace reductions) for the complexity class NL—nondeterministic logspace. Thus a $O(\log n)$-space algorithm for graph reachability implies that the complexity class NL equals L, and resolves one of the fundamental open problems in computational complexity theory. Interestingly, for its undirected version, that is, for the *undirected* graph reachability problem, Reingold [15] gave a remarkable $O(\log n)$-space algorithm. Note that, by using standard algorithmic techniques such as BFS or DFS we can design an algorithm that runs in almost linear-space and in almost linear-time for solving reachability over directed graphs. Savitch gave an algorithm that solves reachability using $O((\log n)^2)$-space. However Savitch's algorithm takes super polynomial-time. While the BFS/DFS algorithm is time-efficient, Savitch's algorithm is space-efficient. Thus it is natural to ask whether there exists an algorithm, for the directed graph reachability problem, that is efficient in both time and space. In a survey article on the graph reachability and related problems, Wigderson [16] asked whether there is a sublinear-space and polynomial-time algorithm for the graph reachability problem. For this question, the best known answer is the algorithm given by Barnes et al. [6] that runs in $O\big(n/2^{\sqrt{\log n}}\big)$-space and polynomial-time. Unfortunately, though the bound $O\big(n/2^{\sqrt{\log n}}\big)$ is "sublinear", it is quite close to being linear. If we view $O\big(n^{1-\varepsilon}\big)$ (for some $\varepsilon > 0$), as "sublinear", it has been open whether such a sublinear-space and polynomial-time algorithm exists for reachability problem.

Recently, there have been some advancements on this sublinear-space and polynomial-time computability for restricted graph classes. (Since all algorithms stated below run in polynomial-time[1], we omit mentioning "polynomial-time" in the following.) The first break through was given by Asano and Doerr [2], who showed an algorithm that solves the reachability problem on directed grid graphs in $\widetilde{O}(n^{1/2+\epsilon})$-space. Inspired by this work, Imai, Nakagawa, Pavan, Vinodchandran, and Watanabe [12] gave an $\widetilde{O}(n^{1/2+\epsilon})$-space algorithm for the reachability problem on directed planar graphs. Later Asano et al. [3] gave an improved $\widetilde{O}(\sqrt{n})$-space algorithm. For directed grid graphs, Ashida and Nakano [5] showed an $\widetilde{O}(n^{1/3})$-space algorithm. Note that in all of the above algorithms the input graphs are planar and they all critically rely on the existence of $O(\sqrt{n})$-size separator. The last algorithm of Ashida and Nakano is interesting in this sense because its space bound is much less than the separator size. Furthermore, these algorithms except for the first one use a separator algorithm proposed by Imai *et al* in [12] that produces a $O(\sqrt{n})$-size separator in $\widetilde{O}(\sqrt{n})$-space. This separator algorithm is based on an algorithm of Gazit and Miller [10] that is designed for computing a separator efficiently on a parallel computation model. Since space-bounded computation and parallel computation share common features, one can expect that the algorithm of Gazit and Miller can be modified naturally to obtain an algorithm that runs in polynomial-time and uses $\widetilde{O}(\sqrt{n})$-space. Unfortunately, it is not easy to translate Gazit and Miller's algorithm to obtain a $\widetilde{O}(\sqrt{n})$-space algorithm for computing a planar separator. The work of [12] provided a sketch of a $\widetilde{O}(\sqrt{n})$-space, polynomial-time algorithm for computing a planar separator, however, several nontrivial details have been left unexplained in that work.

In this paper we completely reconsider the design of a separator algorithm that runs in $\widetilde{O}(\sqrt{n})$-space and polynomial-time on planar graphs. While we borrow many ideas from Gazit and Miller [10] and Imai *et al* [12], our algorithm in this paper differs from both the algorithms in many technical details. In Section 3, we give an outline of the separator algorithm and explain each of the steps and sub-steps in latter sections, so that the interest reader can check our constructions carefully.

2 Preliminaries

In this section we explain basic notions and notation for discussing planar graphs. We then discuss our computational model and explain some assumptions that we can introduce on input data for our separator algorithm. We also recall two separator algorithms from the literature.

Throughout this paper, by a graph we usually mean an undirected graph. For any graph G, we formally use $\mathrm{V}(G)$ and $\mathrm{E}(G)$ to denote the set of vertices of G and the set of edges of G respectively. On the other hand, we will mainly discuss with some fixed graph G, and in this case, we simply use V and E to denote $\mathrm{V}(G)$ and $\mathrm{E}(G)$. For any vertex $v \in V$, let $\mathrm{N}_G(v)$ denote the set of its *neighbors*, namely, vertices adjacent to v in G. For any $U \subseteq V$ (resp., $D \subseteq E$) we use $G[U]$ (resp., $G[D]$) to denote a subgraph of G *induced* by U (resp., by D).

[1]Unfortunately, while their running time are polynomially bounded, these bounds are quite high degree polynomials except the first one by Asano and Doerr.

We assume that an input graph is *simple*, that is, each pair of vertices has at most one edge. Though a non-simple graph may be created by our transformations (in particular, in our base dual-graph), all our arguments will be valid by treating multiple edges (between the same pair of vertices) as different edges. By a *path* of $G = (V, E)$, we mean a "simple" path, that is, a sequence adjacent edges of E where no vertex is visited more than once. Similarly, a *cycle* of $G = (V, E)$ is a sequence adjacent edges of E that comes back to the first vertex where no vertex except for the first one is visited more than once. We simply represent a path and a cycle by a sequence $(v_1, \ldots, v_m)$ of vertices of V such that every $\{v_i, v_{i+1}\}$ (and also $\{v_m, v_1\}$ for a cycle) is an edge of E. Although the same sequence could be used for representing both a path and a cycle, the difference should be clear from the context. We may also use this sequence representation for indicating the "direction" of a path or a cycle.

While our argument should be mathematically precise, we would like to avoid rigorous but tedious topological treatments of planar graphs. Thus, following the previous work in the literature (see, e.g., [13]) we prepare a framework for discussing plane graphs in a combinatorial way.

Definition 2.1 (Planar graph, plane graph, and combinatorial embedding). *A graph G is* planar *if it has a* planar embedding, *a way to arrange the vertices of $\mathrm{N}_G(v)$ on the plane (i.e., $\mathbb{R}^2$) for all $v \in V$ and a drawing of the edges so that no pair of edges intersect each other except at their endpoints. A planar embedding is specified by a* combinatorial embedding, *that is, a set $\{\pi_v : v \in V\}$ of lists, where each π_v is an enumeration of vertices of $\mathrm{N}_G(v)$ in the clockwise order around v under the embedding.*
Remark. *Throughout this paper, we will use a* plane graph *to mean a planar graph that is embedded in the plane following one of such combinatorial embeddings.*

Consider any plane graph G embedded in the plane under a combinatorial embedding $\{\pi_v : v \in V\}$. One of the keys for discussing plane graphs is a way to define the faces of G. Intuitively, the graph G (embedded in the plane) separates the plane into "subplanes," each of which is a "face" of G. Here we formally define the notion of "face" as follows.

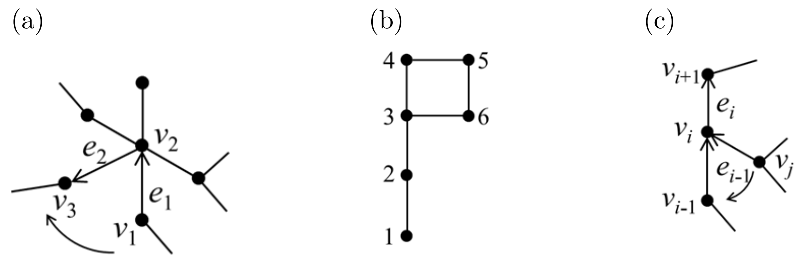
First define the notion of "left-traverse" of G. Consider any edge $\{v_1, v_2\}$ of G, and fix its direction as, e.g., (v_1, v_2), which we regard as the first directed edge e_1 (of the left-traverse). Consider the enumeration π_{v_2} of adjacent vertices of v_2, and let v_3 be the next vertex of v_1 in the enumeration, that is, v_3 is clockwise the first vertex adjacent to v_2 after v_1. Then define $e_2 = (v_2, v_3)$ as the second directed edge. Continue this process of identifying directed edges (based on edges of G) until we come back to the first directed edge e_1. We call this process *left-traverse process*, and a sequence of vertices visited during the process *left-traverse* of G started from (v_1, v_2). Precisely speaking, we do not include the last vertex, i.e., the starting vertex v_1 in the sequence; see Figure 1 for examples. Although a left-traverse is a sequence of vertices, this can be regraded as the sequence of the directed edges in the order that they are identified by the left-traverse process. We remark here the following fact.

Lemma 2.1. *Consider any graph G. For any of its left-traverse $(v_1, v_2, \ldots)$, no directed edge (v_i, v_{i+1}) appears more than once. Thus, no infinite loop occurs in any left-traverse process.*

Proof. [2] Assume to the contrary that some directed edge (v_i, v_{i+1}) appears more than once. Let $e_{i-1} = (v_{i-1}, v_i)$ and $e_i = (v_i, v_{i+1})$ be the directed edges identified in the left-traverse process when v_i appears for the first time. From our assumption, we have another (v_j, v_i) for some $j > i + 1$ right after which e_i appears for the second time in the process. Then since v_{i+1} is clockwise the next vertex of v_{i-1} among all vertices adjacent to v_i, it follows that v_j must be located between v_{i+1} and v_{i-1} in the clockwise enumeration π_{v_i} of $\mathrm{N}_G(v_i)$; that is, v_{i+1} cannot be clockwise the next of v_j in π_{v_i} because at least v_{i-1} comes before v_{i+1} after v_j; see Figure 1 (c). Thus, (v_i, v_{i+1}) cannot be next to (v_j, v_i) in the process; a contradiction. □

Consider any plane graph G with at least two faces. Then any left-traverse $t = (v_1, \ldots, v_m)$ of G separates the plane where G is embedded into at least two "subplanes." From the choice of directed edges, it is clear that there is no vertex of G in the "subplane" left of the directed edges of t; that is, a "face" is identified by this left-traverse. In fact, it has been known that for any plane graph, all its faces are identified by some left-traverse as stated in Proposition 2.1 below. In this paper, we regard this proposition as our definition of the notion of face; throughout this paper, we will identify each face by its corresponding left-traverse, which is called the *boundary* of the face.

[2]The lemma may be clear since the same vertex cannot be the "next" vertex of two vertices in an enumeration of a combinatorial embedding. But in order to give a clear image on the left-traverse notion, we give a bit careful proof here.



(a) An example of the first two edges identified by a left-traverse process. As the next directed edge of $e_1 = (v_1, v_2)$, an edge $e_2 = (v_2, v_3)$ is selected, where v_3 is the clockwise next vertex of v_1 among all adjacent vertices of v_2. By the choice of these edges, no vertex adjacent to v_2 exists in the left of $e_1 \to e_2$, which guarantees that the left of each left-traverse is a "face." (b) A connected but not 2-connected graph. A left-traverse from edge $(3, 4)$ is $(3, 4, 5, 6, 3, 2, 1, 2)$, which defines the outer face. On the other hand, the face inside the square is defined by, e.g., a left-traverse from edge $(4, 3)$, that is, $(4, 3, 6, 5)$. (c) An example for the proof of Lemma 2.1.

Figure 1: Left-traverse

Proposition 2.1. *Consider any plane graph G. Then for any left-traverse of G, there is a unique subplane, i.e., face, located left of the traverse w.r.t. its direction. On the other hand, for any face of G, there is a left-traverse of G that defines the face as a subplane located left of the left-traverse w.r.t. its direction.*
Remark. *Since each left-traverse is determined by a starting directed edge, there are more than one essentially the same left-traverses. Thus, the correspondence between faces and left-traverses is not one-to-one but one-to-many.*

Note that a left-traverse is not a cycle in general; for example, Figure 1 (b). But if G is 2-connected, then any left-traverse of G is a cycle; see, e.g., [7]. (A graph is *2-connected* if every pair $\{u, v\}$ of its vertices are connected by two paths not sharing vertices except for $\{u, v\}$, or equivalently, it has no *cut vertex*.)

Proposition 2.2. *Consider any plane graph G that is also 2-connected. Then any left-traverse is a cycle. That is, every face of G has a boundary consisting of one cycle, which we call a* face boundary cycle*.*

Definition 2.2 (Face boundary representation and face size)**.** *For any 2-connected plane graph G, for any face, its* face boundary representation *is its face boundary cycle c, or more precisely, a sequence of vertices of c in the order so that the face is located left of c w.r.t. the order. The* size *of a face is the number of vertices of its face boundary cycle.*

Definition 2.3 (Incidence and edge-incidence)**.** *Two faces are* incident *(resp.,* edge-incident*) if their boundaries share some vertex (resp., some edge).*
Remark. *In general, we say that two graph objects are* incident *if they share some vertex. For example, we say that two paths are incident if they share some vertex (and possibly more).*

Definition 2.4 (Complete face information)**.** *A* complete face information *of a graph G is a list of all face boundary representations of G and lists of respectively all incident and edge-incident pairs of faces of G.*

While it is not essential for our discussion, we note here that a complete face information of any plane graph with n vertices can be expressed in $\widetilde{O}(n)$-space, which is easy to show by the Euler's formula on the number of vertices, edges, and faces of a plane graph.

We introduce a generalized separator notion, which will be mainly used in this paper. First we define the standard separator notion formally.

Definition 2.5 (Standard separator)**.** *For any graph G with n vertices, and for any parameter $\alpha \in (0, 1)$, an α-*separator *of G is a set S of vertices of G such that the removal of S creates two disconnected subgraphs each of which has at least αn vertices.*

A generalized separator notion is defined for weighted graphs. Here a weighted graph[3] is a plane graph G for which we additionally give a weight to each face of G. Formally, we may represent a weighted graph $G = (V, E)$ by V, E, its combinatorial planar embedding, and its complete *weighted* face information where each face representation is also given its weight. By normalizing weights, we may assume that the total weight is always 1.

Definition 2.6 (Cycle weighted separator). *For any 2-connected weighted graph, and for any $\rho \in (0, 1)$, a* cycle weighted ρ-separator *of the graph is a cycle C that creates two subplanes each of which has weight $\geq \rho$.*
Remark. *The weight of a subplane is the sum of the weights of all faces in the subplane.*

Computational model

For discussing sublinear-space algorithms, we use the standard multi-tape Turing machine model. A multi-tape Turing machine consists of a read-only input tape, a write-only output tape, and a constant number of work tapes. The space complexity of this Turing machine is measured by the total number of cells that can be used as its work tapes. Throughout this paper, we will use n to denote the number of vertices of an input graph and use it as the complexity parameter; that is, we will measure the space and time complexity in terms of n (unless stated explicitly in other ways). As a unit for space complexity, we will often use "word", by which we mean $c_0 \log n$ cells, where c_0 is a constant large enough such that index of a vertex (i.e., a number between 1 and n) can be stored in one word. We use $\widetilde{O}(s(n))$ to mean $O(s(n))$ words. Thus, for example, by an "$\widetilde{O}(s(n))$-space algorithm" we formally mean a Turing machine that implements the algorithm by using $O(s(n) \log n)$-size work tapes, i.e., work tapes consisting of $O(s(n) \log n)$ cells in total, for processing an input graph with n vertices.

For the sake of explanation, we will follow a standard convention and describe a sublinear-space algorithm by a sequence of constant number of sublinear-space subroutines $A_1, \ldots, A_k$ such that each A_i computes, from its given input, some output that is passed to A_{i+1} as input. Note that some of these outputs cannot be stored in a sublinear-size work tape; nevertheless, there is a standard way to design a sublinear-space algorithm based on these subroutines. The key idea is to compute intermediate inputs every time when they are necessary. For example, while running A_i, when it is necessary to see the jth bit of the input to A_i, simply execute A_{i-1} (from the beginning) until it yields the desired jth bit on its work tape, and then resume the computation of A_i using this obtained bit. It is easy to see that this computation can be executed in sublinear-space. Furthermore, while a large amount of extra computation time is needed, we can show that the total running time can be polynomially bounded if all subroutines run in polynomial-time.

Two separator algorithms and Reingold's algorithm for the reachability test

We recall two separator algorithms that will be used to compute a separator in our algorithm.

The first one is the algorithm of Lipton and Tarjan [14].

Proposition 2.3 (Lipton-Tarjan separator algorithm). *For any planar graph with n vertices, there exists a $1/3$-separator of size $2\sqrt{2n}$. Furthermore, there exists a polynomial-time algorithm that finds one of such separators by a breadth-first search. The algorithm can be modified so that when a given (breadth-first search) tree of G is given as an additional input, then the algorithm runs in polynomial-time w.r.t. n and in $\widetilde{O}(k)$-space where k is the depth of the BFS tree.*

The second one is based on an algorithm proposed by Miller in [13]. In that paper, Miller showed a linear-time algorithm computing a cycle weighted 1/3-separator for a given 2-connected plane and weighted graph.

Proposition 2.4. *Consider any plane and 2-connected weighted graph with n vertices such that each of its faces has weight $\leq 2/3$ and size $\leq d$. Then there exists a cycle weighted $1/3$-separator of size $2\sqrt{2\lfloor d/2 \rfloor n}$. Furthermore, there exists an algorithm that finds one of such separators in $O(n)$-time (under the unit cost).*

As remarked in Section 1.2 of [10], for any graph with n vertices and n_{face} faces of size $\leq d$, we have $n \leq 2dn_{\text{face}}$, which implies that $2\sqrt{2\lfloor d/2 \rfloor n} \leq \sqrt{8} \cdot d\sqrt{n_{\text{face}}}$. Also looking into the proof of this proposition, we see that the above algorithm operates only on faces of a given weighted graph if

[3]Our explanation is simplified from [13] by assuming zero weight to all vertices and edges.

its complete face information is given, from which we can expect that it can be modified to run in $\widetilde{O}(n_{\text{face}})$-space. This has been indeed proved formally [4] as follows; in this paper, we will refer this algorithm as *Miller's algorithm*.

Proposition 2.5 (Modification of Miller's separator algorithm [4]). *Consider any plane and 2-connected weighted graph with n vertices and n_{face} faces such that each of its faces has weight $\leq 2/3$ and size $\leq d$. Then there exists a weighted cycle $1/3$-separator of size $O(d\sqrt{n_{\text{face}}})$. Furthermore, there exists an algorithm that finds one of such separators in $\widetilde{O}(n_{\text{face}})$-space if a combinatorial planar embedding and a complete face information are given.*

We also recall the log-space algorithm of Reingold [15] for the *undirected graph reachability problem*. More specifically, we have an $O(\log n)$-space algorithm that determines, for a given graph G and a pair of vertices u and v of G, whether u is reachable to v in $O(\log n)$-space. This algorithm will be used several places in the following; we will call it *Reinglod's algorithm* for the undirected graph reachability test.

3 Outline of the Separator Algorithm

We explain the outline of our separator algorithm and state necessary claims to show the correctness of the algorithm. These claims will be restated and proved in the subsequent sections.

Following the outline proposed by Gazit and Miller [10] we design a separator algorithm that consists of the following steps:

1. Check the planarity condition etc. of an input graph, make some modifications on the input graph to create our base graphs, and prepare supplementary information.

2. Transform the graph obtained at Step 1 to generate a frame-graph. (This step is separated into four sub-steps. In some cases, a desired separator of the base dual-graph is created during some sub-step; the algorithm of Lipton and Tarjan is used there.)

3. By using Miller's algorithm, compute a desired weighted separator of the frame-graph, which can be naturally transformed to a separator for the base dual-graph. Then compute a desired separator for the original input graph.

Below we first explain Step 1, a preprocessing step to create two graphs that we will consider in the subsequent steps. We then introduce the notion of "frame-graph", and describe sub-steps of Step 2 to obtain a frame-graph. Finally, we give the proof of our separator theorem by showing how to achieve Step 3 for obtaining desired separators for the base dual-graph and the input graph. Following the convention explained in Preliminaries section, we design an algorithm for each step and each sub-step so that it gets data as input from the previous step and computes data as output passed to the next step in $\widetilde{O}(\sqrt{n})$-space and polynomial-time.

Notational Remark. We would like to use symbols G and $\widetilde{G}$ to denote the base graphs that we will mainly consider in this paper, where G is obtained by transforming the input graph and $\widetilde{G}$ is its dual-graph. Thus, we will use (if necessary) G_{org} to denote the original input graph. We also use n to denote the number of vertices of G (and $\widetilde{n}$ for the number of vertices of $\widetilde{G}$). Again we use n_{org} to denote the number of vertices of G_{org}. As we will see, we have $n = \Theta(n_{\text{org}}) = \Theta(\widetilde{n})$; hence, there is no difference among n, n_{org}, and $\widetilde{n}$ as a complexity parameter, and we will simply use n for discussing the complexity of algorithms.

Step 1 and our base graphs (Corresponding technical section: Section 4)

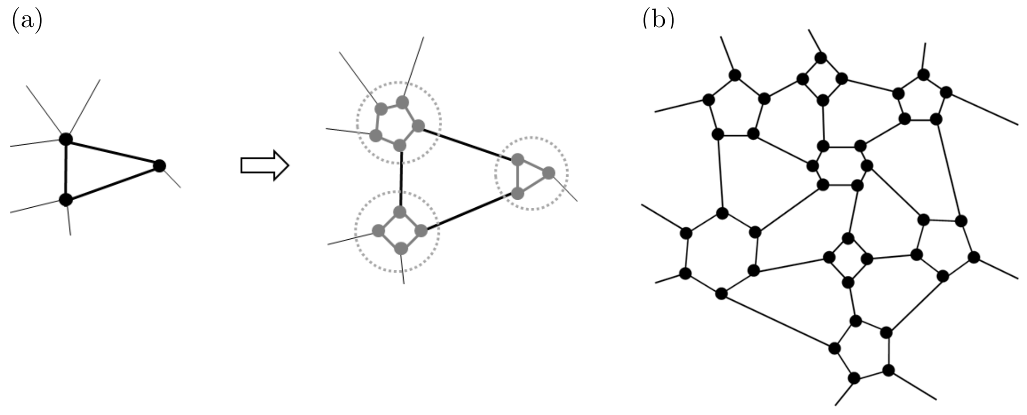
In Step 1 we check the conditions assumed for the input graph G_{org} and prepare two base graphs and some additional information for these graphs. Here Reingold's algorithm for the reachability test plays a key role for achieving this preprocessing in $O(\log n)$-space (and hence, in polynomial-time).

First consider a sub-step for checking input graph conditions. For the graph G_{org} given by its adjacency matrix, we check whether it is connected by using Reingold's algorithm. If G_{org} is not connected, the algorithm considers its largest connected component. If it has less than $30n/31$ vertices, then the algorithm can claim the empty set as a desired separator. Otherwise, the algorithm can work on a subgraph G'_{org} induced by the largest component to obtain its 1/31-separator, which is clearly a 1/31-separator of the original input graph G_{org} (by adding the remaining connected components of G_{org} to a smaller separated subgraph of G'_{org}). Note that identifying the largest connected component and checking its size can be done in $O(\log n)$-space by using Reingold's algorithm.

We then apply the algorithm of Allender and Mahajan [1] for the planarity test. Precisely speaking, they showed an $O(\log n)$-space algorithm for the planarity test by using the undirected graph reachability test as a subroutine, which can be used as a standard $O(\log n)$-space algorithm with Reingold's algorithm. In the same paper, they also gave an $O(\log n)$-space algorithm (by using Reingold's algorithm) that produces a combinatorial planar embedding (if a given graph is planar). Thus, we use their algorithms to check whether the input graph is planar and (if it is so) to compute one of its combinatorial planar embeddings.

In the rest of this paper, we assume that a given input graph G_{org} is connected and planar and that the algorithm is also given one of its combinatorial planar embeddings. (From some technical reason, we also assume that G_{org} is not a trivial single triangle.)

Next we transform the input graph G_{org} into a well-structured graph G. From the above sub-step, we have the combinatorial planar embedding of G_{org}. Then it is not hard to compute its triangulation in $O(\log n)$-space, that is, a combinatorial planar embedding for its triangulated graph (see Section 4 for more explanation). Note here that a triangulated graph is 2-connected, and this 2-connectivity will be preserved in the following sub-steps. As we will see below it would be easier if we can assume that a graph is three-regular. So we transform the obtained triangulated graph to a three-regular graph by vertex expansion shown in Figure 2. Formally, the vertex expansion is defined as follows: Consider any vertex v of a triangulated graph. Since the graph is triangulated (and if it is not a trivial single triangle), we may assume that its degree $\deg(v)$ is at least 3. The *vertex expansion* on v is to replace v with a $\deg(v)$-size face, that is, a face having a boundary cycle consisting of $\deg(v)$ vertices. Each of the $\deg(v)$ edges incident to v is now connected to each vertex on the face boundary cycle following their order under the assumed combinatorial planar embedding. We expand all vertices of the triangulated G_{org} in this way, and the obtained graph is our first *base graph* G.



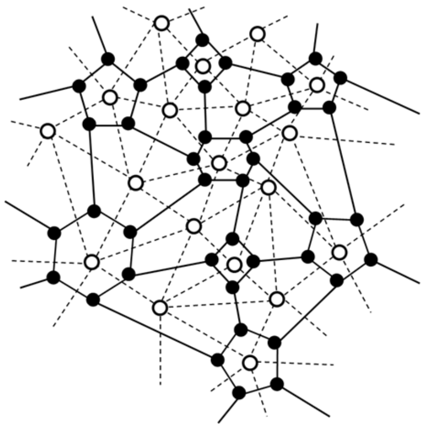
(a) An example of vertex expansion. Each vertex of the left graph is expanded to respectively a triangle, a square, and a pentagon in the right graph. (b) An example of a base graph G_{org} (only its subgraph). All faces are either a face obtained by the vertex expansion or a polygon (triangle $\sim$ hexagon) corresponding to a triangle of G_{org}.

Figure 2: Vertex expansion

Here we point out some properties that are easy to see from Figure 2. That is, (i) G is three-regular, and (ii) all faces of G are either a face obtained by expanding the corresponding vertex of G_{org} or a polygon (triangle $\sim$ hexagon) corresponding to a triangle of G_{org}. The other properties are summarized below.

Claim 1 (Base graph G; restated as Lemma 4.1)**.** *Hereafter let G denote a* base graph*, a graph obtained by applying the vertex expansion on all vertices of the triangulated input graph G_{org} (under its combinatorial planar embedding). Let n denote the number of vertices of G (whereas we use n_{org} to denote the number of vertices of G_{org}). Then G satisfies the following: (i) it is three-regular, (ii) all faces of G are either a face obtained by expanding the corresponding vertex of G_{org} or a polygon* *(triangle $\sim$ hexagon) corresponding to a triangle of $G_{\rm org}$, (iii) $n = \Theta(n_{\rm org})$, and (iv) G has a planar embedding naturally defined from the one for $G_{\rm org}$.*

Gazit and Miller [10] considered a dual-like graph of an input graph; their separator algorithm was explained by using both an input graph and its dual-like graph. We follow this style, but here we consider the standard dual-graph. (See Figure 3 for an example of a dual-graph; it is easy to see from the figure that a dual-graph is triangulated if its base graph is three-regular.)



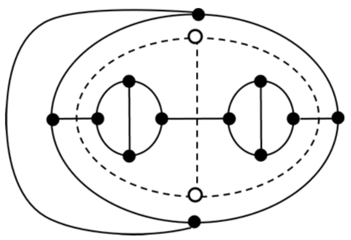
An example of a three-regular base graph (only its subgraph) and its dual-graph. The base graph is indicated by black nodes (for vertices) and solid lines (for edges), while its dual-graph is indicated by white nodes (for dual-vertices) and dashed lines (for dual-edges). As shown here a dual of a three-regular graph is triangulated.

Figure 3: A three-regular graph and its dual-graph

Definition 3.1 (Base dual-graph $\widetilde{G}$)**.** *Throughout this paper, let $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ denote a dual-graph of G (under its planar embedding) where $\widetilde{G}$ is the set of faces of G and $\widetilde{E}$ is the set of dual-edges, i.e., pairs of dual-vertices whose corresponding faces share at least one edge.*
Remark. *We can identify a dual-vertex of $\widetilde{G}$ with its corresponding face of G, and we sometimes say that a dual-vertex $\widetilde{v}$ has an edge e of G, meaning that e is one of the boundary edges of the face corresponding $\widetilde{v}$.*

Notational Remark. We will use a prefix "dual-" to denote a vertex, an edge, a path, and a cycle in $\widetilde{G}$ to distinguish them from those in G, and we will attach "$\tilde{}$" to variables denoting these objects; for example, a vertex of $\widetilde{G}$ is called a *dual-vertex* and it is denoted by, e.g., $\widetilde{v}$. (There are some exceptions. In case we add some other prefix such as "boss-", "frame-", etc. for referring an object of the dual-graph $\widetilde{G}$, we may omit adding "dual-." Also since subplanes and faces are notions involving a topological interpretation on $\mathbb{R}^2$, we will not use "dual-" to denote them even if they are defined w.r.t. the dual-graph $\widetilde{G}$.)

We remark here that the base dual-graph $\widetilde{G}$ may not be a simple graph though the base graph G is simple (by assuming the input graph is a simple graph); see Figure 4 for an example. We assume any reasonable way to identify non-simple dual-edges.



An example of the case where a non-simple dual-graph is created from a simple base graph. As the previous figure, i.e., Figure 3, a base graph is indicated by black nodes and solid lines, while some (not all) of dual-vertices and dual-edges are indicated by white nodes and dashed lines.

Figure 4: An example of the creation of a non-simple dual-graph

It is easy to see that the number $\widetilde{n}$ of dual-vertices of $\widetilde{G}$ is linearly related to n. Furthermore, we have the following relation.

Claim 2 (Restated as Lemma 4.2)**.** *Assume that n is large enough. If $\widetilde{G}$ has a $1/10$-separator of $\widetilde{G}$ of size $O(\sqrt{n})$, then G_{org} has a $1/31$-separator of size $O(\sqrt{n_{\mathrm{org}}})$.*

A frame-graph and four sub-steps of Step 2

A "frame-graph" is a weighted subgraph of the base dual-graph $\widetilde{G}$. Roughly speaking, we transform $\widetilde{G}$ by removing some of the dual-vertices, which creates new faces. The weight of these faces are defined proportional to the number of removed dual-vertices. In order to be able to use Miller's separator algorithm, we require the following conditions to a frame-graph.

Definition 3.2 (Frame-graph)**.** *A* frame-graph *is a weighted subgraph $\widetilde{H}$ of the base dual-graph $\widetilde{G}$ that satisfies the following conditions (we let $k = \sqrt{n}$ and we will fix this usage hereafter):*

(F1) *$\widetilde{H}$ is a weighted subgraph of $\widetilde{G}$ induced by some subset of dual-edges of $\widetilde{G}$. The weight of each face*[4] *f of $\widetilde{H}$ is proportional*[5] *to the number $\widetilde{n}_f$ of dual-vertices of $\widetilde{G}$ located in the face (which are removed from $\widetilde{G}$), and it is less than $1/3$.*

(F2) *$\widetilde{H}$ is 2-connected.*

(F3) *$\widetilde{H}$ contains $O(\widetilde{n}/k)$ faces.*

(F4) *The size of each face of $\widetilde{H}$ is $O(\sqrt{k})$; that is, for each face of $\widetilde{H}$, its boundary dual-cycle consists of $O(\sqrt{k})$ dual-vertices.*

Step 2 of our algorithm is to obtain a frame-graph $\widetilde{H}$ from the base dual-graph $\widetilde{G}$. Roughly speaking, we identify "frame-cycles", dual-cycles that define faces of $\widetilde{H}$, and we compose $\widetilde{H}$ as a collection of such dual-cycles. Step 2 is divided into the following four sub-steps. (Here we only specify the outputs of these sub-steps; some of the technical terms will be explained below. We may assume that the input to these sub-steps is all data prepared as output in Step 1 and the previous sub-steps.)

2.1. Identify Voronoi regions and compute their boss-vertices that are used as a root of a BFS dual-tree of each Voronoi region.
Output: The description of the Voronoi regions. That is, a set $\widetilde{I}$ of the boss-vertices, and for each $\widetilde{b} \in \widetilde{I}$, the description of a BFS dual-tree rooted at $\widetilde{b}$, and a set of boundary cycles for the corresponding Voronoi region.

2.2. Identify ridge edges (if needed), and compute connectors and branch vertices. Then identify "preliminary" frame-cycles[6].
Output: The set of branch vertices, connectors, and the description of preliminary frame-cycles.

2.3. Preprocess preliminary frame-cycles and compute frame-cycles, from which a frame-graph $\widetilde{H}$ is defined. (During this preprocessing step, there are cases where a desired separator of $\widetilde{G}$ is obtained and the algorithm terminates by transforming it to a desired $1/31$-separator of G_{org}. The algorithm of Lipton and Tarjan is used in one of such cases.)
Output: The description of $\widetilde{H}$, that is, its combinatorial planar embedding and its complete weighted face information.

2.4. Identify floor- and ceiling-cycles, and modify $\widetilde{H}$ by adding floor- and ceiling-cycles as new face boundary dual-cycles.
Output: The description of the modified frame-graph $\widetilde{H}$, that is, its combinatorial planar embedding and its complete face information.

We explain sub-steps a bit more in detail so that we can state important properties on their outputs. (Note that some notions are used below without a definition or with an only rough and not precise definition; see the corresponding section for their definitions. Though omitted below, we clearly need

[4]It is clear that $\widetilde{H}$ is a plane graph w.r.t. the embedding naturally inherited from $\widetilde{G}$. Thus, precisely speaking, we assume this embedding for discussing the faces of $\widetilde{H}$.

[5]It is simply $\widetilde{n}_f$ divided by the total number $\widetilde{n}^-$ of dual-vertices removed from $\widetilde{G}$ for defining $\widetilde{H}$. In fact, we will use, for simplicity, $\widetilde{n}_f$ instead of $\widetilde{n}_f/\widetilde{n}^-$ in our algorithm.

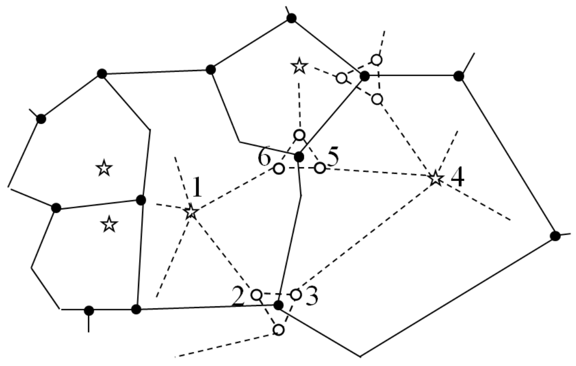
[6]In our technical discussion, we will define these preliminary frame-cycles formally and call them "pre-frame-cycles."

to show that these sub-steps can be done in $\widetilde{O}(\sqrt{n})$-space and polynomial-time, which will be proved in the subsequent technical sections.)

Step 2.1. (Corresponding technical section: Section 5)

We borrow the idea of Gazit and Miller [10] and introduce the notion of Voronoi region. A *Voronoi region* is defined by a set of connected dual-vertices within distance $2k$ from some dual-vertex, which we call a *boss-vertex*. Boss-vertices are selected in a greedy way, and it can be shown that we can cover $\widetilde{G}$ by selecting $\widetilde{n}/k$ boss-vertices. We use $\widetilde{I}$ to denote the set of all selected boss-vertices, and for each boss-vertex $\widetilde{b}$, we use $\mathrm{Vr}(\widetilde{b})$ to denote a Voronoi region having $\widetilde{b}$ as a boss-vertex. Precisely speaking, these Voronoi regions are subplanes dividing the plane where the base plane graph G is drawn. We can show that each Voronoi region is defined by boundary cycles consisting of G's edges; see Figure 5 for an example. The following properties are important.

Claim 3 (Voronoi region; follows from Lemmas 5.2 and 5.3)**.** *We have $|\widetilde{I}| \le \widetilde{n}/k$; that is, there are at most $\widetilde{n}/k$ Voronoi regions. Consider any $\widetilde{b} \in \widetilde{I}$ and its Voronoi region $\mathrm{Vr}(\widetilde{b})$. Its boundary is a set of cycles of G. For any dual-vertex $\widetilde{v} \in \mathrm{Vr}(\widetilde{b})$, there exists a dual-path from $\widetilde{b}$ to $\widetilde{v}$ of length $\le 2k$ consisting of only dual-vertices in $\mathrm{Vr}(\widetilde{b})$. Thus, $\mathrm{Vr}(\widetilde{b})$ has an $O(k)$-depth BFS dual-tree rooted at $\widetilde{b}$.*



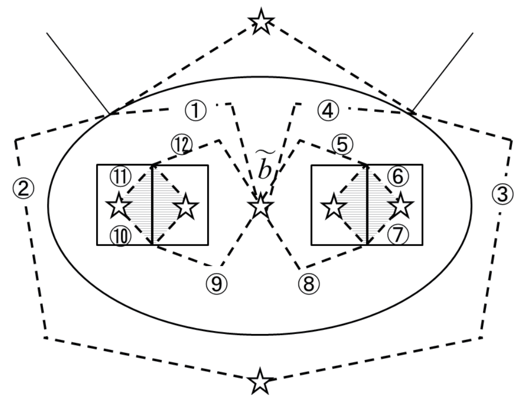
An example of Voronoi regions and dual-cycles that are bases for frame-cycles. Solid lines are the boundaries of Voronoi regions, and black nodes (which are called "branch vertices") are vertices of G that are incident to three Voronoi regions. White nodes are dual-vertices (whose corresponding faces are) incident to these black nodes. In particular, three dual-vertices incident to a branch vertex forms a dual-triangle, which we call "branch-triangle." The boss-vertex of each Voronoi region (which is also a dual-vertex) is indicated by a star. Six dual-vertices labeled with $1 \sim 6$ connected with dashed lines that indicate dual-paths (some of which is a dual-edge) form a dual-cycle; this may be regarded as a "frame-cycle." Note that this dual-cycle contains a part of Voronoi boundary shared by two Voronoi regions that connects two branch vertices.

Figure 5: Voronoi regions and an example of "frame-cycle"

See Figure 5. There are vertices where three Voronoi regions meet, which we call *branch vertices*, and a dual-triangle consisting of three dual-edges incident to a branch vertex is called a *branch-triangle*. As shown in the figure, we can define a dual-cycle in a pair of Voronoi regions sharing a boundary edge that consists of four dual-paths and two dual-edges of two branch-triangles. The dual-cycle following a tour of dual-vertices labeled with $1 \sim 6$ is an example. The four dual-paths are those in the BFS dual-trees of the corresponding Voronoi regions. This dual-cycle is a basis of a frame-cycle; in fact, during this outline explanation, we regard it as a preliminary frame-cycle. From the property of Voronoi regions, we may assume an $O(k)$-depth BFS dual-tree inside of such a preliminary frame-cycle.

Step 2.2. (Corresponding technical section: Section 7)

Dual-cycles like the one in Figure 5 can be used as frame-cycles for defining the target frame-graph $\widetilde{H}$. There is unfortunately a situation where some Voronoi region has more than one boundary cycle, and in this situation, we may have a more complicated dual-cycle as illustrated in Figure 6 whose size cannot be bounded appropriately.



An example of the case where there exists a Voronoi region with multiple Voronoi boundary cycles. Solid lines are Voronoi boundaries. We omit here showing vertices, dual-vertices, and branch-triangles except for boss-vertices. The outmost solid cycle is one of the boundary cycles of a Voronoi region $\mathrm{Vr}(\widetilde{b})$. This $\mathrm{Vr}(\widetilde{b})$ contains four Voronoi regions and it has two boundary cycles between them (i.e., two large squares consisting of two subsquares). Dashed lines are branch dual-paths. There are some "standard" dual-cycles that could be used as frame-cycles; for example, two dual-cycles indicated by dashed lines surrounding shadow parts have the same structure as the preliminary frame-cycle we saw in Figure 5. On the other hand, the graph has a dual-cycle consisting of twelve dual-paths labeled by numbers (where branch-triangles are omitted). The problem here is that we cannot bound the length of such a dual-cycle; thus, it cannot be used for a face boundary of a frame-graph as it is for satisfying the condition (F4).

Figure 6: A Voronoi region with multiple boundary cycles

This situation can be resolved by introducing additional branch vertices by which we can divide a large dual-cycle into small ones. Here again we borrow the idea from Gazit and Miller to select some of the edges of G as "ridge edges"; intuitively, by adding ridge edges to Voronoi boundary edges, some Voronoi boundary cycles are connected and new branch vertices are created. More formally, let B and R denote respectively the set of Voronoi boundary edges and that of ridge edges. Then we can show that $B \cup R$ forms a connected component in G. We regard a degree three vertex in $B \cup R$ as a *branch vertex* (whereas a branch vertex was previously a degree three vertex in B). By introducing new branch vertices, large dual-cycles are divided into small and "standard" dual-cycles, i.e., preliminary frame-cycles.

A path connecting a pair of branch vertices are called a *connector*. This is a generalization of the shared Voronoi boundary that appears in the frame-cycle of Figure 5. In fact, centering each connector, we can define a preliminary frame-cycle. The main task of this sub-step is to obtain all such preliminary frame-cycles of $\widetilde{G}$ from connectors. This one-to-one correspondence between connectors and preliminary frame-cycles is important for a later discussion.

Step 2.3. (Corresponding technical section: Section 8)

Some preliminary frame-cycles may not be appropriate for frame-cycles from the following reasons: (i) there may be a preliminary frame-cycle that defines a large subplane having too many dual-vertices of $\widetilde{G}$, which creates a face of weight more than $1/3$ in $\widetilde{H}$; (ii) there may be a preliminary frame-cycle that splits $\widetilde{G}$ into multiple disconnected subplanes, due to which $\widetilde{H}$ cannot be 2-connected by removing dual-vertices of $\widetilde{G}$ not participating to the frame-cycles. Thus, in this sub-step, we first preprocess the obtained preliminary frame-cycles to modify them to appropriate forms.

During this preprocessing step, we may have a situation where some preliminary frame-cycle itself is a desired separator of $\widetilde{G}$, say, $1/10$-separator of $\widetilde{G}$. Or we can apply the algorithm of Lipton-Tarjan to the subgraph that is inside of some preliminary frame-cycle to obtain a desired separator of $\widetilde{G}$. Note that the latter one is the case where some preliminary frame-cycle is large enough to cover many dual-vertices of $\widetilde{G}$. Since (the inside of) each preliminary frame-cycle is covered by (at most) two Voronoi regions, and each Voronoi region has an $O(k)$-depth BFS dual-tree, we can show by Proposition 2.3 that the algorithm of Lipton-Tarjan runs in $\widetilde{O}(k)$-space. Once a desired separator of $\widetilde{G}$ is obtained, the algorithm can terminate by yielding our target $1/31$-separator of G_{org} by Claim 2 (i.e., by the method stated in its proof).

After this preprocessing step (and a separator is not yet obtained), we now have *frame-cycles* and we can guarantee that each frame-cycle defines a single subplane that does not have so many dual-vertices of $\widetilde{G}$. Thus, we can define our frame-graph $\widetilde{H}$ as a collection of faces defined[7] by these frame-cycles. We should note, on the other hand, that the actual step of our algorithm is designed in a slightly different way. This step is designed to yield a frame-graph $\widetilde{H}$ as an induced dual-graph consisting of dual-edges of all frame-cycles. Thus, we need to prove the following claim. For its proof the one-to-one correspondence between connectors and (preliminary) frame-cycles plays an important role.

Claim 4 (Face of $\widetilde{H}$; restated as Lemmas 8.3 and 8.4)**.** *Each frame-cycle defines exactly one face in $\widetilde{H}$. Furthermore, there is no face in $\widetilde{H}$ other than ones defined by either a frame-cycle or a branch-triangle.*

Based on this claim, for a complete weighted face information of $\widetilde{H}$, the algorithm needs to output lists of frame-cycles, their weights, and their incidence relations. Note that the weight of each frame-cycle is simply the number of dual-vertices of $\widetilde{G}$ located its inside. From the preprocessing step, we can guarantee that it is less than $\widetilde{n}^{-}/3$, where $\widetilde{n}^{-}$ is the total number of dual-vertices removed from $\widetilde{G}$ to define $\widetilde{H}$. Thus, the obtained $\widetilde{H}$ satisfies the condition (F1).

We also prove that $\widetilde{H}$ satisfies (F2) and (F3).

Claim 5 (Restated as Lemmas 8.1)**.** *$\widetilde{H}$ is 2-connected.*

Claim 6 (Number of connectors and fame-cycles; restated as Lemma 8.5)**.** *The numbers of connectors is bounded by $O(\widetilde{n}/k)$. Hence, the number of faces of $\widetilde{H}$, that is, frame-cycles and branch-triangles is bounded by $O(\widetilde{n}/k)$.*

Step 2.4. (Corresponding technical section: Section 9)

Let us see again the frame-cycle illustrated in Figure 5. It consists of four BFS dual-paths and two branch-triangle edges. While we can bound the length of each BFS dual-path by $O(k)$, this bound is not enough for satisfying the condition (F4) that requires an $O(\sqrt{k})$ bound for the length of frame-cycles. We again borrow the idea of Gazit and Miller to introduce the notion of "floor-cycle" and "ceiling-cycle" to overcome this problem.

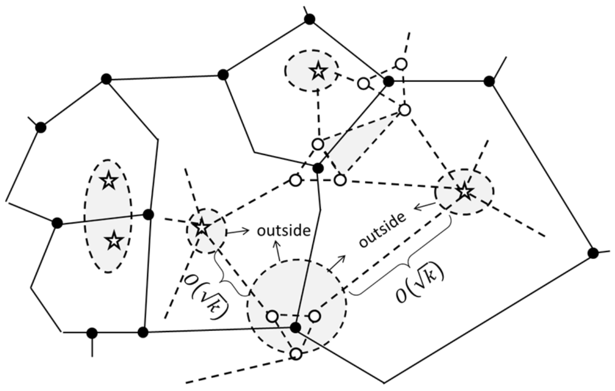
Floor- and ceiling-cycles are defined by using dual-vertices of the same level. A dual-vertex assigned a *level*, that is the distance from its boss-vertex (which is a temporal definition just for an explanation here). We consider connected components of dual-vertices of the same level; in particular, we are interested in connected components of size $\leq \sqrt{k}$. A *neck* of level ℓ is a connected component of level ℓ dual-vertices whose size is bounded by $\sqrt{k}$. In such a neck we can find (at least) two dual-cycles, one facing dual-vertices of level $\ell+1$ and another facing dual-vertices of level $\ell-1$. A *floor-cycle* is a dual-cycle of the former type that has enough number of dual-vertices (i.e., $2\widetilde{n}/3$ dual-vertices) in its "outside", the side the dual-cycle where level $\ell+1$ dual-vertices are located. Similarly, a *ceiling-cycle* is a dual-cycle of the latter type that has more than $2\widetilde{n}/3$ dual-vertices in its "outside", the side the dual-cycle where level $\ell-1$ dual-vertices are located. If multiple nested such dual-cycles exist, then we take the one with the largest level (resp., the smallest level) as the floor-cycle (resp., the ceiling-cycle).

Consider any frame-cycle $\widetilde{c}$ of $\widetilde{H}$. We can identify one floor-cycle containing its boss-vertex "inside." On the other hand, a ceiling-cycle (if it exits) is located around its branch-triangle containing some dual-vertices of the branch-triangle "inside" (Figure 7). By removing all dual-vertices inside of these floor- and ceiling-cycles (thereby creating new faces), we can reduce the length of each dual-path of the frame-cycle $\widetilde{c}$ remained "outside" of these new faces to $O(\sqrt{k})$.

Claim 7 (Reducing dual-path length of frame-cycles; restated as Lemma 9.8)**.** *Consider any frame-cycle $\widetilde{c}$, and let $\widetilde{p}$ be any dual-path between its boss-vertex and the next dual-vertex of a branch-triangle used in $\widetilde{c}$. There exist one floor-cycle and at most one ceiling-cycle crossing $\widetilde{p}$; let $\widetilde{p}'$ be the part of $\widetilde{p}$ located outside of these cycles. Then the length of $\widetilde{p}'$ is $O(\sqrt{k})$.*

In sub-step 2.4, for a given $\widetilde{G}$ (and $\widetilde{H}$) we identify floor- and ceiling-cycles and modify the frame-graph $\widetilde{H}$ by introducing new faces by removing all dual-vertices of $\widetilde{H}$ from the inside of these floor- and ceiling-cycles. We refer the obtained dual-graph $\widetilde{H}'$ as a *modified frame-graph.* The weight of its

[7]Precisely speaking, faces are defined also by branch-triangles, but in the following explanation, we sometimes omit mentioning branch-triangles for simplicity.



Cycles indicated by dashed lines surrounding shadow parts are floor- or ceiling-cycles. By the modification of $\widetilde{H}$ in sub-step 2.4, their "insides" (i.e., these shadow parts) are removed so that floor- and ceiling-cycles become are new face boundary dual-cycles; then the length of each dual-path of a frame-cycle from a boss-vertex to the next face boundary dual-cycle is reduced to $O(\sqrt{k})$.

Figure 7: Floor- and ceiling-cycles

faces is defined in the same way as $\widetilde{H}$; that is, the weight of each face is defined by the number of removed dual-vertices of $\widetilde{G}$ in the face. Then from the property of the original frame-graph $\widetilde{H}$ and by our definition of floor- and ceiling-cycles, we can show that the modified frame-graph $\widetilde{H}'$ satisfies the condition (F1) $\sim$ (F3). Furthermore, from the above claim (and since the length of each floor- and ceiling-cycle is $O(\sqrt{k})$ by definition), we have an $O(\sqrt{k})$ bound for the length of each face boundary dual-cycle. Therefore, the obtained modified frame-graph $\widetilde{H}'$ satisfies all the required conditions (F1) $\sim$ (F4). For simplifying our notation, we will use from now on $\widetilde{H}$ to denote this modified frame-graph.

Step 3 and the proof of our separator theorem

Now that a modified frame-graph $\widetilde{H}$ satisfying the conditions (F1) $\sim$ (F4) is obtained. Then the last step of our algorithm is to apply Miller's algorithm to obtain a weighted separator for $\widetilde{H}$ and compute our desired separators for $\widetilde{G}$ and $G_{\rm org}$.

Let us first prepare some parameters needed for the analysis. We use $\widetilde{n}_{\widetilde{H}}$ and $\widetilde{n}_{\widetilde{H}\rm face}$ to denote respectively the number of dual-vertices and faces of $\widetilde{H}$, and let d be its max. face size. Then from the conditions (F3) and (F4), we have $\widetilde{n}_{\widetilde{H}\rm face} = O(\widetilde{n}/k) = O(n^{1/2})$ and $d = O(\sqrt{k}) = O(n^{1/4})$ respectively; hence, $\widetilde{n}_{\widetilde{H}} \le d \times \widetilde{n}_{\widetilde{H}\rm face} = O(n^{3/4})$. Thus, the number of dual-vertices removed from $\widetilde{G}$ to obtain $\widetilde{H}$, which we denote by $\widetilde{n}^-$, is $\widetilde{n} - \widetilde{n}_{\widetilde{H}} = \Omega(n) - O(n^{3/4})$; we assume that n is large enough so that $\widetilde{n}^- \ge 3\widetilde{n}/4$ holds.

We consider the application of Miller's algorithm to $\widetilde{H}$. This creates a cycle 1/3-separator of $\widetilde{H}$. From the condition (F4) and Proposition 2.5, it follows that the size of this separator is $O(d\sqrt{\widetilde{n}_{\widetilde{H}}}) = O(\sqrt{n})$, and that Miller's algorithm can be executed in $\widetilde{O}(\widetilde{n}_{\widetilde{H}\rm face})$-space (= $\widetilde{O}(\sqrt{n})$-space). Then since the obtained separator is a dual-cycle, it can be used as a 1/4-separator of $\widetilde{G}$ as the following lemma claims. (*Cf.* A separator of $\widetilde{H}$ is not necessarily a separator of $\widetilde{G}$ in general.) Thus, in Step 3, we first apply Miller's algorithm to the frame graph $\widetilde{H}$ to obtain a weighted 1/3-separator (that is indeed a 1/4-separator of $\widetilde{G}$) in $\widetilde{O}(\sqrt{n})$-space, and then for the obtained separator for $\widetilde{G}$, we apply the method stated in the proof of Claim 2 to compute a desired 1/31-separator of $G_{\rm org}$ in $\widetilde{O}(1)$-space.

Lemma 3.1. *Any weighted dual-cycle* 1/3*-separator of a modified frame-graph* $\widetilde{H}$ *is a dual-cycle* 1/4*-separator of* $\widetilde{G}$.

Proof. Let $\widetilde{C}$ denote a weighted dual-cycle 1/3-separator of $\widetilde{H}$. Recall that $\widetilde{H}$ is a dual-edge induced subgraph of $\widetilde{G}$. Thus, any dual-cycle of $\widetilde{H}$ is also a dual-cycle of $\widetilde{G}$; thus, the dual-cycle separator $\widetilde{C}$ of $\widetilde{H}$ is a dual-cycle and hence a separator of $\widetilde{G}$. Recall that the weight of each face of $\widetilde{H}$ is the number of dual-vertices of $\widetilde{G}$ that are located in the face and that are removed when defining $\widetilde{H}$ normalized by the total number $\widetilde{n}^-$ of removed dual-vertices. Thus, since $\widetilde{C}$ is a weighted 1/3-separator, two subplanes separated by C have weight $\geq 1/3$; hence, the numbers of dual-vertices of these subplanes in $\widetilde{G}$ are at least $\widetilde{n}^{-}/3 \geq \widetilde{n}/4$ (as we checked before this lemma). Therefore, $\widetilde{C}$ is a 1/4-separator of $\widetilde{G}$. □

4 Base Graphs G and $\widetilde{G}$

We explain Step 1 of the algorithm, a preprocessing step for computing the base graphs G and $\widetilde{G}$ and supplementary information. Here we prove the key properties of these base graphs claimed in Outline section. We also give some basic algorithms for achieving this step.

We begin by defining the notion of "triangulation" formally.

Definition 4.1 (Triangulation)**.** *A plane graph is* triangulated *(w.r.t. its planar embedding) if addition of any edge results in a nonplanar graph. For a plane graph, its* triangulation *means to add edges to the plane graph until it gets triangulated w.r.t. this planar embedding.*

As explained in Outline section, we may assume, after the preprocessing sub-step of Step 1, that the input graph G_{org} is planar and connected and that the algorithm can use one of its combinatorial embeddings. Then in the rest of Step 1, the algorithm conducts the following tasks: (i) triangulating the input graph, (ii) applying the vertex expansion to this triangulated graph to obtain the base graph G, and (iii) computing its dual as the base dual-graph $\widetilde{G}$.

We explain the key properties of the base graphs. First one is for the base graph G.

Lemma 4.1 (Restatement of Claim 1)**.** *The following holds for G and G_{org}: (i) G is three-regular, (ii) all faces of G are either a face obtained by expanding the corresponding vertex of G_{org} or a polygon (triangle $\sim$ hexagon) corresponding to a triangle of G_{org}, (iii) $n = \Theta(n_{\text{org}})$, and (iv) it has a planar embedding naturally defined from the one for G_{org}.*

Proof. Properties (i) and (ii) are clear from the definition; see Figure 2 and the explanation there. Thus, we prove (iii) and (iv) here. Recall that the vertex expansion on v is to replace v with a $\deg_{\text{org}}(v)$-size face, by which we introduce $\deg_{\text{org}}(v)$ new vertices replacing v. Hence, by applying the vertex expansion to all vertices of $G_{\text{org}} = (V_{\text{org}}, E_{\text{org}})$, we create $\sum_{v\in V_{\text{org}}} \deg_{\text{org}}(v) = 2|E_{\text{org}}|$ new vertices, which is at least n_{org} and at most $6n_{\text{org}}$ since $|E_{\text{org}}| \leq 3n_{\text{org}}$ due to the planarity of G_{org} and the Euler's formula. This proves (iii). For showing (iv), consider any vertex $v \in V_{\text{org}}$. Let $e_0, \ldots, e_{d-1}$ be edges incident to v indexed in the order of the given combinatorial embedding of G_{org}. We may consider that the vertex expansion on v introduces a new vertex v'_i corresponding to each edge e_i, where v'_i has an edge e'_1 corresponding to e_i and two edges e'_0 and e'_2 connecting to $v'_{(i-1) \bmod d}$ and $v'_{(i+1) \bmod d}$. Then it is easy to see that the order e'_0, e'_1, e'_2 is a combinatorial embedding for v'_i under the embedding of G naturally inherited from G_{org}. □

Next consider the base dual-graph $\widetilde{G}$. Recall that it is defined a dual-graph of G (Definition 3.1) and it is triangulated thanks to the three-regularity of G (Figure 3). Here we show a way to get a separator for G_{org} from a separator for $\widetilde{G}$.

Lemma 4.2 (Restatement of Claim 2)**.** *If $\widetilde{G}$ has a 1/10-separator of $\widetilde{G}$ of size $O(\sqrt{n})$, then G_{org} has a* 1/31*-separator of size $O(\sqrt{n_{\text{org}}})$.*

Proof. First, we introduce a way to define a set of vertices of G (resp., G_{org}) that corresponds to a given dual vertex of $\widetilde{G}$. Consider any vertex $\widetilde{v}$ of $\widetilde{G}$. Note that $\widetilde{v}$ is a face in G; we define $\text{inv}(\widetilde{v})$ by a set of vertices of G that are incident to this face. Then define $\text{inv}_{\text{org}}(\widetilde{v})$ by a set of vertices whose vertex expansion intersects with $\text{inv}(\widetilde{v})$. For $\text{inv}_{\text{org}}(\widetilde{v})$ the following more direct interpretation would be helpful. Note that $\widetilde{v}$ is a face in G; on the other hand, the above lemma claims (as property (ii)) that every face of G is either a face obtained by expanding a vertex v of G_{org} or a polygon (triangle $\sim$ hexagon) face that corresponds to a triangle face $\{v_1, v_2, v_3\}$ of G_{org}. Hence, we have $\text{inv}_{\text{org}}(\widetilde{v}) = \{v\}$ for the former case and $\text{inv}_{\text{org}}(\widetilde{v}) = \{v_1, v_2, v_3\}$ for the latter case. For any set $\widetilde{W}$ of dual vertices of $\widetilde{G}$, we define $\text{inv}(\widetilde{W})$ and $\text{inv}_{\text{org}}(\widetilde{W})$ by $\text{inv}(\widetilde{W}) = \bigcup_{\widetilde{v}\in\widetilde{W}} \text{inv}(\widetilde{v})$ and $\text{inv}_{\text{org}}(\widetilde{W}) = \bigcup_{\widetilde{v}\in\widetilde{W}} \text{inv}_{\text{org}}(\widetilde{v})$, respectively.

Now for a given separator $\widetilde{S}$, we show that $S := \text{inv}_{\text{org}}(\widetilde{S})$ is a separator of G_{org} with a desired property. Note first that we have $|S| \leq 3|\widetilde{S}|$ from the above interpretation of inv_{org}. Thus, from the assumption, we have $|S| = O(\sqrt{n})$ $(= O(\sqrt{n_{\text{org}}}))$. Below we use $\widetilde{H}_1$ and $\widetilde{H}_2$ to denote sets of vertices of $\widetilde{G}$ separated by $\widetilde{S}$. Also for each $i \in \{1,2\}$, we let $H_i = \mathrm{inv}_{\mathrm{org}}(\widetilde{H}_i) \setminus S$. These symbols are used also to denote their corresponding induced graphs.

We show that S is indeed a separator of G separating H_1 and H_2. For this we consider $S' := \mathrm{inv}(\widetilde{S})$ and $H'_i := \mathrm{inv}(\widetilde{H}_i) \setminus S'$ for each $i \in \{1,2\}$, and show that H'_1 and H'_2 are separated by S'. Suppose otherwise, and assume that some $v'_1 \in H'_1$ and $v'_2 \in H'_2$ are connected after removing S'. Then there should be a path p' connecting v'_1 and v'_2 that has no vertex in S'. Consider a set P' of faces of G that share an edge with p'. Then a set of vertices of $\widetilde{G}$ corresponding the faces of P' forms a connected component in $\widetilde{G}$. This is due to the fact that any two dual-vertices of $\widetilde{G}$ corresponding faces of G incident to some common vertex of G must have an edge between them, thanks to the three-regularity of G. Also it is clear that P' has no dual-vertex in $\widetilde{S}$, and that there are two dual-vertices whose corresponding faces are incident to v'_1 and v'_2 that are in $\widetilde{H}_1$ and $\widetilde{H}_2$ respectively. This contradicts that $\widetilde{S}$ separates $\widetilde{H}_1$ and $\widetilde{H}_2$.

Next we show that S is a 1/31-separator. Without losing generality we may assume that $|H_1| \le |H_2|$, and here we show that $|H_1| \ge n/31$ for sufficiently large n.

Consider first vertices of $\widetilde{H}_1$ and $\widetilde{G}$. As explained above, each vertex of $\widetilde{G}$ corresponds to either a vertex of G_{org} or a triangle face of G_{org}; we call a vertex of $\widetilde{G}$ a *v-vertex* for the former case and a *f-vertex* for the latter case. The converse relation also holds; that is, each vertex of G_{org} and each triangle face of G_{org} corresponds to some vertex of $\widetilde{G}$. Thus, we have $\widetilde{n} = n_{\mathrm{org}} + F$, where $\widetilde{n}$ is the number vertices of $\widetilde{G}$ and F is the number of faces of G_{org}. Let V_1 and F_1 denote respectively the number of v-vertices and f-vertices of $\widetilde{H}_1$. Then we have from our assumption that

$$V_1 + F_1 \ \ge\ \widetilde{n}/10 \ \ge\ n_{\mathrm{org}}/10. \tag{1}$$

Now consider a graph H_1^+ induced by $\mathrm{inv}_{\mathrm{org}}(\widetilde{H}_1)$. (Recall that $H_1 = \mathrm{inv}_{\mathrm{org}}(\widetilde{H}_1) \setminus S$.) Let n_1^+, e_1^+, and f_1^+ denote the number of vertices, edges, and faces of H_1^+ respectively. We let $x = |H_1^+ \cap S|$. Then we have $n_1^+ = V_1 + x$, $f_1^+ \le F_1 + x$ and $2e_1^+ \ge 3F_1$. The two inequalities hold because H_1^+ might have faces that are f-vertices of $\widetilde{S}$ and the number of them is at most x. Then apply the Euler's formula for plane graphs to H_1^+, we have $n_1^+ = e_1^+ - f_1^+ + 2$, which derives $V_1 - 0.5F_1 \ge -2x + 2$. Hence, by using (1) we have

$$3V_1 \ \ge\ n_{\mathrm{org}}/10 - 4x + 4 \ \ge\ n_{\mathrm{org}}/10 - 4x.$$

Recall that $H_1 = H_1^+ \setminus S$. Thus, we have

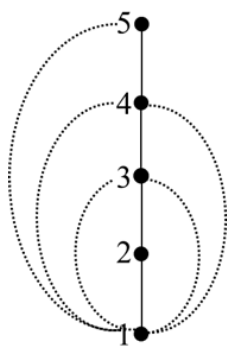
$$|H_1| \ \ge\ n_1^+ - |S| \ =\ V_1 + x - |S| \ \ge\ n_{\mathrm{org}}/30 - x/3 - |S|.$$

Then the desired bound holds for sufficiently large n_{org} since $x \le |S|$ and $|S| = O(\sqrt{n_{\mathrm{org}}})$. □

Algorithms for Step 1

We discuss algorithms for achieving Step 1. As explained in Outline section, thanks to the previous work of Reingold [15] and Allender and Mahajan [1], we have $O(\log n)$-space algorithms for checking the connectivity and planarity and for computing a combinatorial embedding of the input graph. Thus, our remaining algorithmic tasks are (i) triangulating the input graph G_{org}, (ii) computing a three-regular graph (i.e.., the base graph G) by applying the vertex-expansion to the triangulated graph, and (iii) computing the dual-base graph $\widetilde{G}$. (By "computing a graph" we formally mean to compute its embedding and its complete face information.) Here we explain how to achieve each task in $\widetilde{O}(1)$-space.

First consider the triangulation. The task of triangulation is essentially to introduce edges to separate non-triangle faces into triangles. Hence, for the triangulation task, we first need to identify each face, i.e., each traverse of G_{org}. Consider any directed edge (u,v) for any undirected edge $\{u,v\}$ of G_{org}, and consider the task of computing the left-traverse of G_{org} from (u,v). As we remarked in Preliminary section (Lemma 2.1), no same directed edge is identified during the left-traverse process (until coming back to (u,v)). Thus, for conducting the left-traverse process, we only need to remember (u,v) besides the directed edge currently identified in the process; hence, only some constant number of variables are needed for computing each left-traverse. For an obtained left-traverse $t = (v_1, \ldots, v_m)$, the triangulation of the face corresponding to the left-traverse is easy. Since the face corresponding t is the subspace located from (v_m, v_1) to (v_1, v_2) clockwise, we simply add edges $\{v_1, v_3\}$, …, $\{v_1, v_{m-1}\}$ for triangulating this face. For the combinatorial embedding π_{v_1} of vertices adjacent to v_1, we need to



An example of triangulating a face corresponding to a left-traverse. The original plane graph is a path whose edges are indicated by solid lines, while newly added edges are indicated by dotted lines. Here we consider a left-traverse $(1, 2, 3, 4, 5, 4, 3, 2)$, which defines a single face in the original graph. This face is divided into triangle faces by adding edges $\{1, 3\}$, $\{1, 4\}$, $\{1, 5\}$, $\{1, 4'\}$, and $\{1, 3'\}$. The combinatorial embedding π_1 for $\mathrm{N}_G(1)$ is modified from (2) to $(2, 3', 4', 5, 4, 3)$. We remark here that the graph becomes a simple graph again after the vertex expansion.

Figure 8: An example of triangulating a face

insert $v_3, \ldots, v_{m-1}$ between v_m and v_1 in the reverse order. Note that multiple edges may be created, which should be treated as different edges; see Figure 8.

Algorithmically, it is easier if we consider the triangulation in two sub-steps. First enumerate all left-traverses of G_{org}. Here for the second sub-step, we design the algorithm for this sub-step so that it does output a traverse $t = (v_1, \ldots)$ only if v_1 has the smallest index among all vertices in t. Then in the second sub-step, the algorithm modifies the combinatorial embedding π_v for each vertex v of G_{org}. For this, the algorithm searches through all left-traverses produced in the first sub-step for left-traverses in which v appears. In the case where v appears in a left-traverse t as the first vertex (say, $t = (v, v_2, \ldots, v_m)$), a sequence $v_{m-1}, \ldots, v_3$ is inserted between v_m and v_2 in π_v. On the other hand, in the case where v appears somewhere middle of t (say, $t = (v_1, \ldots, v_i, v, v_{i+2}, \ldots)$), v_1 is inserted between v_i and v_{i+2} in π_v. Note that each left-traverse causes the insertion in a different part of π_v. The algorithm produces the updated π_v's for all $v \in G_{\mathrm{org}}$, and then update $\mathrm{E}(G_{\mathrm{org}})$ by adding the introduced edges.

Now for the tasks (ii) and (iii), the algorithm first enumerates all triangle face boundary cycles of the triangulated G_{org}, which in fact can be done easily when introducing edges during the above triangulation. Once all triangle face boundary cycles are obtained, the rest of tasks (ii) and (iii) can be done easily by using some constant number of variables because local information is enough for the vertex expansion for obtaining the base graph G and also for computing the dual-graph $\widetilde{G}$. Note that the computation of face boundary dual-cycles of $\widetilde{G}$ is nothing but that of left-traverses of $\widetilde{G}$.

5 Voronoi Region

In this section, we formally define the notion of Voronoi region and explain the properties of Voronoi regions. We then explain Step 2.1 of our algorithm for computing Voronoi regions.

We start with introducing the notion of "region", an important notion throughout this paper.

Definition 5.1 (Region and boundary). *A* region *is a set R of dual-vertices of $\widetilde{G}$ such that any two dual-vertices of R is connected by some dual-path consisting of only dual-vertices of R. In other words, R is a region if $\widetilde{G}[R]$, the subgraph of $\widetilde{G}$ induced by R, is connected. We also consider region R as a set of faces of G, and a set of edges defined by*

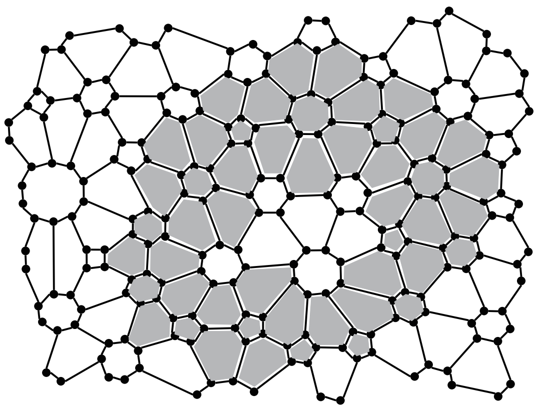
$$\{\, e \in E \mid \text{there exists exactly one face in } R \text{ that contains } e \,\}$$

is the boundary *of R.*

Remark. *Though a region is defined as a set of dual-vertices of $\widetilde{G}$, (formally speaking) it should be regarded as a subplane in G defined by the corresponding set of faces of G. Thus, we will use symbols without $\widetilde{}$ for regions.*

We recall the following important fact from [13] that is provable for three-regular graphs.

Proposition 5.1. *Any region of a three-regular graph (e.g., our base graph G) has a boundary consisting of simple cycles.*



An example of a region and its boundary. Here black nodes and solid lines are those in G; each face is regarded as a dual-vertex of $\widetilde{G}$. The shaded part is a region. The boundary of this region consists of two cycles. In general Miller showed that in three-regular graph, the boundary of any region is a set of simple cycles.

Figure 9: A region and its boundary

We introduce the notion of "Voronoi region", a specific family of regions. For any dual-vertices $\widetilde{u}$ and $\widetilde{v}$, by $\mathrm{dist}(\widetilde{u}, \widetilde{v})$ we mean the distance between $\widetilde{u}$ and $\widetilde{v}$, that is, the length of a shortest dual-path between $\widetilde{u}$ and $\widetilde{v}$. Consider any dual-vertex $\widetilde{v}$. Technically, it would be easier if we can identify the "nearest" dual-vertex to $\widetilde{v}$; for this purpose, we introduce a total order (denoted by $<_{\widetilde{v}}$) in $\widetilde{V}$ based on the distance from $\widetilde{v}$. For any dual-vertices $\widetilde{u}$ and $\widetilde{w}$, we say that $\widetilde{u}$ is *nearer* to $\widetilde{v}$ than $\widetilde{w}$ (written as $\widetilde{u} <_{\widetilde{v}} \widetilde{w}$) if we have either

- $\mathrm{dist}(\widetilde{u}, \widetilde{v}) < \mathrm{dist}(\widetilde{w}, \widetilde{v})$, or
- $\mathrm{dist}(\widetilde{u}, \widetilde{v}) = \mathrm{dist}(\widetilde{w}, \widetilde{v})$, and $\widetilde{u}$ has a smaller index[8] than $\widetilde{w}$.

For any dual-vertex $\widetilde{v}$ and any integer $d \geq 0$, let $\mathrm{L}(\widetilde{v}, d)$ denote the set of dual-vertices whose distance from $\widetilde{v}$ is d. We introduce two types of neighborhoods. (*Remark.* Since $\mathrm{L}(\widetilde{v}, d)$'s and the neighborhoods defined below are also regions, we do not use $\widetilde{\ }$ for their symbols. Recall that k is the parameter that is set $\sqrt{n}$ throughout this paper.)

Definition 5.2 (k-neighborhood)**.** *For any dual-vertex $\widetilde{v}$, let $\mathrm{d_{nb}}(\widetilde{v})$ be the largest integer d such that $|\cup_{0\leq i\leq d} \mathrm{L}(\widetilde{v}, i)| < k$ holds. Then we define $\mathrm{N}_k(\widetilde{v})$ and $\mathrm{N}_k^+(\widetilde{v})$ by*

$$\mathrm{N}_k(\widetilde{v}) \;=\; \bigcup_{0\leq i\leq d_0} \mathrm{L}(\widetilde{v}, i), \quad \text{and} \quad \mathrm{N}_k^+(\widetilde{v}) \;=\; \bigcup_{0\leq i\leq d_0+1} \mathrm{L}(\widetilde{v}, i),$$

*where $d_0 = \mathrm{d_{nb}}(\widetilde{v})$. We call $\mathrm{N}_k(\widetilde{v})$ and $\mathrm{N}_k^+(\widetilde{v})$ a k-*neighborhood *and a k-*neighborhood$^+$ *respectively.*
Remark. *By $\mathrm{N}_k(\widetilde{v})$ and $\mathrm{N}_k^+(\widetilde{v})$ we also mean subgraphs of $\widetilde{G}$ induced by sets $\mathrm{N}_k(\widetilde{v})$ and $\mathrm{N}_k^+(\widetilde{v})$.*

The following fact is immediate from the above definition.

Lemma 5.1. *For any dual-vertex $\widetilde{v}$, we have the following: (i) $\mathrm{N}_k(\widetilde{v}) \subseteq \mathrm{N}_k^+(\widetilde{v})$, (ii) both $\mathrm{N}_k(\widetilde{v})$ and $\mathrm{N}_k^+(\widetilde{v})$ are regions, (iii) $|\mathrm{N}_k(\widetilde{v})| < k$ and $|\mathrm{N}_k^+(\widetilde{v})| \geq k$, and (iv) the distance between $\widetilde{v}$ and any vertex in $\mathrm{N}_k^+(\widetilde{v})$ is at most k.*

We select representative k-neighborhoods, which will be used for defining Voronoi regions.

Definition 5.3. *A subset $\widetilde{I}$ of $\widetilde{V}$ is a k-*maximal independent set *if it satisfies following conditions:*

- *For any two dual-vertices $\widetilde{b}_1, \widetilde{b}_2 \in \widetilde{I}$, $\mathrm{N}_k^+(\widetilde{b}_1) \cap \mathrm{N}_k^+(\widetilde{b}_2) = \emptyset$.*
- *For any dual-vertex $\widetilde{v} \notin \widetilde{I}$, there exists a dual-vertex $\widetilde{b} \in \widetilde{I}$ such that $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}) \neq \emptyset$.*

The following properties are immediate from the definition.

Lemma 5.2. *Consider any k-maximal independent set $\widetilde{I}$ of $\widetilde{G}$. Then we have (i) $|\widetilde{I}| \leq \widetilde{n}/k$, and (ii) for any dual-vertex $\widetilde{v} \in \widetilde{V}$, we have some dual-vertex $\widetilde{b} \in \widetilde{I}$ and some dual-vertex $\widetilde{u}$ such that $\widetilde{u} \in \mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b})$, and we have $\mathrm{dist}(\widetilde{v}, \widetilde{b}) \leq 2k$.*

[8] We assume that dual-vertices are indexed in a certain way.

Notational Remark. (The set $\widetilde{I}$)
Though we explain our algorithm design for Voronoi regions later, we briefly mention that a standard greedy method can be used to design an $\widetilde{O}(\sqrt{n})$-space and polynomial-time algorithm that computes a k-maximal independent set. From now on, let us use $\widetilde{I}$ to denote the k-maximal independent set obtained by this algorithm for the base dual-graph $\widetilde{G}$.

Now we define Voronoi regions w.r.t. the above defined $\widetilde{I}$. Intuitively, for each $\widetilde{b} \in \widetilde{I}$, we would like to define the "Voronoi region" $\mathrm{Vr}(\widetilde{b})$ of $\widetilde{b}$ by a set of dual-vertices for which $\widetilde{b}$ is closest among all dual-vertices in $\widetilde{I}$. But such a dual-vertex in $\widetilde{I}$ that is closest to $\widetilde{v}$ may not be computable within our desired space bound. Thus, we use "closest k-neighborhood" to determine our Voronoi regions, by which we can identify Voronoi regions in $O(k + \widetilde{n}/k)$-space.

Definition 5.4 (Voronoi region and boss-vertex)**.** *For any set W of dual-vertices, $\mathrm{nrst}_{\widetilde{v}}(W)$ denotes a dual-vertex $\widetilde{u}$ in W such that $\widetilde{u} <_{\widetilde{v}} \widetilde{w}$ of all $\widetilde{w} \in W \setminus \{\widetilde{u}\}$. For any sets W and W' of dual-vertices, we write $W <_{\widetilde{v}} W'$ if $\mathrm{nrst}_{\widetilde{v}}(W) <_{\widetilde{v}} \mathrm{nrst}_{\widetilde{v}}(W')$. For any dual-vertex $\widetilde{v}$, the* boss-vertex *of $\widetilde{v}$ (denoted by $\mathrm{boss}(\widetilde{v})$) is a dual-vertex $\widetilde{b} \in \widetilde{I}$ such that $\mathrm{N}_k^+(\widetilde{b}) <_{\widetilde{v}} \mathrm{N}_k^+(\widetilde{b}')$ for any $\widetilde{b}' \in \widetilde{I} \setminus \{\widetilde{b}\}$. The* Voronoi region *of $\widetilde{b}$ (denoted by $\mathrm{Vr}(\widetilde{b})$) is a set of dual-vertices $\widetilde{v}$ such that $\mathrm{boss}(\widetilde{v}) = \widetilde{b}$.*
Remark. *In a nutshell, $\widetilde{b}$ is the boss-vertex of $\widetilde{v}$ if and only if its k-neighborhood is nearest to $\widetilde{v}$ among all dual-vertices in $\widetilde{I}$. It may be the case that there is some other dual-vertex in $\widetilde{I}$ that is nearer to $\widetilde{v}$.*

By definition, for every dual-vertex $\widetilde{v} \in \widetilde{V}$, there exists a unique Voronoi region $\mathrm{Vr}(\widetilde{b})$ that contains $\widetilde{v}$. Furthermore, for any Voronoi region $\mathrm{Vr}(\widetilde{b})$, we show below that it is indeed a region and that every dual-vertex in $\mathrm{Vr}(\widetilde{b})$ is reachable by a dual-path in $\mathrm{Vr}(\widetilde{b})$ of length $\leq 4k$.

Lemma 5.3. *For any dual-vertex $\widetilde{b} \in \widetilde{I}$ and any dual-vertex $\widetilde{v} \in \mathrm{Vr}(\widetilde{b})$, there exists a dual-path from $\widetilde{b}$ to $\widetilde{v}$ of length $\leq 2k$ consisting of only dual-vertices in $\mathrm{Vr}(\widetilde{b})$. (Hence, any pair of dual-vertices in $\mathrm{Vr}(\widetilde{b})$ is connected by a dual-path in $\mathrm{Vr}(\widetilde{b})$ of length $\leq 4k$.)*

Proof. Consider any dual-vertex $\widetilde{v} \in \mathrm{Vr}(\widetilde{b})$. By Lemma 5.2 (ii), there exists some $\widetilde{b}' \in \widetilde{I}$ such that $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}') \neq \emptyset$ holds. Clearly, $\widetilde{b}$ also satisfies $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}) \neq \emptyset$ because otherwise $\widetilde{b}$ cannot be the boss-vertex of $\widetilde{v}$. Thus, there is a dual-path from $\widetilde{v}$ to $\widetilde{b}$ of length $\leq 2k$.

For the lemma, it suffices to show that there exists a dual-path of length $\leq 2k$ from $\widetilde{v}$ to $\widetilde{b}$ consisting of only vertices in $\mathrm{Vr}(\widetilde{b})$. For this, consider $\widetilde{u} := \mathrm{nrst}_{\widetilde{v}}(\mathrm{N}_k^+(\widetilde{b}))$, i.e., the dual-vertex in $\mathrm{N}_k^+(\widetilde{b})$ nearest to $\widetilde{v}$, which in fact is the dual-vertex in the whole $\cup_{\widetilde{b}' \in \widetilde{I}} \mathrm{N}_k^+(\widetilde{b}')$ nearest to $\widetilde{v}$ witnessing that $\widetilde{v} \in \mathrm{Vr}(\widetilde{b})$. Clearly, any shortest dual-path from $\widetilde{u}$ to $\widetilde{b}$ belongs to $\mathrm{N}_k^+(\widetilde{b})$ and its length is at most k. (It is easy to see that $\mathrm{N}_k^+(\widetilde{b}) \subseteq \mathrm{Vr}(\widetilde{b})$.) Consider any shortest dual-path $\widetilde{v}$ to $\widetilde{u}$. Again it is clear that its length is at most k. We show that all dual-vertices on the dual-path belong to $\mathrm{Vr}(\widetilde{b})$. Suppose otherwise; that is, there exists some dual-vertex $\widetilde{v}'$ on the dual-path that belongs to the other $\mathrm{Vr}(\widetilde{b}')$. Then for this $\widetilde{v}'$, there exists some $\widetilde{u}' \in \mathrm{Vr}(\widetilde{b}')$ that is nearer to $\widetilde{v}'$ than $\widetilde{u}$. But since $\widetilde{v}'$ is on one of the shortest dual-paths from $\widetilde{v}$ to $\widetilde{u}$, this means that $\widetilde{u}'$ is also nearer to $\widetilde{v}$ than $\widetilde{u}$, a contradiction. Therefore the lemma follows. ☐

The lemma shows that each Voronoi region $\mathrm{Vr}(\widetilde{b})$ has a spanning BFS dual-tree on (the subgraph induced by) the Voronoi region with $\widetilde{b}$ as a root and that the depth of such trees is at most $2k$. Furthermore, it follows from Lemma 5.2, the number of Voronoi regions (i.e., the size of $\widetilde{I}$) is bounded by $\widetilde{n}/k$. This proves the properties stated in Claim 3. Before showing the algorithms for Step 2.1, we show one more lemma.

Lemma 5.4. *For any dual-vertex $\widetilde{b} \in \widetilde{I}$ and any dual-vertex $\widetilde{v} \in \mathrm{Vr}(\widetilde{b})$, there exists an algorithm that computes $\mathrm{dist}(\widetilde{v}, \widetilde{b})$ in $\widetilde{O}(k)$-space.*

Proof. For any dual-vertex $\widetilde{u}$, we can enumerate all dual-vertices in $\mathrm{N}_k^+(\widetilde{u})$ in $\widetilde{O}(k)$-space (the details will be described in the Algorithms for Step 2.1). Thus, if $\widetilde{v}$ is in $\mathrm{N}_k^+(\widetilde{b})$, we can compute $\mathrm{dist}(\widetilde{v}, \widetilde{b})$ in $\widetilde{O}(k)$-space. Assume $\widetilde{v}$ does not belong to $\mathrm{N}_k^+(\widetilde{b})$. In this case, $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}) \neq \emptyset$ holds, and let $\widetilde{u}$ be a vertex in $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b})$. We suppose that there is a dual-vertex $\widetilde{w} \notin \mathrm{N}_k^+(\widetilde{v}) \cup \mathrm{N}_k^+(\widetilde{b})$ in one of the shortest dual-paths between $\widetilde{v}$ and $\widetilde{b}$. The distances $\mathrm{dist}(\widetilde{v}, \widetilde{u})$ and $\mathrm{dist}(\widetilde{b}, \widetilde{u})$ are at most $\mathrm{d_{nb}}(\widetilde{v}) + 1$ and $\mathrm{d_{nb}}(\widetilde{b}) + 1$ respectively, and the shortest dual-paths are clearly included in their k-neighborhoods$^+$.

Since $\widetilde{w} \notin \mathrm{N}_k^+(\widetilde{v}) \cup \mathrm{N}_k^+(\widetilde{b})$, the distances $\mathrm{dist}(\widetilde{v}, \widetilde{w})$ and $\mathrm{dist}(\widetilde{b}, \widetilde{w})$ are more than $\mathrm{d}_{\mathrm{nb}}(\widetilde{v})+1$ and $\mathrm{d}_{\mathrm{nb}}(\widetilde{b})+1$ respectively. Thus $\widetilde{w}$ cannot become a dual-vertex in the shortest dual-path, a contradiction. Therefore, we can compute $\mathrm{dist}(\widetilde{v}, \widetilde{b})$ by calculating $\mathrm{dist}(\widetilde{v}, \widetilde{u}) + \mathrm{dist}(\widetilde{b}, \widetilde{u})$ for any $\widetilde{u} \in \mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b})$ and taking their minimum. □

Algorithms for Step 2.1

We discuss algorithms for achieving Step 2.1. For algorithms we show below, it is rather easy to see that they run within polynomial-time in n; thus, we omit explaining their time complexity. We also explain algorithms as if one algorithm can pass a large amount of data to another algorithm; but as explained in Preliminaries section this process can be realized without using a space for keeping such data.

The task of Step 2.1 is to compute information on the Voronoi regions of G. More specifically, we consider an algorithm that outputs

(a) the set $\widetilde{I}$ of boss-vertices;

(b) a family $\{\mathrm{N}_k(\widetilde{b})\}_{\widetilde{b} \in \widetilde{I}}$ of k-neighborhoods;

(c) a list of $(\widetilde{v}, \mathrm{boss}(\widetilde{v}), \widetilde{v}_{\mathrm{pre}})$ for all $\widetilde{v} \in \widetilde{V}$, where $\mathrm{boss}(\widetilde{v})$ is the boss-vertex of the Voronoi region that $\widetilde{v}$ belongs to and $\widetilde{v}_{\mathrm{pre}}$ is the dual-vertex that is a parent in a BFS dual-tree of $\mathrm{Vr}(\mathrm{boss}(\widetilde{v}))$; and

(d) a list of boundary cycles of all Voronoi regions and lists of pairs of Voronoi regions sharing a vertex (resp., an edge).

A key algorithmic tool we use here is a BFS algorithm traversing at most t dual-vertices from a given source dual-vertex $\widetilde{v}$. In this paper we consider specific BFS that is convenient for designing space efficient algorithms. We define a tree T satisfying the following condition as a *BFS tree* (from a source dual-vertex $\widetilde{v}$): $\widetilde{w}$ is a child of $\widetilde{u}$ in T if and only if $\widetilde{u}$ has the smallest index among dual-vertices that are adjacent to $\widetilde{w}$ and satisfies $\mathrm{dist}(\widetilde{v}, \widetilde{w}) - \mathrm{dist}(\widetilde{v}, \widetilde{u}) = 1$. For computing this BFS tree, we keep all vertices having the same level, namely the same distance from the root, and collect the vertices in the next level by processing smaller indexed vertex earlier; *Cf.* In a standard BFS algorithm, we process vertices in the order added to a queue regardless of the indices. It is easy to see that the algorithm runs in $\widetilde{O}(t)$-space. Then we have an $\widetilde{O}(k)$-space algorithm that enumerates, for a given $\widetilde{v}$, all elements of $\mathrm{N}_k(\widetilde{v})$, the set of the first k closest dual-vertices to $\widetilde{v}$ in $\widetilde{G}$. We may assume that the distance to $\widetilde{v}$ is also computed and kept for all enumerated dual-vertices. Recall that $\mathrm{d}_{\mathrm{nb}}(\widetilde{v}) = \max_{\widetilde{u} \in \mathrm{N}_k(\widetilde{v})} \mathrm{dist}(\widetilde{u}, \widetilde{v})$. By using this algorithm, we can design an algorithm that decides whether a given dual-vertex $\widetilde{u}$ is in $\mathrm{N}_k^+(\widetilde{v})$, where $\mathrm{N}_k^+(\widetilde{v})$ contains additionally all dual-vertices whose distance from $\widetilde{v}$ is $\mathrm{d}_{\mathrm{nb}}(\widetilde{v}) + 1$. Though we cannot keep $\mathrm{N}_k^+(\widetilde{v})$ since it could become very large, we can determine whether $\widetilde{u} \in \mathrm{N}_k^+(\widetilde{v})$ by checking whether it is adjacent to some $\widetilde{w} \in \mathrm{N}_k(\widetilde{v})$ whose distance from $\widetilde{v}$ is $\le \mathrm{d}_{\mathrm{nb}}(\widetilde{v})$. Then by using this algorithm, we can design an $\widetilde{O}(k)$-space algorithm that determines, for given $\widetilde{v}$ and $\widetilde{b}$, whether $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}) \neq \emptyset$; it simply search in $\widetilde{V}$ a dual-vertex that belongs to both $\mathrm{N}_k^+(\widetilde{v})$ and $\mathrm{N}_k^+(\widetilde{b})$. We may assume that the algorithm can also identify the dual-vertex in $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}) \neq \emptyset$ that is nearest to $\widetilde{v}$, namely, $\mathrm{nrst}_{\widetilde{v}}(\mathrm{N}_k^+(\widetilde{b}))$, and compute its distance to $\widetilde{v}$.

Armed with these algorithmics we can compute (a) $\sim$ (d) in $\widetilde{O}(k+\widetilde{n}/k)$-space. First, for computing $\widetilde{I}$ as the output (a), we simply collect dual-vertices to $\widetilde{I}$ in a greedy way until there is no dual-vertex $\widetilde{v}$ such that $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}) = \emptyset$ for all so far obtained boss-vertices $\widetilde{b}$. Note that the working memory for keeping the obtained boss-vertices is needed here to achieve the greedy algorithm. Hence, we need $\widetilde{O}(k+\widetilde{n}/k)$-space since we can bound $|\widetilde{I}| \le \widetilde{n}/k$. We may assume that $\mathrm{N}_k(\widetilde{b})$ is computed (and produced as the output (b)) every time a new boss-vertex $\widetilde{b}$ is obtained.

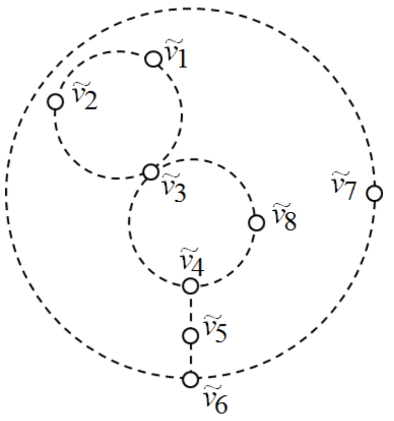
Next let us fix any dual-vertex $\widetilde{v}$, and consider how to compute $(\widetilde{v}, \mathrm{boss}(\widetilde{v}), \widetilde{v}_{\mathrm{pre}})$ as the output (c). For computing $\widetilde{b}_{\widetilde{v}} := \mathrm{boss}(\widetilde{v})$, we enumerate all $\widetilde{b} \in \widetilde{I}$ such that $\mathrm{N}_k^+(\widetilde{v}) \cap \mathrm{N}_k^+(\widetilde{b}) \neq \emptyset$; then $\widetilde{b}_{\widetilde{v}}$ is obtained as $\widetilde{b}$ such that $\mathrm{nrst}_{\widetilde{v}}(\mathrm{N}_k^+(\widetilde{b}))$ is the nearest to $\widetilde{v}$. According to our BFS tree definition, $\widetilde{v}_{\mathrm{pre}}$ needs to satisfy that it is adjacent to $\widetilde{v}$ and $\mathrm{dist}(\widetilde{b}, \widetilde{v}) - \mathrm{dist}(\widetilde{b}, \widetilde{v}_{\mathrm{pre}}) = 1$ holds. By Lemma 5.4, we can enumerate all such dual-vertices in $\widetilde{O}(k)$-space, and $\widetilde{v}_{\mathrm{pre}}$ is the dual-vertex having the smallest index among the candidates.

Finally, consider the computation for (d), that is, a list of boundary cycles of all Voronoi regions and lists of their incidence and edge-incidence relations. Consider any Voronoi region $\mathrm{Vr}(\widetilde{b})$ (specified its boss-vertex $\widetilde{b}$). Since it is easy to determine, for a given edge of G, whether it is a boundary edge of $\mathrm{Vr}(\widetilde{b})$, we can enumerate all boundary edges in $\widetilde{O}(1)$-space. Then since the boundary of $\mathrm{Vr}(\widetilde{b})$ is a collection of cycles of G, we can simply use the $\widetilde{O}(1)$-space algorithm for computing a left-traverse for identifying all boundary cycles of $\mathrm{Vr}(\widetilde{b})$ from the set of its boundary edges. Thus, a list of boundary cycles of all Voronoi regions and lists of pairs of Voronoi regions sharing a vertex (resp., an edge) are $\widetilde{O}(1)$-space computable.

6 Multiple-Dual-Cycle (m.d.-cycle)

In this section we introduce the notion of "multiple dual-cycle with an orientation" (in short, m.d.-cycle), which will be used to discuss faces defined by dual-cycles. This section is a preliminary section for our later discussion, and it does not correspond to any algorithmic step.

An m.d.-cycle $\widetilde{c}$ is a sequence of dual-vertices connected with dual-edges (with a direction implied by the sequence) that defines nonoverlapping subplane(s) under the assumed planar embedding. Intuitively, an m.d.-cycle is simply a collection of incident dual-cycles and paths; see Figure 10 for examples. Formally we have the following definition.



This figure shows examples of m.d.-cycles. White nodes are dual-vertices that are connected by dual-edges indicated by dashed lines. We may specify an m.d.-cycle by $(\widetilde{v}_1,\widetilde{v}_2,\widetilde{v}_3,\widetilde{v}_4,\widetilde{v}_5,\widetilde{v}_6,\widetilde{v}_7,\widetilde{v}_6,\widetilde{v}_5,\widetilde{v}_4,\widetilde{v}_8,\widetilde{v}_3,\widetilde{v}_1)$, which traverses a set of directed dual-edges under the anti-clockwise order (i.e., turning always right). Similarly, $(\widetilde{v}_1,\widetilde{v}_3,\widetilde{v}_8,\widetilde{v}_4,\widetilde{v}_5,\widetilde{v}_6,\widetilde{v}_7,\widetilde{v}_6,\widetilde{v}_5,\widetilde{v}_4,\widetilde{v}_3,\widetilde{v}_2,\widetilde{v}_1)$ is a m.d.-cycle traversing the same set of dual-vertices (with a different set of directed dual-edges, though) under the clockwise order (i.e., turning always left). Note that non-duplicate dual-edges form dual-cycles, and duplicate dual-edges form a dual-path.

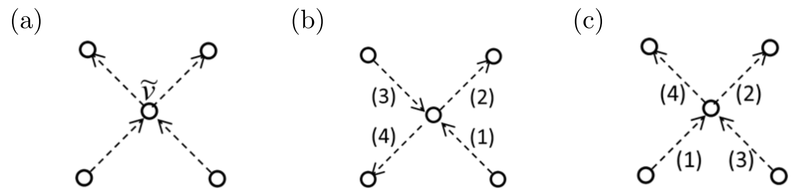
An example of the cases that one needs to be careful is the difference between sequences $(\widetilde{v}_1,\widetilde{v}_2,\widetilde{v}_3,\widetilde{v}_8,\widetilde{v}_4,\widetilde{v}_5,\widetilde{v}_6,\widetilde{v}_7,\widetilde{v}_6,\widetilde{v}_5,\widetilde{v}_4,\widetilde{v}_3,\widetilde{v}_2,\widetilde{v}_1)$ and $(\widetilde{v}_1,\widetilde{v}_2,\widetilde{v}_3,\widetilde{v}_8,\widetilde{v}_4,\widetilde{v}_5,\widetilde{v}_6,\widetilde{v}_7,\widetilde{v}_6,\widetilde{v}_5,\widetilde{v}_4,\widetilde{v}_3,\widetilde{v}_1)$; the difference is only the last three (or two) dual-vertices. Both sequences traverse all dual-vertices, but the latter one is not an m.d.-cycle while the former one is an m.d.-cycle. The difference is the usage of four directed dual-edges at $\widetilde{v}_3$.

Figure 10: M.d.-cycles

Definition 6.1 (m.d.-cycle)**.** *An* m.d.-cycle $\widetilde{c}$ *(under the clockwise/anti-clockwise order) is a sequence* $(\widetilde{v}_1,\widetilde{v}_2)$, $(\widetilde{v}_2,\widetilde{v}_3),\ldots,(\widetilde{v}_{m-1},\widetilde{v}_m)$ *of directed dual-edges satisfying the following conditions: (i)* $\widetilde{v}_1=\widetilde{v}_m$*, that is, it starts and ends with the same dual-vertex; (ii) every directed dual-edge* $(\widetilde{v}_i,\widetilde{v}_{i+1})$ *of* $\widetilde{c}$ *is based on a dual-edge* $\{\widetilde{v}_i,\widetilde{v}_{i+1}\}$ *of* $\widetilde{G}$*; (iii) no directed dual-edge appears more than once in* $\widetilde{c}$ *(while it is possible that some directed dual-edge and its reverse may both appear in* $\widetilde{c}$*); and (iv) for every pair* $(\widetilde{v}_{i-1},\widetilde{v}_i)$ *and* $(\widetilde{v}_i,\widetilde{v}_{i+1})$ *of consecutive directed dual-edges,* $(\widetilde{v}_i,\widetilde{v}_{i+1})$ *must be the clockwise (resp., anti-clockwise) next directed dual-edge from* $\widetilde{v}_i$ *appearing in* $\widetilde{c}$*. (We regard the reverse directed dual-edge* $(\widetilde{v},\widetilde{u})$ *of* $(\widetilde{u},\widetilde{v})$ *the furthest dual-edge from* $(\widetilde{u},\widetilde{v})$ *both clockwise and anti-clockwise.)*

Remark. *We use a sequence* $(\widetilde{v}_1,\widetilde{v}_2,\ldots,\widetilde{v}_m)$ *of dual-vertices for specifying a m.d.-cycle* $\widetilde{c}$*, by which we mean a sequence* $(\widetilde{v}_1,\widetilde{v}_2),\ldots,(\widetilde{v}_{m-1},\widetilde{v}_m)$ *of dual-edges of* $\widetilde{c}$*. We also regard* $\widetilde{c}$ *as a directed dual graph induced by its directed dual-edges.*

Definition 6.2 (Duplicate/nonduplicate dual-edge and line part)**.** *For any m.d.-cycle* $\widetilde{c}$*, a directed dual-edge* $(\widetilde{u},\widetilde{v})$ *of* $\widetilde{c}$ *is called a* duplicate dual-edge *if its reverse* $(\widetilde{v},\widetilde{u})$ *also appears in* $\widetilde{c}$*. Otherwise, it is called a* nonduplicate dual-edge.



Examples for showing the condition of m.d.-cycles. (a) A combination of directed dual-edges around a dual-vertex $\widetilde{v}$ that cannot appear "syntactically" in any m.d.-cycle. (Here we consider this particular combination around $\widetilde{v}$. There exist some m.d.-cycle that have these directed dual-edges (and more) around $\widetilde{v}$. By "syntactically" we mean that this combination does not appear as a part of any m.d.-cycle in any order.) (b) A set of directed dual-edges that may appear in some m.d.-cycle depending on their order; that is, in the order of (1),(2), …, (3),(4) under the anti-clockwise order, and in the order of (1),(4), …, (3),(2) under the clockwise order. (c) A set of directed dual-edges that does not appear in any m.d.-cycle in the order of (1),(2), …, (3),(4), or in the order of (3),(4), …, (1),(2). Let us call this property the *noncrossing property* of m.d.-cycle, which we will use several times later.

Figure 11: Examples of invalid/valid directed dual-edge sets for m.d.-cycles

Remark. *It is easy to see (Figure 10) that duplicate directed dual-edges of $\widetilde{c}$ form dual-path(s) in $\widetilde{c}$; this motivate us to call the set of duplicate directed dual-edges of $\widetilde{c}$ the* line part *of $\widetilde{c}$. On the other hand, nonduplicate dual-edges of $\widetilde{c}$ form dual-cycle(s). We will ignore the case where $\widetilde{c}$ consists of only duplicate dual-edges, and in the following, we may assume that $\widetilde{c}$ has at least one dual-cycle.*

Consider any m.d.-cycle $\widetilde{c}$. As mentioned above, we assume that it has at least one dual-cycle. Hence, the plane (where $\widetilde{G}$ is embedded) is separated by $\widetilde{c}$. We would like to classify these subplanes as "inside" or "outside" of $\widetilde{c}$. Intuitively speaking, we define the inside/outside of $\widetilde{c}$ by the left/right of the directed dual-edges of $\widetilde{c}$.

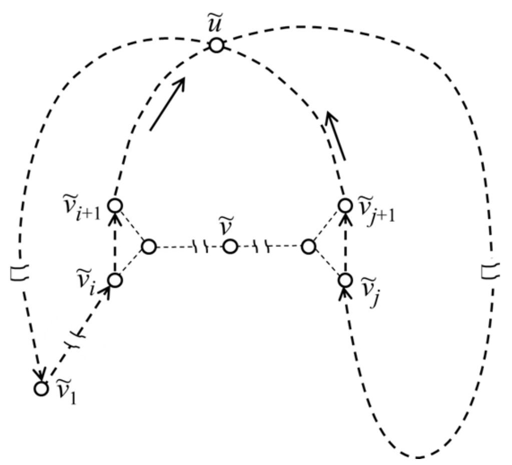
We define the left/right notion formally, and show that one can indeed determine for a given dual-vertex, whether it is located left or right of $\widetilde{c}$ uniquely. In the following explanation, we abuse the notation $\widetilde{G} \setminus \widetilde{c}$ to denote a subgraph of $\widetilde{G}$ obtained by "removing" $\widetilde{c}$; precisely, $\widetilde{G} \setminus \widetilde{c}$ is $\widetilde{G}[\widetilde{V} \setminus \mathrm{V}(\widetilde{c})]$, a subgraph of $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ induced by $\widetilde{V} \setminus \mathrm{V}(\widetilde{c})$. Also in the following, for any dual-edge $\widetilde{e}$ of $\widetilde{G}$, by a *dual-triangle upon* $\widetilde{e}$ and its *top-vertex*, we mean respectively a dual-triangle that uses $\widetilde{e}$ as one of its three dual-edges and its dual-vertex opposite to $\widetilde{e}$.

We first introduce the notion of "left/right adjacent." Consider any directed dual-edge $\widetilde{e}$ of $\widetilde{c}$. Since $\widetilde{G}$ is triangulated, there are two dual-triangles upon $\widetilde{e}$. We regard the one located left (resp., right) of $\widetilde{e}$ w.r.t. the direction of $\widetilde{e}$ as *left adjacent-triangle* (resp., *right adjacent-triangle*). Then the top-vertex of the left (resp., right) adjacent-triangle is called the *left adjacent-vertex* (resp., *right adjacent-vertex*) of $\widetilde{e}$. While these notions are defined for any dual-edge of $\widetilde{c}$, we consider only nonduplicate dual-edges to define the notion of "left/right side" of $\widetilde{c}$. For any dual-vertex we say it is in the *left* (resp., *right*) of $\widetilde{c}$ if it is connected in $\widetilde{G} \setminus \widetilde{c}$ to the left (resp., right) adjacent-vertex of some nonduplicate dual-edge of $\widetilde{c}$. The following lemma shows that the side of $\widetilde{c}$ is uniquely determined in this way.

Lemma 6.1. *Consider any m.d.-cycle $\widetilde{c}$ of $\widetilde{G}$. W.r.t. $\widetilde{c}$, the side of every dual-vertex of $\widetilde{G} \setminus \widetilde{c}$ is uniquely determined as left or right. For any duplicate dual-edge of $\widetilde{c}$, its left and right adjacent-vertices are in the same side; that is, they are both in the left (or in the right) of $\widetilde{c}$.*

Proof. We give a proof following Figure 12. Consider some m.d.-cycle $\widetilde{c}$, and suppose that there exists a dual-vertex $\widetilde{v}$ that is in the both left and right of $\widetilde{c}$. Let $(\widetilde{v}_i, \widetilde{v}_{i+1})$ and $(\widetilde{v}_j, \widetilde{v}_{j+1})$ be directed nonduplicate dual-edges of $\widetilde{c}$ witnessing that $\widetilde{v}$ is in the left/right of $\widetilde{c}$ respectively. Thus, there is a dual-path to $\widetilde{v}$ from the right (resp., left) adjacent-vertex of $(\widetilde{v}_i, \widetilde{v}_{i+1})$ (resp., $(\widetilde{v}_j, \widetilde{v}_{j+1})$) that has no common dual-vertex with $\widetilde{c}$. Without losing generality, we assume that $i < j$; that is, $(\widetilde{v}_i, \widetilde{v}_{i+1})$ appears earlier than $(\widetilde{v}_j, \widetilde{v}_{j+1})$ in $\widetilde{c}$. Consider two subsequences of $\widetilde{c}$, one from $\widetilde{v}_{i+1}$ to $\widetilde{v}_j$ and another from $\widetilde{v}_{j+1}$ to $\widetilde{v}_1$. They must cross at some dual-vertex $\widetilde{u}$ as shown in Figure 12, contradicting the noncrossing property of m.d.-cycle (Figure 11). □

Armed with this lemma, we can formally define the side of an m.d.-cycle. Intuitively, the inside of an m.d.-cycle $\widetilde{c}$ is subplane(s) located left of the non line part of $\widetilde{c}$.



The m.d.-cycle $\widetilde{c}$ considered in the proof of the lemma is indicated by bold dashed line with arrows showing its direction.

Figure 12: An example for the proof of Lemma 6.1

Definition 6.3 (Inside/outside of an m.d.-cycle)**.** *Consider any m.d.-cycle $\widetilde{c}$. A dual-vertex is* inside *(resp.,* outside*) of $\widetilde{c}$ if it is in the left (resp., right) of $\widetilde{c}$. A dual-triangle is* inside *(resp.,* outside*) of $\widetilde{c}$ if its top-vertex is inside (resp., outside) of $\widetilde{c}$. In general, the* inside *(resp.,* outside*) of $\widetilde{c}$ is the set of subplanes of the plane dual graph $\widetilde{G}$ consisting of faces (not including $\widetilde{c}$ itself) defined by dual-triangles inside (resp., outside) of $\widetilde{c}$.*

7 Dividing Voronoi Regions

In this section we introduce the notion of "ridge edge" and define the notion of "pre-frame-cycle" that is a base for a frame-cycle and a frame-graph. For our algorithm, this section corresponds to Step 2.2, and we explain algorithms for computing ridge edges and pre-frame-cycles.

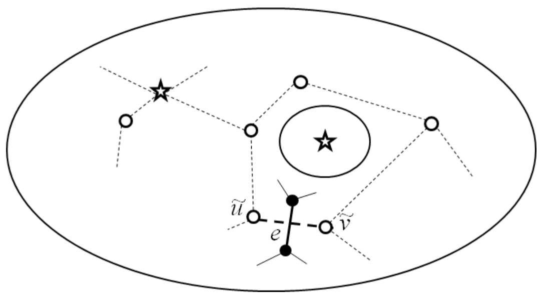
While a dual-cycle shown in Figures 5 is a good candidate for a frame-cycle, there is a situation where a more complicated dual-cycle is obtained similarly as illustrated in Figure 6, and we cannot bound in general the size of such dual-cycles. Such a situation occurs if some Voronoi region has multiple boundary cycles. Here we introduce "ridge edges" for adding more branch vertices, thereby dividing Voronoi regions and large dual-cycles appropriately.

Consider any $\widetilde{b} \in \widetilde{I}$ such that Voronoi region $\mathrm{Vr}(\widetilde{b})$ has more than one boundary cycle. Let us fix these $\widetilde{b}$ and $\mathrm{Vr}(\widetilde{b})$ for a while. Let $\widetilde{T}$ be the BFS dual-tree of $\mathrm{Vr}(\widetilde{b})$ that is obtained at the step 2.1 of our algorithm. Recall that it is an $O(k)$-depth spanning tree of dual-vertices in $\mathrm{Vr}(\widetilde{b})$. Consider any two adjacent dual-vertices $\widetilde{u}$ and $\widetilde{v}$ that are not connected in $\widetilde{T}$. Then by connecting $\widetilde{u}$ and $\widetilde{v}$ by $\widetilde{e} = \{\widetilde{u}, \widetilde{v}\}$, a dual-cycle is created[9] with two dual-paths of $\widetilde{T}$. This dual-cycle divides the plane into two subplanes. We consider the case where this separation divides the set of boundary cycles of $\mathrm{Vr}(\widetilde{b})$. Below we say that a dual-edge $\widetilde{e} = \{\widetilde{u}, \widetilde{v}\}$ *crosses* an edge e (and in parallel, e *crosses* $\widetilde{e}$) if two faces corresponding to $\widetilde{u}$ and $\widetilde{v}$ share the edge e.

Definition 7.1 (Ridge edge)**.** *Consider* $\mathrm{Vr}(\widetilde{b})$ *as a subgraph of G. An edge e of* $\mathrm{Vr}(\widetilde{b})$ *is a* ridge edge *if the dual-edge $\widetilde{e} = \{\widetilde{u}, \widetilde{v}\}$ of* $\mathrm{Vr}(\widetilde{b})$ *crossing e satisfies the following (below we let $\widetilde{c}(\widetilde{e})$ denote the dual-cycle induced by $\widetilde{e}$ and two dual-paths of $\widetilde{T}$ from the boss-vertex respectively to $\widetilde{u}$ and $\widetilde{v}$):*

- *$\widetilde{e}$ is not a dual-edge of the BFS dual-tree $\widetilde{T}$ of* $\mathrm{Vr}(\widetilde{b})$*, and*
- *each of two subplanes separated by $\widetilde{c}(e)$ contains at least one boundary cycle of* $\mathrm{Vr}(\widetilde{b})$.

[9]Precisely speaking, the created one should be considered as an m.d.-cycle instead of a simple cycle because two dual-paths of $\widetilde{T}$ may be merged before the boss-vertex. But since it does not have more than one dual-cycle, we discuss below, for simplicity, as if it were a simple cycle with an orientation so that we can specify its inside/outside.



An example of a ridge edge. A bold dashed line is a dual-edge connecting two dual-vertices $\widetilde{u}$ and $\widetilde{v}$ that creates a dual-cycle with BFS dual-paths containing a Voronoi boundary cycle (indicated by a solid line) in its both sides. Then an edge crossing this dual-edge $\{\widetilde{u}, \widetilde{v}\}$ (indicated by a bold line) is a ridge edge.

Figure 13: Ridge edge

Below we use $R_{\widetilde{b}}$ to denote the set of ridge edges in $\mathrm{Vr}(\widetilde{b})$. Use $B_{\widetilde{b}}$ to denote the set of boundary edges of $\mathrm{Vr}(\widetilde{b})$. These sets are sometimes regarded as subgraphs of G.

We first point out a simple but important property that is immediate from the definition: namely, no BFS dual-path of $\widetilde{T}$ crosses $R_{\widetilde{b}} \cup B_{\widetilde{b}}$. We then show the following two properties of $R_{\widetilde{b}}$ and $B_{\widetilde{b}}$.

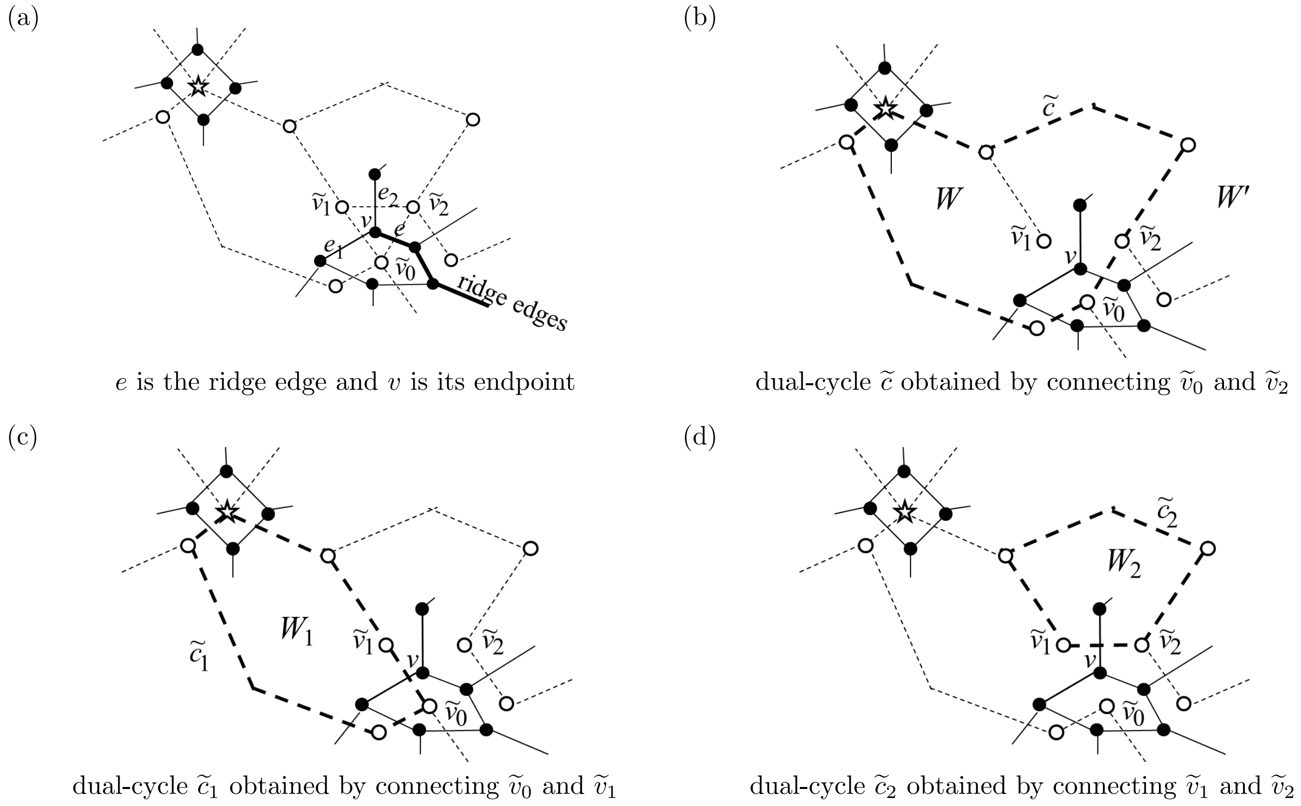


Figure 14: Examples for the proof of Lemma 7.1

Lemma 7.1. *$R_{\widetilde{b}}$ as a subgraph of G is a forest. Furthermore, every leaf of $R_{\widetilde{b}}$ is incident to some boundary edge of $B_{\widetilde{b}}$.*

Proof. Consider any connected component K of $R_{\widetilde{b}}$. We first show that K does not contain a cycle. If K had a cycle, then the cycle divides $\mathrm{Vr}(\widetilde{b})$ into more than two subplanes each of which contains at least one dual-vertex (since dual-vertices correspond to faces of G). From the definition of a ridge edge, any dual-edge of the BFS dual-tree $\widetilde{T}$ cannot cross a ridge edge; hence, there must be a subplane whose dual-vertices are not in the BFS dual-tree. This contradicts to the fact that $\widetilde{T}$ is a spanning tree of $\mathrm{Vr}(\widetilde{b})$.

Next we show that every leaf of K is incident to the boundary of $\mathrm{Vr}(\widetilde{b})$. Here we assume otherwise; that is, there is some leaf v of a connected component of K that is not incident to the boundary of $\mathrm{Vr}(\widetilde{b})$ and lead a contradiction.

Let e be the ridge edge that is incident to v. We consider three faces incident to v that are regarded as dual-vertices $\widetilde{v}_0$, $\widetilde{v}_1$, and $\widetilde{v}_2$ indexed in clockwise starting from the one adjacent to e so that a dual-edge $\{\widetilde{v}_0, \widetilde{v}_2\}$ crosses e. Let e_1 and e_2 are edges incident to v that are crossed by edges $\{\widetilde{v}_0, \widetilde{v}_1\}$ and $\{\widetilde{v}_1, \widetilde{v}_2\}$ respectively (Figure 14 (a)). Note that neither e_1 nor e_2 is a ridge edge (because v is a leaf of some connected component consisting of ridge edges). If $\widetilde{v}_1$ does not belong to $\mathrm{Vr}(\widetilde{b})$, e_1 and e_2 become boundary edges since $\widetilde{v}_0$ and $\widetilde{v}_2$ are in $\mathrm{Vr}(\widetilde{b})$, and v is incident to the boundary, a contradiction. Thus $\widetilde{v}_1$ is in $\mathrm{Vr}(\widetilde{b})$.

Consider a dual-cycle $\widetilde{c}$ induced by (two dual-paths of) the BFS dual-tree $\widetilde{T}$ and a dual-edge $\{\widetilde{v}_0, \widetilde{v}_2\}$ (Figure 14 (b)). The cycle $\widetilde{c}$ divides the plane into two subplanes; let W and W' denote them. Note that both contain at least one boundary cycle of $\mathrm{Vr}(\widetilde{b})$ since e is a ridge edge. Without losing generality, we may assume that W is the subplane containing v. Note that W contains $\widetilde{v}_0$, $\widetilde{v}_1$, and $\widetilde{v}_2$.

We have three cases:
(i) The dual-edges $\{\widetilde{v}_0, \widetilde{v}_1\}$ and $\{\widetilde{v}_1, \widetilde{v}_2\}$ are both in the BFS dual-tree $\widetilde{T}$: In this case, W becomes the triangle consisting of $\widetilde{v}_0$, $\widetilde{v}_1$ and $\widetilde{v}_2$. Obviously, W has no boundary cycle of $\mathrm{Vr}(\widetilde{b})$.
(ii) Either one of $\{\widetilde{v}_0, \widetilde{v}_1\}$ and $\{\widetilde{v}_1, \widetilde{v}_2\}$ is in $\widetilde{T}$: Without loss of generality, we assume that $\{\widetilde{v}_0, \widetilde{v}_1\}$ is in $\widetilde{T}$ and $\{\widetilde{v}_1, \widetilde{v}_2\}$ is not in $\widetilde{T}$. Consider a dual-cycle $\widetilde{c}'$ induced by $\widetilde{T}$ and a dual-edge $\{\widetilde{v}_1, \widetilde{v}_2\}$, and let the divided subplane by $\widetilde{c}'$ that does not contain v be U. The difference of W and U, namely $W \setminus U$, is the triangle consisting of $\widetilde{v}_0$, $\widetilde{v}_1$ and $\widetilde{v}_2$. Thus U contains a boundary cycle of $\mathrm{Vr}(\widetilde{b})$; hence, e_2 becomes a ridge edge. A contradiction.
(iii) Neither $\{\widetilde{v}_0, \widetilde{v}_1\}$ nor $\{\widetilde{v}_1, \widetilde{v}_2\}$ is in $\widetilde{T}$: Consider a dual-cycle $\widetilde{c}_1$ induced by $\widetilde{T}$ and a dual-edge $\{\widetilde{v}_0, \widetilde{v}_1\}$, and let W_1 denote one of the two subplanes divided by $\widetilde{c}_1$ that is entirely contained in W (Figure 14 (c)). Note that W_1 has no boundary cycle of $\mathrm{Vr}(\widetilde{b})$ because otherwise the other side of $\widetilde{c}_1$, which contains W', has no boundary of $\mathrm{Vr}(\widetilde{b})$ (since e_1 is not a ridge edge), contradicting the above mentioned fact that W' has at least one boundary cycle of $\mathrm{Vr}(\widetilde{b})$. Similarly, the subplane W_2 defined by a dual-cycle $\widetilde{c}_2$ induced by $\widetilde{T}$ and an edge $\{\widetilde{v}_1, \widetilde{v}_2\}$ (Figure 14 (d)) has no boundary cycle of $\mathrm{Vr}(\widetilde{b})$. Note, however, v is only one vertex of G that is in $W \setminus (W_1 \cup W_2)$. Thus, W has no boundary cycle of $\mathrm{Vr}(\widetilde{b})$. A contradiction. □

Lemma 7.2. *$B_{\widetilde{b}} \cup R_{\widetilde{b}}$ as a graph is connected.*

Proof. For proving by contradiction, suppose that $B_{\widetilde{b}} \cup R_{\widetilde{b}}$ is not connected. Let C and C' be two connected components of $B_{\widetilde{b}} \cup R_{\widetilde{b}}$. Here we note that both C and C' consist of some Voronoi boundary cycles and ridge edges connecting them, which is immediate from the above lemma.

Let $\widetilde{c}$ denote a set of dual-vertices of $\mathrm{Vr}(\widetilde{b})$ incident to edges of C. Due to the three regularity of G, it is easy to see that $\widetilde{c}$ forms an m.d.-cycle under the anti-clockwise[10] order whose orientation is determined so that C is located its outside. Note here the following two properties of $\widetilde{c}$: Firstly, no dual-vertex of $\mathrm{Vr}(\widetilde{b})$ belongs to the outside of $\widetilde{c}$; if otherwise, i.e., if there were some $\widetilde{v} \in \mathrm{Vr}(\widetilde{b})$ outside of $\widetilde{c}$, then the BFS dual-path from $\widetilde{b}$ to $\widetilde{v}$ should cross some edge of C, which would not occur by the basic property of $B_{\widetilde{b}}$ and $R_{\widetilde{b}}$. Thus, all dual-vertices of $\mathrm{Vr}(\widetilde{b})$ (including dual-vertices of $\widetilde{c}$) are located inside of $\widetilde{c}$. Then clearly, C' is also included in the inside of $\widetilde{c}$. Secondly, no edge of $B_{\widetilde{b}} \cup R_{\widetilde{b}}$ crosses $\widetilde{c}$, which is immediate from the first property.

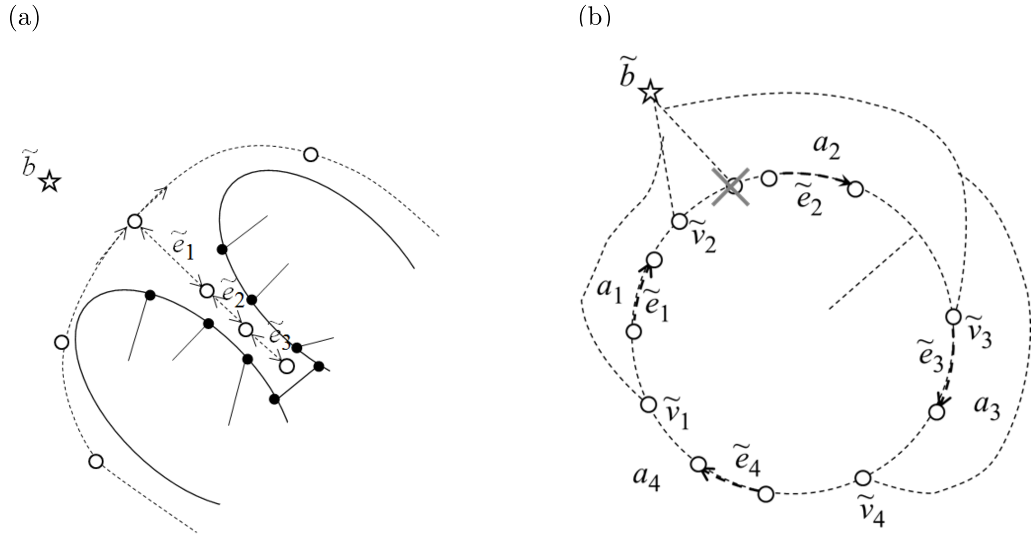
Among directed dual-edges of $\widetilde{c}$, there must be some dual-edges[11] that are not dual-edges of the BFS dual-tree $\widetilde{T}$. Let $\widetilde{e}_1, \ldots, \widetilde{e}_h$ denote such directed dual-edges of $\widetilde{c} \setminus \widetilde{T}$ listed in the order of the m.d.-cycle $\widetilde{c}$ (e.g., Figure 15 (b)). For each $\widetilde{e}_i$, consider a pair of *directed* dual-paths of $\widetilde{T}$ from the boss-vertex $\widetilde{b}$ to the tail of $\widetilde{e}_i$ and the head of $\widetilde{e}_i$ to $\widetilde{b}$, with the direction consistent with the directed dual-edge $\widetilde{e}_i$. Then this pair of dual-paths and $\widetilde{e}_i$ define a dual-cycle[12], which we denote by $\widetilde{a}_i$. Also denote by a_i the subplane(s) defined as the inside of $\widetilde{a}_i$. By our choice of the direction, $a_1, \ldots, a_h$ are

[10]The choice of the anti-clockwise order is necessary to define an m.d.-cycle consisting of *all* vertices of $\widetilde{c}$ for the case where $\widetilde{c}$ has a line part like the one shown in Figure 15 (a) (see Remark of Definition 6.2 for the definition of "line part"). Since C is located in the outside (i.e., the right) of the m.d.-cycle, no line part of the m.d.-cycle is created in its left. Thus, choosing dual-vertices in the anti-clockwise order is necessary and sufficient to collect all dual-vertices of $\widetilde{c}$.

[11]These dual-edges are not from the line part of $\widetilde{c}$; see the explanation of Figure 15 (a). Thus, its direction in $\widetilde{c}$ is uniquely determined.

[12]Again formally speaking, this may not be a real cycle because two dual-paths of the BFS dual-tree may merge before the boss-vertex. But for simplicity, we regard it as a simple dual-cycle.

included in the inside of $\widetilde{c}$; also it is clear that there is no overlap between them. Furthermore, every vertex of G located inside of $\widetilde{c}$ is in some a_i; see the explanation of Figure 15 (b). In this sense, the set of subplanes $a_1, \ldots, a_h$ is a partition of the inside of $\widetilde{c}$. In particular, there must be some $\widetilde{a}_i$ that contains some and hence all vertices of C' because no edges of C' can cross the dual-edges of $\widetilde{a}_i$. Then $\widetilde{e}_i$ satisfies the condition given in Definition 7.1. That is, the edge that crosses $\widetilde{e}_i$ is a ridge edge, which contradicts the second property of $\widetilde{c}$ confirmed above. □



Examples for the m.d.-cycle $\widetilde{c}$ considered in the proof of Lemma 7.2. (a) An example for the case where an m.d.-cycle $\widetilde{c}$ has a line part. Solid lines indicate edges of C, while dashed lines and white nodes are dual-edges and dual-vertices respectively of the m.d.-cycle $\widetilde{c}$ incident to C. Note that dual-edges of a line part, i.e., $\widetilde{e}_1, \widetilde{e}_2, \widetilde{e}_3$, are all dual-edges of the BFS dual-tree $\widetilde{T}$ from $\widetilde{b}$. Suppose otherwise and, say, $\widetilde{e}_2$ were non-tree dual-edge; then at least one of the two BFS dual-paths from $\widetilde{b}$ to $\widetilde{e}_2$ must cross C, a contradiction. (b) Thin dashed lines are dual-paths of the BFS dual-tree $\widetilde{T}$ from $\widetilde{b}$. Bold dashed lines are dual-edges $\widetilde{e}_1, \ldots, \widetilde{e}_4$ of $\widetilde{c}$ that do not belong to $\widetilde{T}$, where their order and directions are consistent with $\widetilde{c}$. For each $\widetilde{e}_i$, an dual-cycle is defined by two BFS dual-paths to the two end points of $\widetilde{e}_i$ and $\widetilde{e}_i$, with the direction consistent with that of $\widetilde{e}_i$, which define a subplane a_i as its inside. Clearly, a_i is located inside of $\widetilde{c}$, and furthermore, $a_1, \ldots, a_4$ are the partition of the inside of $\widetilde{c}$. To see this, for each $\widetilde{e}_i$, consider $\widetilde{v}_i$ (resp., $\widetilde{v}_{i+1}$) that is the first dual-vertex in the BFS dual-path from $\widetilde{b}$ to the tail (resp., the head) of $\widetilde{e}_i$. Then it is easy to see that each pair a_i and a_{i+1} of adjacent faces share the same boundary from $\widetilde{b}$ to $\widetilde{v}_{i+1}$; no situation like the one crossed by `X` occurs because thin dashed lines are dual-tree edges. Thus, all vertices located inside $\widetilde{c}$ must be in some face a_i.

Figure 15: Examples for the proof of Lemma 7.2

Now we are ready to define notions of "branch vertex" and "connector" formally.

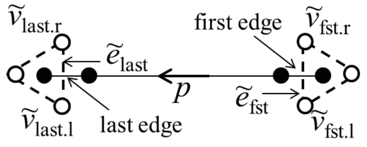
Definition 7.2 (B, R, and branch vertex). *Hereafter we use B and R to denote respectively the set of Voronoi boundary edges and the set of ridge edges defined for G and $\widetilde{G}$. Regard $B \cup R$ as an induced subgraph of G. A* branch vertex *is a degree three vertex in this graph. For each branch vertex v, a dual-triangle consisting of three dual-vertices incident to v is a* branch-triangle.

Remark. *From the above two lemmas on $B_{\widetilde{b}} \cup R_{\widetilde{b}}$, it is easy to see that $B \cup R$, as a graph, is a connected graph, and that every vertex of $B \cup R$ has degree either 2 or 3.*

Definition 7.3 (Connector). *Two branch vertices are* adjacent *if they are connected by edges in $B \cup R$ with no other branch vertex on it. The path connecting adjacent branch vertices is a* connector.

Remark. *It is easy to see that each connector consists of Voronoi boundary edges only or ridge edges only. Note that it is possible that some branch vertex is adjacent to itself; that is, it is connected to itself by a cycle connector. We fix some simple way to give a direction to each connector, and in the following, we may assume that each connector has this direction.*

We are ready to define the notion of "pre-frame-cycle", which will be used as a basis for defining "frame-cycles" and "frame-graph." We prepare some notation. Consider any connector p (see Figure 16). Following the remark of Definition 7.3, we may assume that p has a direction, and we use v_{fst} and v_{last} be the start and end vertices of p w.r.t. this direction. Note that v_{fst} and v_{last} are an adjacent pair of branch vertices. Let us call an edge of p having v_{fst} as its end point the *first edge*, and similarly, an edge of p having v_{last} as its end point the *last edge*. We will refer a part of p removing the first and last edges as a *body* of p and denote it by p'. Also let $\widetilde{e}_{\mathrm{fst}} = \{\widetilde{v}_{\mathrm{fst.r}}, \widetilde{v}_{\mathrm{fst.l}}\}$ and $\widetilde{e}_{\mathrm{last}} = \{\widetilde{v}_{\mathrm{last.r}}, \widetilde{v}_{\mathrm{last.l}}\}$ be dual-edges that cross the first and the last edges respectively, where $\widetilde{v}_{\mathrm{fst.l}}$ (resp., $\widetilde{v}_{\mathrm{last.l}}$ is the one located left of the first (resp., the last) edge, and $\widetilde{v}_{\mathrm{fst.r}}$ (resp., $\widetilde{v}_{\mathrm{last.r}}$) is the one located right of the first (resp., the last) edge.



$\widetilde{v}_{\mathrm{last.r}}$
$\widetilde{e}_{\mathrm{last}}$
first edge
$\widetilde{v}_{\mathrm{fst.r}}$
p
$\widetilde{e}_{\mathrm{fst}}$
$\widetilde{v}_{\mathrm{last.l}}$
last edge
$\widetilde{v}_{\mathrm{fst.l}}$

Figure 16: Notation on a connector

We can easily see that both $\widetilde{v}_{\mathrm{fst.l}}$ and $\widetilde{v}_{\mathrm{last.l}}$ (similarly, both $\widetilde{v}_{\mathrm{last.r}}$ and $\widetilde{v}_{\mathrm{last.r}}$) have the same boss dual-vertex.

Lemma 7.3. $\mathrm{boss}(\widetilde{v}_{\mathrm{fst.l}}) = \mathrm{boss}(\widetilde{v}_{\mathrm{last.l}})$ *and* $\mathrm{boss}(\widetilde{v}_{\mathrm{fst.r}}) = \mathrm{boss}(\widetilde{v}_{\mathrm{last.r}})$.

Proof. Consider dual-vertices incident to p that are located on the left side of p. We name them $\widetilde{v}_0 = \widetilde{v}_{\mathrm{fst.l}}, \widetilde{v}_1, \ldots, \widetilde{v}_{t-1}, \widetilde{v}_t = \widetilde{v}_{\mathrm{last.l}}$ so that $\widetilde{v}_i$ and $\widetilde{v}_{i+1}$ are adjacent. Assume that there exists $\widetilde{v}_i$ such that $\mathrm{boss}(\widetilde{v}_i) \neq \mathrm{boss}(\widetilde{v}_{i+1})$, then an edge e that crosses $\{\widetilde{v}_i, \widetilde{v}_{i+1}\}$ must be an edge of a Voronoi boundary. But since e is incident to p, the end point of e that is on p must be a branch vertex. This contradicts that p is a connecter. Therefore, there is no i such that $\mathrm{boss}(\widetilde{v}_i) \neq \mathrm{boss}(\widetilde{v}_{i+1})$, and hence $\mathrm{boss}(\widetilde{v}_{\mathrm{fst.l}}) = \mathrm{boss}(\widetilde{v}_{\mathrm{last.l}})$. With a similar argument, we can show $\mathrm{boss}(\widetilde{v}_{\mathrm{fst.r}}) = \mathrm{boss}(\widetilde{v}_{\mathrm{last.r}})$. □

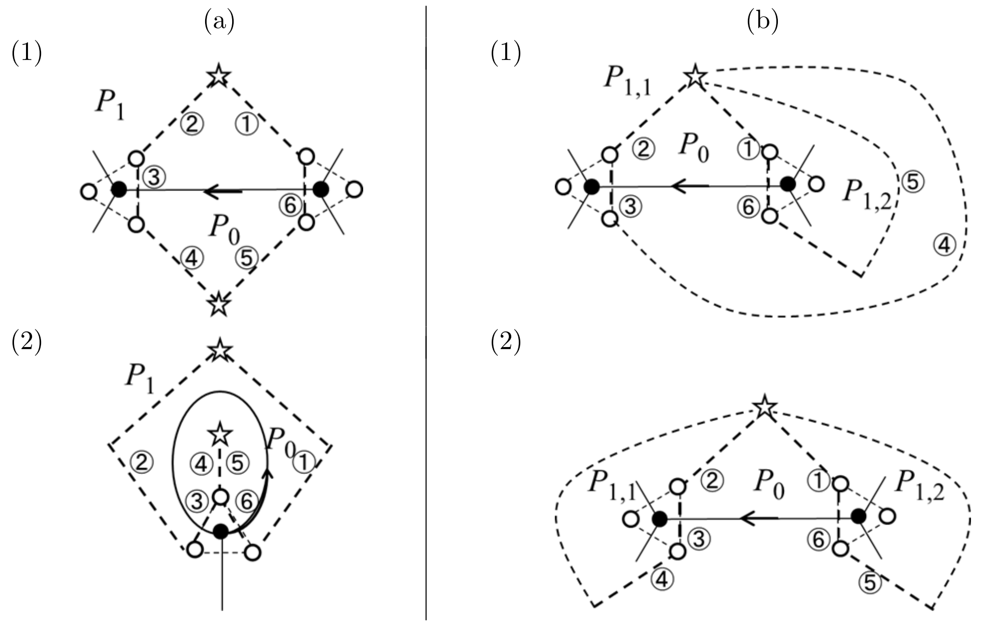
Then the notion of "pre-frame-cycle" is defined as follows. (This notion will be revised in the next section. Thus, let us call the one defined here as "the first version.")

Definition 7.4 (pre-frame-cycle, the first version)**.** *Use notation defined in Figure 16; also see Figure 17. For any connector p, a* pre-frame-cycle *(*w.r.t. *p) is a "directed dual-cycle" that consists of (1) a BFS dual-path from $\widetilde{v}_{\mathrm{fst.r}}$ to its boss-vertex, (2) a BFS dual-path from this boss-vertex to $\widetilde{v}_{\mathrm{last.r}}$, (3) a branch-triangle dual-edge $\widetilde{e}_{\mathrm{last}}$ from $\widetilde{v}_{\mathrm{last.r}}$ to $\widetilde{v}_{\mathrm{last.l}}$, (4) a BFS dual-path from $\widetilde{v}_{\mathrm{last.l}}$ to its boss-vertex, (5) a BFS dual-path from this boss-vertex to $\widetilde{v}_{\mathrm{fst.l}}$, and (6) a branch-triangle dual-edge $\widetilde{e}_{\mathrm{fst}}$ from $\widetilde{v}_{\mathrm{fst.l}}$ to $\widetilde{v}_{\mathrm{fst.r}}$. The direction of this dual-cycle is defined naturally from the above order of dual-edges and dual-paths, from which the connector p is located in the left, i.e., the inside, of the pre-frame-cycle.*
Remark. *As shown in Figure 17, a pre-frame-cycle is not a simple dual-cycle in general; it may have a line part and/or it may consist of two dual-cycles. Thus, it seems better to regard it as an m.d.-cycle, more specifically, an m.d.-cycle under the clockwise order from Figure 17. Unfortunately, though, there are cases where an important part of a pre-frame-cycle is missed if we consider an m.d.-cycle under the clockwise order; see, e.g., Figure 18 (b), (c). Furthermore, we will eliminate the cases where more than two dual-cycles appear as Figure 17 (b). Thus, though some pre-frame-cycle may have a line part, we regard it as a simple directed dual-cycle.*

A dual-cycle we saw in Figure 5 is in fact a pre-frame-cycle given as Figure 17 (a) (1). On the other hand, a dual-cycle like the one in Figure 6 is now divided into several pre-frame-cycles thanks to the introduction of new branch vertices; see Figure 19.

Algorithms for Step 2.2

The task of this step is to compute information of the pre-frame-cycles of G. We again explain as if each step receives the output of the previous step as input, and omit the analysis of time complexity. Note that we can use the outputs of Step 2.1 as input; the set of boss-vertices $\widetilde{I}$, the BFS tree $\widetilde{T}_{\widetilde{b}}$ of every Voronoi region $\mathrm{Vr}(\widetilde{b})$ ($\widetilde{b} \in \widetilde{I}$), a list of boundary cycles of all Voronoi regions and a list of pairs of Voronoi regions sharing a vertex. We consider an algorithm that outputs



This figure shows four typical pre-frame-cycles. pre-frame-cycles are defined by bold dashed lines following the order indicated by numbers. Two boss-vertices are used for a pre-frame-cycle in type (a), and only one boss-vertex is used in type (b). Note that two subplanes are created as the inside/outside of the pre-frame-cycles in (a), whereas three subplanes are created in (b). Also note that a connector, i.e., a solid line with an arrow, is included (except its end points) in the inside subplane P_0 of a pre-frame-cycle for every type. A pre-frame-cycle of type (a) (2) has a line part, and more examples of pre-frame-cycles with a line part are shown in Figure 18.

Figure 17: Typical pre-frame-cycles

(a) the set $B \cup R$ of Voronoi boundary edges and ridge edges;

(b) the set of branch vertices of $B \cup R$;

(c) a list of connectors; and

(d) a list of pre-frame-cycles[13].

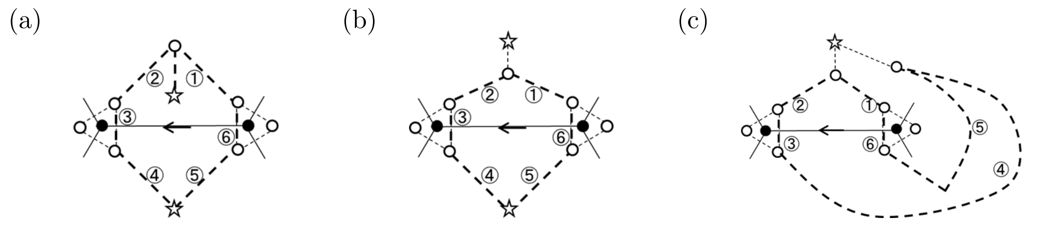
First, for computing $B \cup R$ as the output of (a), we compute $R_{\widetilde{b}}$ for each $\widetilde{b} \in \widetilde{I}$. Note that the boundaries $B_{\widetilde{b}}$ are the input of the algorithm. If $\mathrm{Vr}(\widetilde{b})$ has only one boundary cycle, $R_{\widetilde{b}}$ is empty from the definition of ridge edge. Assume $\mathrm{Vr}(\widetilde{b})$ has at least two boundary cycles. For each non-tree edge $\widetilde{e} = \{\widetilde{u}, \widetilde{v}\}$ of $\mathrm{Vr}(\widetilde{b})$, we compute the path from $\widetilde{u}$ to $\widetilde{v}$ on $\widetilde{T}_{\widetilde{b}}$ with a standard DFS (depth-first search) algorithm. Since the diameter of $\widetilde{T}_{\widetilde{b}}$ is $O(k)$, this runs in $\widetilde{O}(k)$-space. We denote the dual-cycle[14] consisting of this path and $\widetilde{e}$ as $\widetilde{c}_{\widetilde{e}}$, where its direction is determined in any appropriate way. For each boundary cycle of $B_{\widetilde{b}}$, fix one dual-vertex $\widetilde{v}_B$ of $\mathrm{Vr}(\widetilde{b})$ adjacent to $B_{\widetilde{b}}$, and run a DFS algorithm from $\widetilde{v}_B$ on $\widetilde{T}_{\widetilde{b}}$ until getting to some vertex of $\widetilde{c}_{\widetilde{e}}$, and check the side of $\widetilde{v}_B$ (and hence, the boundary cycle) is located, which is again computed in $\widetilde{O}(k)$-space. If both sides of $\widetilde{c}_{\widetilde{e}}$ have at least one boundary cycle, we add the edge crossing $\widetilde{e}$ to $R_{\widetilde{b}}$.

Next consider how to compute the branch vertices. For any vertex v of G, we simply check whether v appears in $B \cup R$ three times or not.

Now we compute connectors as output (c). For each branch vertex v, we traverse on $B \cup R$ from v to all three directions until reaching another branch vertex u (u may be equal to v). The path from v to u is a connector. We outputs the path only if v has smaller or equal index than u for not outputting a connector twice. The algorithm needs $\widetilde{O}(1)$-space.

[13]This was not the output of Step 2.2 explained in Outline section; but we added this for the sake of explanation in the next section

[14]Here again we explain, for simplicity, as if $\widetilde{c}_{\widetilde{e}}$ is a simple dual-cycle, though it may have a line part.



Pre-frame-cycles may have line parts when it has two BFS dual-paths (with the same boss-vertex) merges before reaching to their boss-vertex. Here are some typical examples. Figures (b) and (c) show the cases where the line part indicated by thin dashed lines (and hence the corresponding boss-vertex) cannot be included if we regard them as m.d.-cycles under the clockwise order.

Figure 18: Examples of line parts of pre-frame-cycles

Finally, we compute pre-frame-cycle as output (d). For each connector, we compute and output BFS dual-paths and branch-triangle dual-edges according to the definition of pre-frame-cycle. The dual-paths can be computed by DFS in $\widetilde{O}(k)$-space. Then a sequence (similar to an m.d.-cycle) representing a pre-frame-cycle can be easily obtained from these dual-paths and branch-triangle dual-edges in $\widetilde{O}(k)$-space.

8 Frame-Graph

We define the notion of "frame-graph" formally and show that an obtained frame-graph satisfying the condition (F1) $\sim$ (F3). This section corresponds to Step 2.3 of our algorithm.

Roughly speaking, our frame-graph is defined by removing all dual-vertices (and dual-edges) not participating in any pre-frame-cycle or branch-triangle, and these pre-frame-cycle and branch-triangles define all the faces of the frame-graph. The actual situation is a bit more complicated in general because of the type (b) of Figure 14 in which the 2-connectivity may not be guaranteed by removing unused dual-edges. Fortunately, we can show that this situation can be avoided by preprocessing pre-frame-cycles and merging two subplanes for type (b) graphs (if necessary).

8.1 Preprocessing pre-frame-cycles

We show a preprocessing step to simplify pre-frame-cycles so that each simplified pre-frame-cycle becomes a simple dual-cycle by removing (at most two) line parts. Thus, we will simply call such a simplified pre-frame-cycles as a dual-cycle although it may contain a line part.

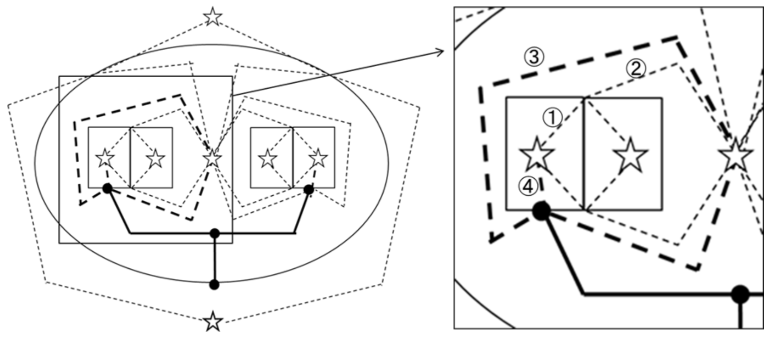
In this preprocessing step, we go through all pre-frame-cycles, and for each pre-frame-cycle $\widetilde{c}$, we check the number of dual-vertices in each subplane defined as the inside of $\widetilde{c}$ and then take one of the following actions depending on these numbers: (i) do nothing, that is, keep $\widetilde{c}$ as it is (only for the type (a)), (ii) use the algorithm of Lipton and Tarjan (Proposition 2.3) to compute a separator and use it as a desired separator of $\widetilde{G}$, (iii) merge two subplanes (only for the type (b)), or (iv) use the pre-frame-cycle $\widetilde{c}$ as a desired separator of $\widetilde{G}$. Recall that we can compute a target separator of $G_{\rm org}$ from a separator of $\widetilde{G}$ as explained in Outline section (i.e., Claim 2). Thus, the whole algorithm terminates with a desired output if either (ii) or (iv) is executed.

Now we explain this outline in detail. Consider any pre-frame-cycle $\widetilde{c}$, and let us use the symbols given in Figure 17 for $\widetilde{c}$. In particular, P_0 denotes a subplane defined as the inside of $\widetilde{c}$ that has the body of a connector p, and let P_1 (resp., $P_{1,1}$ and $P_{1,2}$) a subplane(s) located outside of $\widetilde{c}$. Let $\widetilde{n}_0$ (resp., $\widetilde{n}_1$, $\widetilde{n}_{1,1}$, $\widetilde{n}_{1,2}$) be the number of dual-vertices in P_0 (resp., P_1, $P_{1,1}$, $P_{1,2}$), not including dual-vertices on its boundary $\widetilde{c}$. Let $\widetilde{n}_{\widetilde{c}}$ denote the number of dual-vertices of $\widetilde{c}$; recall that $\widetilde{n}_{\widetilde{c}} = O(k)$ $(= O(\sqrt{n}))$, which can be assumed negligible compared with $\widetilde{n}$.

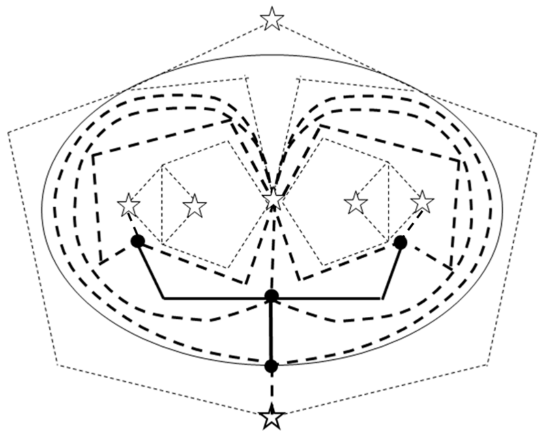
First consider the type (a). We have the following three cases (here we do not have the case (iii) mentioned above):

(i) If $\widetilde{n}_0 < \widetilde{n}/4$: do nothing and use $\widetilde{c}$ as a pre-frame-cycle.

(a)



(b)



An example of new branch vertices that introduce new pre-frame-cycles. We use the Voronoi region of Figure 6. Bold solid lines are paths consisting of ridge edges connecting three Voronoi boundary cycles (which are indicated by thin solid lines in (a)). Black vertices are new branch vertices, from which new branch-triangles (not shown here) are created. Then new branch dual-paths are introduced; for visibility, only three of them are shown as bold dashed lines in (a) while all of them are shown as bold dashed lines in (b). In (a), a pre-frame-cycle is indicated by its component four dual-paths given numbers in the order of its direction. Note that this directed dual-cycle contains one connector. Similarly, we can see in (b) (though a bit busy figure) that the long dual-cycle of Figure 6 is now separated into pre-frame-cycles.

Figure 19: New branch vertices and obtained pre-frame-cycles

(ii) If $\widetilde{n}_0 \geq 2\widetilde{n}/3$: Note first that P_0, that is, the inside of $\widetilde{c}$ is included in the union of two Voronoi regions. Consider one of these Voronoi regions having at least half of dual-vertices in P_0. Let $\widetilde{G}_0$ be an induced subgraph of $\widetilde{G}$ consisting of dual-vertices of $\widetilde{G}$ located in this part; note that it has at least $\widetilde{n}/3$ dual-vertices. Recall that we have a BFS dual-tree covering dual-vertices of $\widetilde{G}_0$ of depth $O(k)$ $(= O(\sqrt{n}))$, and also that this dual-tree (Section 5) and the list of dual-vertices inside of $\widetilde{c}$ (Section 7) are given as input. Thus, we may assume that a BFS dual-tree of $\widetilde{G}_0$ is given. Then we can apply the algorithm of Lipton and Tarjan to $\widetilde{G}_0$ to obtain a separator of $\widetilde{G}_0$ that separates $\widetilde{G}_0$ into two subgraphs each of which has at least $(\widetilde{n}/3)/3 = \widetilde{n}/9$ dual-vertices. Since the work space needed for the algorithm of Lipton and Tarjan is linearly bounded by the depth of a BFS dual-tree of a given graph, this computation can be executed in $\widetilde{O}(\sqrt{n})$-space. Clearly, the obtained separator is also a 1/9-separator for $\widetilde{G}$, that is, our desired separator.

(iv) Otherwise: In this case we have $\widetilde{n}_0 \geq \widetilde{n}/4$ and $\widetilde{n}_1 \geq \widetilde{n} - \widetilde{n}_0 - \widetilde{n}_{\widetilde{c}}$, which is larger than, say, $\widetilde{n}/4$. Thus, we can output $\widetilde{c}$ as a 1/4-separator of $\widetilde{G}$.

Next consider the type (b). We have the following three cases (here we do not have the case (i) mentioned above):

(ii) If $\widetilde{n}_0 \geq \widetilde{n}/3$: In this case, for an induced subgraph $\widetilde{G}_0$ of $\widetilde{G}$ consisting of dual-vertices of $\widetilde{G}$ in P_0, we take the same action as (ii) for the type (a).

(iii) If either $\widetilde{n}_0+\widetilde{n}_{1,1} < \widetilde{n}/4$ or $\widetilde{n}_0+\widetilde{n}_{1,2} < \widetilde{n}/4$: Suppose that $\widetilde{n}_0+\widetilde{n}_{1,1} < \widetilde{n}/4$. Then we merge P_0 and $P_{1,1}$, and use a dual-cycle $\widetilde{c}'$ bounding this subplane as a replacement of $\widetilde{c}$ and *all* pre-frame-cycles in P_0 and $P_{1,1}$. More precisely, $\widetilde{c}'$ is a sequence of dual-edges following (5) $\rightarrow$ (6) $\rightarrow$ (1) of (b) in Figure 17. Then remove all pre-frame-cycles located left of $\widetilde{c}'$, i.e., in the subplane $P_0 \cup P_{1,1}$, from the list of pre-frame-cycles; that is, a new pre-frame-cycle $\widetilde{c}'$ becomes the boundary dual-cycle of the subplane $P_0 \cup P_{1,1}$ and no pre-frame-cycle exist in it. The case where $\widetilde{n}_0 + \widetilde{n}_{1,2} < \widetilde{n}/4$ is handled in a similar way, and use a dual-cycle following (2) $\rightarrow$ (3) $\rightarrow$ (4) to replace $\widetilde{c}$ and all pre-frame-cycles in the subplane $P_0 \cup P_{1,2}$.

(iv) Otherwise: In this case, we have $\widetilde{n}_{1,1} + \widetilde{n}_{1,2} \geq \widetilde{n} - \widetilde{n}_0 - \widetilde{n}_{\widetilde{c}}$, which is larger than, say, $\widetilde{n}/2$ (since $\widetilde{n}_0 < \widetilde{n}/3$). Thus, either $\widetilde{n}_{1,1}$ or $\widetilde{n}_{1,2}$ must be larger than $\widetilde{n}/4$. Let us assume that $\widetilde{n}_{1,1} > \widetilde{n}/4$. Then since $\widetilde{n}_0 + \widetilde{n}_{1,2} \geq \widetilde{n}/4$, $\widetilde{c}$ is a 1/4-separator separating $\widetilde{G}$ into subgraphs located in $P_{1,1}$ and $P_0 \cup P_{1,2}$. Thus, we output $\widetilde{c}$ as a desired separator of $\widetilde{G}$.

In the rest of this paper, we consider the situation where the algorithm does not terminate during this preprocessing and we still need to compute a separator. Also we revise our notion of pre-frame-cycle, and consider these dual-cycles obtained by this preprocessing as *pre-frame-cycles*. We should remark here that the inside of each pre-frame-cycle has less than $\widetilde{n}/4$ dual-vertices of $\widetilde{G}$.

Now we define "frame-graph" formally as follows.

Definition 8.1 (Frame-graph)**.** *For each pre-frame-cycle $\widetilde{c}$ of $\widetilde{G}$, let $\widetilde{E}_{\widetilde{c}}$ be the set of dual-edges that appear once in $\widetilde{c}$. Let $\widetilde{E}_1$ be the union of $\widetilde{E}_{\widetilde{c}}$ for all pre-frame-cycles $\widetilde{c}$, and let $\widetilde{E}_2$ be the set of all branch-triangle edges. Then a* frame-graph *of $\widetilde{G}$ is a subgraph $\widetilde{H} = (\widetilde{U}, \widetilde{D})$ of $\widetilde{G}$, where $\widetilde{D} = \widetilde{E}_1 \cup \widetilde{E}_2$ and $\widetilde{U}$ is the set of all dual-vertices that are end points of dual-edges of $\widetilde{D}$. A frame-graph is a plane graph under the embedding of $\widetilde{G}$ restricted to $\widetilde{U}$ and $\widetilde{D}$.*

Remark. *From the definition of "pre-frame-cycle" (i.e., Figure 17 and our preprocessing step), a pre-frame-cycle $\widetilde{c}$ is one dual-cycle that may have at most two line part(s) if dual-paths merge before its boss-vertices; hence, the set $\widetilde{E}_{\widetilde{c}}$ of dual-edges that appear once is exactly the dual-cycle part of $\widetilde{c}$. Thus, we call it a* frame-cycle *and denote it by $(\widetilde{c})^-$ if we want to consider it as a directed dual-cycle whose direction is consistent with $\widetilde{c}$ (while $\widetilde{E}_{\widetilde{c}}$ is simply a set of dual-edges).*

As explained in Outline section, a frame-graph $\widetilde{H}$ needs to be a weighted subgraph of $\widetilde{G}$. Here to meet the condition (F1), we define the weight of each face of $\widetilde{H}$ as follows.

Definition 8.2 (Frame-graph and face weight)**.** *For each face of a frame-graph $\widetilde{H}$, its* weight *is the number of dual-vertices of $\widetilde{G}$ located in the face (that are removed to define $\widetilde{H}$) divided by the total number $\widetilde{n}^-$ of dual-vertices removed from $\widetilde{G}$.*

Algorithms for Step 2.3

The task of this step is to compute $\widetilde{H}$, that is, its complete weighted face information, which consists of the list of branch-triangles and frame-cycles (as face boundary dual-cycles), the weights of faces defined by them, and their incidence relations. Since we have already computed pre-frame-cycles in Step 2.2 and the preprocessing step, we only need to consider a way to remove line parts from each pre-frame-cycle to obtain the corresponding frame-cycle. From the definition of pre-frame-cycle, Note that a line part exists in a pre-frame-cycle only if two dual-paths from two branch-triangles merge before the boss-vertex; such a line part can be identified by a standard DFS using $\widetilde{O}(k)$-space. Thus, we can compute the list of all frame-cycles of $\widetilde{H}$ in $\widetilde{O}(k)$-space. From the obtained list of frame-cycles, it is easy to compute their incidence relations. Note that the number of removed dual-vertices from the inside of each frame-cycle has been already computed in the preprocessing step.

8.2 Properties of the frame-graph.

We show that the frame-graph $\widetilde{H}$ satisfies the conditions (F1) $\sim$ (F3) given in Outline section.

We start with showing the 2-connectivity of $\widetilde{H}$, that is, the condition (F2).

Lemma 8.1. *$\widetilde{H} = (\widetilde{U}, \widetilde{D})$ is 2-connented.*

Proof. For any two distinct dual-vertices $\widetilde{u}, \widetilde{v} \in \widetilde{U}$, we show that there exists two vertex disjoint dual-paths from $\widetilde{u}$ to $\widetilde{v}$ in $\widetilde{H}$. When both $\widetilde{u}$ and $\widetilde{v}$ are on the same frame-cycle or the same branch dual-triangle, there exist two dual-paths that are parts of the dual-cycle from $\widetilde{u}$ to $\widetilde{v}$ clockwise and anti-clockwise. Suppose $\widetilde{u}$ and $\widetilde{v}$ are on two frame-cycles $\widetilde{c}$ and $\widetilde{c}'$ respectively. Let p_0 and p_* be the connectors whose bodies are contained in $\widetilde{c}$ and $\widetilde{c}'$ respectively. Since $B \cup R$ is connected, there is a sequence of connecters $p_0 = p_1, p_2, \ldots, p_k = p_*$ such that p_i and p_{i+1} share one endpoint for each i, $1 \le i < k$. Note also that for each i, $1 < i < k$, the body of p_i is contained in a frame-cycle $\widetilde{c}_i$ of the type (a)(1) of Figure 17. We re-define the directions of p_i so that we can traverse from p_1 to p_k. For every connector p_i, it is obvious that two dual-paths on $\widetilde{c}_i$ one from $\widetilde{v}_{\mathrm{fst.l}}$ to $\widetilde{v}_{\mathrm{last.l}}$ another from $\widetilde{v}_{\mathrm{fst.r}}$ to $\widetilde{v}_{\mathrm{last.r}}$ share no dual-vertex. We call these two dual-paths left- and right-path respectively. Then we can construct a path $\widetilde{p}_L$ (resp., $\widetilde{p}_R$) from $\widetilde{v}_{\mathrm{fst.l}}$ (resp., $\widetilde{v}_{\mathrm{fst.r}}$) of p_2 to $\widetilde{v}_{\mathrm{last.l}}$ (resp., $\widetilde{v}_{\mathrm{last.r}}$) of p_{k-1} by using only left- (resp., right-) paths and dual-edges of the corresponding branch-triangles; clearly, these two dual-paths $\widetilde{p}_L$ and $\widetilde{p}_R$ share no dual-vertex. That is, they are two vertex disjoint dual-paths from $\widetilde{u}$ to $\widetilde{v}$. (We need to consider the cases where $\widetilde{u}$ or $\widetilde{v}$ is on a branch-triangle or on a frame-cycle based on a pre-frame-cycle created in the preprocessing step; these cases can be treated similarly, and we omit detail explanation for these cases.) □

Next we consider the conditions (F1) and (F3). For this we need to identify faces of $\widetilde{H}$. Intuitively, it is almost clear that every face of $\widetilde{H}$ is defined by either a branch-triangle or a frame-cycle. We prove below this intuition. Recall that we define our frame-graph $\widetilde{H}$ by collecting necessary dual-edges instead of by removing dual-vertices from the inside of each pre-frame-cycle. Thus, we need to prove that there is no dual-vertex of $\widetilde{H}$ inside of each frame-cycle to guarantee that each frame-cycle indeed is a boundary of a face of $\widetilde{H}$. We also need to prove that there is no other face other than those defined by branch-triangles or frame-cycles.

We first discuss (almost) one-to-one correspondence between pre-frame-cycles and connectors in $\widetilde{G}$. Consider any pre-frame-cycle $\widetilde{c}$ that is not among those introduced as a boundary dual-cycle of two merged subplanes in the preprocessing step. From the definition of pre-frame-cycle, it is clear that the inside of $\widetilde{c}$ has the body of some connector. We show that it indeed does not contain any other edge of $B \cup R$.

Lemma 8.2. *Consider $\widetilde{G}$ and its pre-frame-cycle $\widetilde{c}$ that is not introduced in the preprocessing step. Let p' be the body of a connector p that is located inside of $\widetilde{c}$. Then there is no edge of $B \cup R$ except p' that is inside of $\widetilde{c}$.*

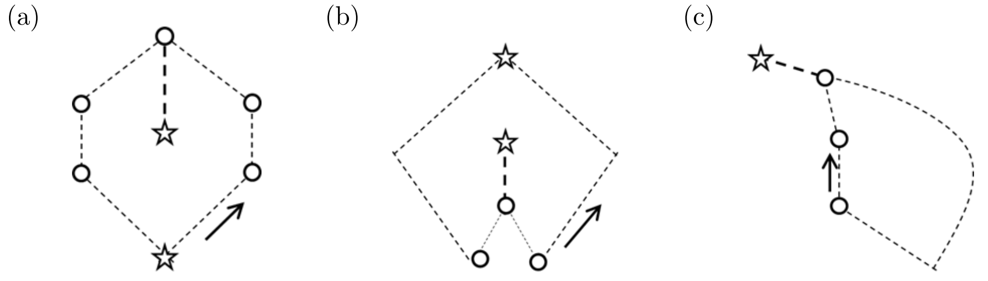
Proof. We use the notation of Figure 16 here with p and $\widetilde{c}$ of the lemma. Since the BFS dual-tree of each Voronoi region does not cross any edge of $R \cup B$, only two dual-edges $\widetilde{e}_{\mathrm{fst}}$ and $\widetilde{e}_{\mathrm{last}}$ of $\widetilde{c}$ are crossing edges of $R \cup B$. Therefore, $R \cup B$ is parted into at most three connected components by $\widetilde{c}$, and each component is in either in the left side of $\widetilde{c}$ (that is, inside of $\widetilde{c}$) or in the right side of $\widetilde{c}$ (that is, not outside of $\widetilde{c}$). Clearly, one component of them is p' and it is inside of $\widetilde{c}$. The others[15] contain the branch vertices of p, i.e., two end points of p that are not in the pre-frame-cycle $\widetilde{c}$; hence, those components have no edge in the inside of $\widetilde{c}$. □

We now show that there is no dual-vertex of $\widetilde{H}$ inside of each pre-frame-cycle. Thus, each frame-cycle is indeed a boundary of a face of $\widetilde{H}$. In the following, we consider $\widetilde{H}^+$ that is defined as a subgraph of $\widetilde{G}$ consisting of all dual-vertices and dual-edges of branch-triangles and pre-frame-cycles. Note that $\widetilde{H}$ is obtained from $\widetilde{H}^+$ by removing dual-vertices and dual-edges not participating in frame-cycles or branch-triangles.

Lemma 8.3. *Consider any pre-frame-cycle $\widetilde{c}$. No dual-edge of $\widetilde{H}^+ \setminus \widetilde{c}$ is located inside of $\widetilde{c}$. Thus, the inside of $(\widetilde{c})^-$ has no dual-vertex of $\widetilde{H}$.*
Remark. *It is possible that the inside of $(\widetilde{c})^-$ has some dual-vertex of $\widetilde{c}$, which is a dual-vertex of a line part of $\widetilde{c}$ that is removed in $\widetilde{H}$ (Figure 20).*

Proof. Consider any pre-frame-cycle $\widetilde{c}$. If it is one of those introduced in the preprocessing step, then the lemma is immediate because the inside of $\widetilde{c}$ has no pre-frame-cycle (due to the preprocessing step), and hence all dual-vertices are removed from there when defining $\widetilde{H}$. Thus, in the following, we consider a pre-frame-cycle $\widetilde{c}$ of the type (a) of Figure 17. In particular, we assume for simplicity that

[15]There is a case where p forms a cycle from a branch vertex (Figure 17 (a)(2)), in which case $B \cup R$ has only one component other than p'; this case can be treated similarly.



Typical examples of line parts of pre-frame-cycles that are removed from $\widetilde{H}^+$ when defining $\widetilde{H}$. Dashed lines (both bold and thin) are pre-frame-cycles (where the arrows indicate the directions for defining their insides), bold dashed lines are their line parts removed in $\widetilde{H}$. Figure (c) indicates the case where a new pre-frame-cycle is created by merging two suplanes in the preprocessing step.

Figure 20: Examples of line parts removed in $\widetilde{H}$

it is of the type (a)(1); the type (a)(2) can be treated similarly. Let p' be the body of a connector located inside of $\widetilde{c}$.

Assume to the contrary that some dual-edge $\widetilde{e}$ of the other pre-frame-cycle $\widetilde{c}'$ exists inside of $\widetilde{c}$. It cannot be a dual-edge of a branch-triangle because if so, the inside of $\widetilde{c}$ would have an edge of $B \cup R$ other than p', contradicting to Lemma 8.2 above. Hence, it must be a part of a dual-path $\widetilde{p}$ of a BFS dual-tree. Since there is no Voronoi boundary (besides p'), the boss of this BFS dual-tree (and one end of the dual-path $\widetilde{p}$) is the boss-vertex of $\widetilde{c}$ located in the same side of p' as $\widetilde{e}$; let us denote it $\widetilde{b}$. On the other hand, the other end of this dual-path in $\widetilde{c}'$ must be a branch-triangle that is located outside of $\widetilde{c}$ because the corresponding branch vertex must be outside of $\widetilde{c}$. Thus, the dual-path $\widetilde{p}$ must cross $\widetilde{c}$, and let $\widetilde{w}$ denote a dual-vertex of the crossing point; that is, $\widetilde{w}$ is a dual-vertex in both $\widetilde{p}$ and $\widetilde{c}$. Clearly, its boss-vertex is $\widetilde{b}$ because $\widetilde{p}$ cannot cross p'. Thus, we have two dual-paths from $\widetilde{w}$ to $\widetilde{b}$, one following $\widetilde{c}$ and another following $\widetilde{p}$, contradicting the fact that each dual-vertex has a unique BFS dual-path to its boss-vertex. □

From this lemma, we can claim that each frame-cycle is a boundary of a face in $\widetilde{H}$ because it is a dual-cycle (see Remark of Lemma 8.1) having no dual-vertex in its inside in $\widetilde{H}$.

Lemma 8.4. *The boundary of any face of frame-graph $\widetilde{H}$ is either a frame-cycle or a branch-triangle.*

Proof. Consider any face of $\widetilde{H}$. Since $\widetilde{H}$ is 2-connected, the face has a single boundary dual-cycle $\widetilde{c}$ (Proposition 2.1). Let us assume that this is not a branch-triangle and show that it indeed is a frame-cycle.

Note first that $\widetilde{c}$ has at least one dual-edge from some branch-triangle. This is because $\widetilde{H}$ consists of dual-edges of disjoint BFS dual-trees of $\widetilde{G}$ (from disjoint Voronoi regions) and branch dual-triangles and we need at least one dual-edge of some branch-triangle since no dual-cycle is formed by dual-edges of disjoint dual-trees. Let $\widetilde{e}$ and $\widetilde{t}$ be respectively such a dual-edge and the branch-triangle with this dual-edge. Note also that $\widetilde{c}$ and $\widetilde{t}$ are (the boundaries of) adjacent faces sharing dual-edge $\widetilde{e}$.

Here we overlap the original plane graph G with $\widetilde{H}$. Then by definition there exists a branch vertex in branch-triangle $\widetilde{t}$ where three connectors of $B \cup R$ are merged. Clearly, there should be one connector that starts with an edge crossing dual-edge $\widetilde{e}$. Thus, $\widetilde{c}$ contains this connector. On the other hand, there must be a frame-cycle $\widetilde{c}'$ whose inside contains the body of this connector (Definition 7.4[16]). Note that both $\widetilde{c}$ and $\widetilde{c}'$ are boundaries of some faces of $\widetilde{H}$. Therefore, $\widetilde{c}$ must be the same as $\widetilde{c}'$ since two faces cannot overlap; that is, $\widetilde{c}$ is a frame-cycle. □

Now that we have identified all faces of $\widetilde{H}$, we consider the conditions (F1) and (F3) for $\widetilde{H}$. Recall that the condition (F1) requires that each face weight is less than 1/3. Since faces are defined by either a branch-triangle or a frame-cycle, and the former case is trivial, we need to consider faces defined a frame-cycle. Note that the preprocessing step guarantees that the number of dual-vertices of $\widetilde{G}$ inside

[16]Precisely speaking, we also need to consider frame-cycles introduced in the preprocessing step; but it is easy to see that this fact remains true even after the preprocessing step.

of each frame-cycle is less than $\widetilde{n}/4$. Furthermore, for the total number $\widetilde{n}^-$ of removed dual-vertex, we will see later (in Section 9) that $\widetilde{n}^- = \widetilde{n} - O(\sqrt{n})$; hence, we have $\widetilde{n}/4 < \widetilde{n}^-/3$ for sufficiently large n, satisfying the weight requirement of (F1).

For the condition (F3), that is, for showing that $\widetilde{H}$ has $O(n/k)$ faces, we count the number of connectors and branch vertices.

Lemma 8.5. *The number of connectors is $O(n/k)$. Also, the number of branch vertices is $O(n/k)$.*

Proof. We consider a plane graph $G' = (V', E')$ defined from $B \cup R$, where V' is the set of branch vertices of $B \cup R$, and E' is the set of edges between adjacent branch vertices in $B \cup R$; thus, each edge of E' corresponds to some connector. We assume the planar embedding following that of G. Since G' is plane, it satisfies Euler's formula $|F'| + |V'| = |E'| + 2$, where F' is the set of faces of G'. Since the degree of any branch vertex is 3, we have $2|E'| = 3|V'|$. By substituting this to the above Euler's formula, we have $3|F'| + 2|E'| = 3|E'| + 6$, implying $|E'| = 3|F'| - 6 = O(|F'|)$.

Here we note that there is one to one correspondence between faces of G' and Voronoi regions. To see this, consider a graph consisting of only B edges, i.e., Voronoi boundary edges. Then the one-to-one correspondence is clear. On the other hand, adding ridge edges does not divide any Voronoi region; otherwise, the BFS dual-tree of the divided Voronoi region crosses some ridge edge, contradicting the definition of ridge edges. Hence, no new face is created by adding R edges. Thus, we have $|F'| = O(n/k)$. Therefore, we have $|E'|$, which is the number of connectors, is $O(n/k)$. Also we can bound the number of branch vertices by $O(n/k)$ because $2|E'| = 3|V'|$. □

Then the following corollary is immediate from the one-to-one correspondence respectively between branch-triangles and branch vertices and between frame-cycles and connectors.

Corollary 8.6. *The number of faces of $\widetilde{H}$ is $O(n/k)$.*

9 Floor and Ceiling Modification

In this section, we formally define the notions of floor- and ceiling-cycles, and explain their properties. We then explain Step 2.4 of our algorithm for computing floor- and ceiling-cycles and modified $\widetilde{H}$ as a final output of Step 2.

We begin with defining the notion of "core". Recall that $\mathrm{d_{nb}}(\widetilde{v})$ is the largest d such that $|\cup_{0 \le i \le d} \mathrm{L}(\widetilde{v}, i)| < k$ holds.

Definition 9.1 (Core)**.** *For any boss-vertex $\widetilde{b} \in \widetilde{I}$, let $\mathrm{d_{core}}(\widetilde{b})$ denote the largest $d \le \mathrm{d_{nb}}(\widetilde{b})$ such that $|\mathrm{L}(\widetilde{b}, d)| \le \sqrt{k}$. The* core *of $\widetilde{b}$ (denoted by $\mathrm{Core}(\widetilde{b})$) is defined by*

$$\mathrm{Core}(\widetilde{b}) \;=\; \bigcup_{0 \le i \le \mathrm{d_{core}}(\widetilde{b})} \mathrm{L}(\widetilde{b}, i).$$

We first note the following relation.

Lemma 9.1. *For any $\widetilde{b} \in \widetilde{I}$, we have $\mathrm{d_{nb}}(\widetilde{b}) - \mathrm{d_{core}}(\widetilde{b}) \le \sqrt{k}$.*

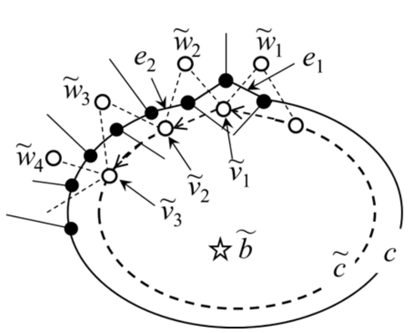
Proof. Suppose otherwise, that is, $\mathrm{d_{core}}(\widetilde{b}) + \sqrt{k} < \mathrm{d_{nb}}(\widetilde{b})$ holds. Then we have

$$\bigcup_{i=1}^{\sqrt{k}} \mathrm{L}(\widetilde{b}, \mathrm{d_{core}}(\widetilde{b}) + i) \;\subseteq\; \mathrm{N}_k(\widetilde{b}).$$

On the other hand, by definition, for any d such that $\mathrm{d_{core}}(\widetilde{b}) < d \le \mathrm{d_{nb}}(\widetilde{b})$, we have $|\mathrm{L}(\widetilde{v}, d)| > \sqrt{k}$, from which we have $|\mathrm{N}_k(\widetilde{b})| > k$, contradicting the definition of k-neighborhood. □

Consider any boss-vertex $\widetilde{b} \in \widetilde{I}$ and its core $\mathrm{Core}(\widetilde{b})$. By definition $\mathrm{Core}(\widetilde{b})$ is a subset of $\mathrm{N}_k(\widetilde{b})$. Hence, $k' := |\mathrm{Core}(\widetilde{b})| < k$; that is, the core contains less than k dual-vertices. It is also a region. Thus, its boundary is a set of disjoint cycles of G. Let c be one of such boundary cycles. We regard the subplane separated by c not containing any dual-vertex of the core as the *outside* of c. The cycle c is called a *core boundary cycle* if its outside contains at least $2\widetilde{n}/3$ dual-vertices. It may be the case that no core boundary cycle exists. In this case, by removing the core, we would have separated subgraphs

$\widetilde{G}_1, \ldots, \widetilde{G}_t$ from which we can define $\cup_{1 \le i \le t_0} \widetilde{G}_i$ and $\cup_{t_0 < i \le t} \widetilde{G}_i$ for some t_0 such that both parts have at least $(\widetilde{n}/3) - k'$ dual-vertices. Hence, the core itself is, say, a 1/4-separator of $\widetilde{G}$ for sufficiently large $k = \sqrt{n}$ and $\widetilde{n}$, and its size is $k' \le \sqrt{n}$; thus, the core itself can be used as a desired separator from which we can construct a target separator of $G_{\rm org}$ using the method stated in Lemma 4.2. Therefore, in the following we may assume that each core has a core boundary cycle. Note also that more than one core boundary cycles do not exist for each boss-vertex because the outside of such cycles are disjoint and no disjoint two sets can contain $2\widetilde{n}/3$ dual-vertices in each. Then for such a core boundary cycle, consider a set $\widetilde{c}$ of dual-vertices in Core($\widetilde{b}$) (regarded as faces of G) sharing a boundary edge with the core boundary cycle. We show in Lemma 9.2 below that a subgraph of $\widetilde{G}$ induced by $\widetilde{c}$ in fact forms a dual-cycle. This observation leads the following definition.

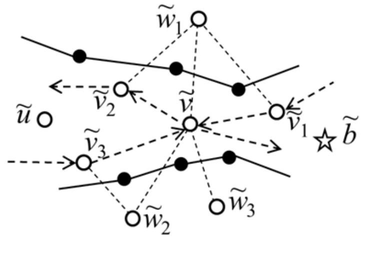


A bold solid line indicates the core boundary cycle c stated in the lemma, and some of its vertices are shown as black nodes. White nodes are some of dual-vertices that share a face boundary edge with c. In particular, dual-vertices $\widetilde{v}_1$ and $\widetilde{v}_2$ are those in Core($\widetilde{b}$) sharing boundary edges with c, and edges e_1 and e_2 are representatives of such boundary edges discussed in the proof. It is easy to see that such dual-vertices form an m.d.-cycle $\widetilde{c}$ under the anti-clockwise order that is indicated by a bold dashed line with arrows showing its direction.

Figure 21: An example of core boundary cycle and core-cycle for the proof of Lemma 9.2 (1)

Definition 9.2 (Core boundary cycle and core-cycle). *For each core* Core($\widetilde{b}$)*, its* core boundary cycle *is a cycle such that (i) it is one of the boundary cycles of the region* Core($\widetilde{b}$)*, and (ii) its outside has at least $2\widetilde{n}/3$ dual-vertices. The* core-cycle *of* Core($\widetilde{b}$) *is a directed dual-cycle induced by the set of dual-vertices in* Core($\widetilde{b}$) *sharing an edge with the core boundary cycle. The* inside *of the core-cycle is the side with the boss-vertex $\widetilde{b}$.*
Remark. *As defined above, a core boundary cycle is a cycle in G whereas a core-cycle is a dual-cycle in $\widetilde{G}$. We will use similar wording later; e.g., interior boundary cycle vs. interior-cycle, etc.*

Lemma 9.2. *Consider any boss-vertex $\widetilde{b}$ and its core* Core($\widetilde{b}$)*, and let c be its core boundary cycle. Then we can define an m.d.-cycle $\widetilde{c}$ under the anticlockwise order consisting of all dual-vertices of* Core($\widetilde{b}$) *sharing an edge with c. This $\widetilde{c}$ in fact is a dual-cycle. (We may assume that c is directed so that its inside, i.e., the left side w.r.t. the direction, contains* Core($\widetilde{b}$)*. The direction of $\widetilde{c}$ is also determined consistent with that of c, by which $\widetilde{b}$ is located the inside of $\widetilde{c}$.)*
Remark. *From the relation between c and $\widetilde{c}$, we refer $\widetilde{c}$ as the m.d.-cycle* left-adjacent *to c.*



A graph consisting of black vertices connected by solid edges is a part of the core boundary cycle c stated in the lemma. A part of the m.d.-cycle $\widetilde{c}$ is indicated by bold dashed directed edges connecting white vertices as $\widetilde{v}_1$, $\widetilde{v}$, $\widetilde{v}_2$, …, $\widetilde{v}_3$, and $\widetilde{v}$, …. Its sub m.d.-cycle $\widetilde{c}'$ considered in the proof corresponds to the part $\widetilde{v}$, $\widetilde{v}_2$, …, $\widetilde{v}_3$, and $\widetilde{v}$. Here $\widetilde{w}_1, \widetilde{w}_2$, and $\widetilde{w}_3$ are dual-vertices located in the opposite side, i.e., the outside of c from, e.g., $\widetilde{v}$.

Figure 22: An example for the proof of Lemma 9.2 (2)

Proof. Let $d_0 = {\rm d}_{\rm core}(\widetilde{b})$; hence, Core($\widetilde{b}$) $= \cup_{d \le d_0} {\rm L}(\widetilde{b}, d)$. We first show the existence of the m.d.-cycle $\widetilde{c}$ stated in the lemma (see Figure 21). Consider any edge e_1 of the boundary cycle c, and let $\widetilde{v}_1$ be a dual-vertex in Core($\widetilde{b}$) (regarded as a face of G) that has e_1 as one of its face boundary edges. Clearly,

c has edges that are not boundary edges of $\widetilde{v}_1$, and let e_2 be the first such edge from e_1 following the direction so that $\widetilde{v}_1$ is located left. Then we have some dual-vertex $\widetilde{v}_2$ in $\mathrm{Core}(\widetilde{b})$ having e_2 (as its face boundary edge). Similarly, we can find e_3 and $\widetilde{v}_3$, and so on until coming back to e_1. Let $\widetilde{c}$ be the obtained sequence of dual-vertices. Then $\widetilde{c}$ is an m.d.-cycle under the anti-clockwise order that has all dual-vertices of $\mathrm{Core}(\widetilde{b})$ in its inside. This is because there is no dual-vertex of $\mathrm{Core}(\widetilde{b})$ from $\widetilde{v}_{i-1}$ to $\widetilde{v}_{i+1}$ anti-clockwise among all dual-vertices adjacent to $\widetilde{v}_i$. In fact, it is also easy to see that all $\widetilde{v}_i$'s are in $\mathrm{L}(\widetilde{b}, d_0)$ facing $\widetilde{w}_j$'s in $\mathrm{L}(\widetilde{b}, d_0+1)$.

We then prove that $\widetilde{c}$ is indeed a dual-cycle (see Figure 22). For this, it suffices to show that the degree of every dual-vertices of $\widetilde{c}$ is two in $\widetilde{c}$. Suppose otherwise; that is, we had a dual-vertex $\widetilde{v}$ connecting more than two dual-vertices in $\widetilde{c}$. Hence, the m.d.-cycle $\widetilde{c}$ would be expressed as a sequence $(\widetilde{v}_1, \widetilde{v}, \widetilde{v}_2, \ldots, \widetilde{v}_3, \widetilde{v}, \widetilde{v}_4, \ldots)$. (It may be the case that $\widetilde{v}_4$ is the same as $\widetilde{v}_3$.) We can split this into two subsequences: $(\widetilde{v}, \widetilde{v}_2, \ldots, \widetilde{v}_3)$ and the remaining one, i.e., $(\widetilde{v}_1, \widetilde{v}, \widetilde{v}_4, \ldots)$. Note that both are m.d.-cycles, and clearly, one of them has the boss-vertex $\widetilde{b}$ in its inside while the other one does not. Without losing generality, we assume that $\widetilde{c}' = (\widetilde{v}, \widetilde{v}_2, \ldots, \widetilde{v}_3)$ does not have the boss-vertex $\widetilde{b}$ in its inside. Now consider, e.g., the dual-vertex $\widetilde{v}_2$. Since $\widetilde{v}_2 \in \mathrm{Core}(\widetilde{b})$, its distance from $\widetilde{b}$ is $\le d_0$; hence, there must be some dual-vertex $\widetilde{u} \in \mathrm{L}(\widetilde{b}, d_0-1)$ inside of $\widetilde{c}'$ adjacent to $\widetilde{v}_2$. Thus, there must be a path of length $d_0 - 1$ from $\widetilde{b}$ to $\widetilde{u}$. On the other hand, this path should go through $\widetilde{v}$ because $\widetilde{b}$ is in the inside of the other m.d.-cycle. But then the distance of $\widetilde{v}$ becomes less than d_0, contradicting the assumption that $\widetilde{v}$ shares a boundary edge with c, which means that there exists a dual-vertex $\widetilde{w}_1$ that shares this boundary edge with $\widetilde{v}$ in the outside of $\mathrm{Core}(\widetilde{b})$, i.e., in $\mathrm{L}(\widetilde{b}, d_0+1)$. □

We state here the following size bounds, which are immediate from the definition.

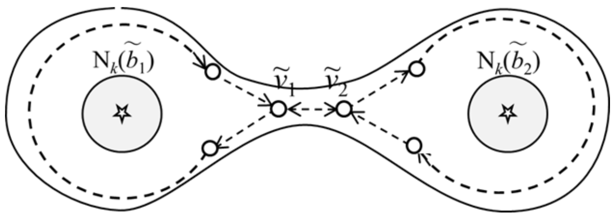
Lemma 9.3. *Each core-cycle has at most $\sqrt{k}$ dual-vertices and at most $\widetilde{n}/3$ dual-vertices in its inside.*

We define the notions of neck, floor, and ceiling. For this we introduce the notion of "level." The main difference from the intuitive explanation in Outline section is that it is defined in terms of the distance from *all* k-neighborhoods (instead of a single boss-vertex). For any $\ell \ge 1$, let $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$ denote a set of dual-vertices $\widetilde{v}$ whose distance from its nearest k-neighborhood in $\{\mathrm{N}_k(\widetilde{b})\}_{\widetilde{b}\in\widetilde{I}}$ is ℓ. More formally, it is defined by

$$\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell) \;=\; \left\{\, \widetilde{v} \,|\, \mathrm{dist}(\widetilde{v}, \widetilde{v}_{\mathrm{nrst}}) = \ell, \text{ where } \widetilde{v}_{\mathrm{nrst}} = \mathrm{nrst}_{\widetilde{v}}(\mathrm{N}_k(\mathrm{boss}(\widetilde{v})))\,\right\}$$

Clearly, a family $\{\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)\}_{\ell\ge1}$ is a partition of $\widetilde{V}' := \widetilde{V} \setminus \cup_{\widetilde{b}\in\widetilde{I}} \mathrm{N}_k(\widetilde{b})$. For any dual-vertex $\widetilde{v} \in \widetilde{V}'$ (resp., any set $U \subseteq \widetilde{V}'$ of dual-vertices), the *level* of $\widetilde{v}$ (resp., U) is ℓ such that $\widetilde{v} \in \widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$ holds (resp., $U \subseteq \widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$ holds). We use $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$ also to denote a subgraph of $\widetilde{G}$ induced by $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$, which we call a *sliced graph* of level ℓ.

We first note a property of sliced graphs that allows us to define interior and exterior boundary cycles.


$\mathrm{N}_k(\widetilde{b}_1)$
$\widetilde{v}_1$
$\widetilde{v}_2$
$\mathrm{N}_k(\widetilde{b}_2)$

An example of a connected component C of some level graph $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$. A solid line indicates its exterior boundary cycle c, and a dashed line indicates the m.d.-cycle $\widetilde{c}$ in C right-adjacent to c; note that the edges of $\widetilde{c}$ are traversed under the clockwise order. The level of both $\widetilde{v}_1$ and $\widetilde{v}_2$ (and in fact all dual-vertices of $\widetilde{c}$) is ℓ, which is determined by the distance from either $\mathrm{N}_k(\widetilde{b}_1)$ (e.g., for $\widetilde{v}_1$) or $\mathrm{N}_k(\widetilde{b}_2)$ (e.g., for $\widetilde{v}_2$).

Figure 23: Exterior boundary and its right-adjacent m.d.-cycle

Lemma 9.4 (Interior and exterior boundary cycles)**.** *For any $\ell \ge 1$, consider any connected component C of sliced graph $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$ with a dual-vertex adjacent to some dual-vertex of level $\ell+1$. Then C has at* *least two disjoint boundary cycles:* interior boundary cycle *and* exterior boundary cycle *(see the proof for their definition). Furthermore, for these boundary cycles, we can define m.d.-cycles like core-cycles. More precisely, consider any one of such boundary cycles c. Then a set of dual-vertices of C sharing a boundary edge with c forms an m.d.-cycle $\widetilde{c}$ under the clockwise order.*

Remark. *It may be the case that the above $\widetilde{c}$ is not a simple dual-cycle; see Figure 23. (*Cf. *A core-cycle is a simple dual-cycle.) Below we will refer this $\widetilde{c}$ as the m.d.-cycle in C* right-adjacent *to c. On the other hand, a cycle c is called the* base *of $\widetilde{c}$.*

Proof. By definition C is a region, and hence, its boundary is a disjoint union of cycles of G. Since C consists of only level ℓ dual-vertices, each boundary cycle edge is an edge between a pair of dual-vertices of level either $\ell - 1$ and ℓ or ℓ and $\ell + 1$. Let us call the former one an *interior edge* and the latter one an *exterior edge*. Note that an interior edge must exist to define level ℓ dual-vertices of C, and that an exterior edge must exist because of the assumption of the lemma. We show that no boundary cycle has a vertex incident with both interior and exterior boundary edges. Suppose otherwise; that is, some vertex v exists that is incident with both interior and exterior boundary edges. Since G is three regular, this v is incident to three faces, one corresponding to a level ℓ dual-vertex (of C), one corresponding to a level $\ell - 1$ dual-vertex, and one corresponding to a level $\ell + 1$ dual-vertex. But this contradicts the definition of level. Thus, C, as a region, has at least the following two disjoint boundary cycles: an *interior boundary cycle* consisting of interior edges and an *exterior boundary cycle* consisting of exterior edges.

Now consider any one of such boundary cycles of C, and denote it by c. Then by the same argument as the proof of Lemma 9.2, we can define an m.d.-cycle $\widetilde{c}$ consisting of all and only dual-vertices of C that share boundary edges with c. (We may assume that the direction of $\widetilde{c}$ is determined so that vertices of c are located its inside (Figure 24), which is different from core-cycles. From this difference, we need to consider the clockwise order when identifying an m.d.-cycle right-adjacent to a given interior/exterior boundary cycle.) □

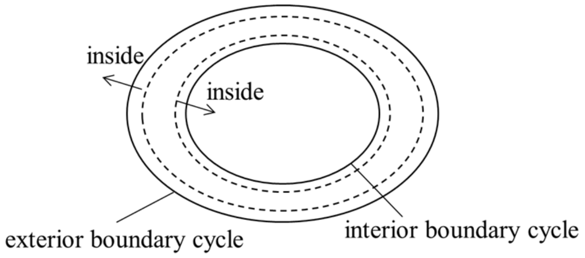


Figure 24: Interior and exterior boundary cycles and their right-adjacent m.d.-cycles

Based on this lemma, we introduce the notion of interior- and exterior-cycles. (Recall here that every m.d.-cycle is a collection of dual-cycles incident or connected by a line part.)

Definition 9.3 (Interior- and exterior-cycles)**.** *Let c be any interior (resp., exterior) boundary cycle of some connected component C of $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$. Let $\widetilde{c}$ be the m.d.-cycle in C right-adjacent to c, and let $\widetilde{D}_{\widetilde{c}}$ be the set of dual-cycles obtained from $\widetilde{c}$ by removing all its line parts. An* interior-cycle *(resp.,* exterior-cycle*) is a dual-cycle in $\widetilde{D}_{\widetilde{c}}$.*

Remark. *Note that $\widetilde{D}_{\widetilde{c}}$ is defined for each boundary cycle. There may be more than one interior (resp., exterior) boundary cycles, but we focus on one boundary cycle c for defining $\widetilde{c}$. We assume that each interior- and exterior-cycle is directed following its base m.d.-cycle $\widetilde{c}$; the notions of inside/outside are the same as $\widetilde{c}$ of Figure 24.*

We show some basic properties of interior- and exterior-cycles. Below we say that two dual-cycles *have an overlap* if each has a dual-vertex of the other dual-cycle in its inside (or intuitively, they intersect at least at two dual-vertices).

Lemma 9.5. *There is no overlap among interior-, exterior-, and core-cycles. The inside of each interior-cycle contains at least one boss-vertex.*

Proof. Recall that every interior- or exterior-cycle is a subgraph of sliced graph $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$ for some ℓ, and that $\{\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)\}_{\ell\geq 1}$ is a partition of $\widetilde{V} \setminus \cup_{\widetilde{b}\in\widetilde{I}}\mathrm{N}_k(\widetilde{b})$. On the other hand, every core-cycle is in $\mathrm{N}_k(\widetilde{b})$ for some $\widetilde{b}$. Hence, in order to show that there is no overlap among interior-, exterior-, and core-cycles, it suffices to show that any two m.d.-cycles of level ℓ that could become interior- or exterior-cycles have no overlap. Suppose otherwise; that is, suppose that we have m.d.-cycles $\widetilde{c}$ and $\widetilde{c}'$ that are right-adjacent to some interior or exterior boundary cycles, say, c and c' respectively of some connected component C of some sliced graph $\widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell)$. (Note that $\widetilde{c}$ and $\widetilde{c}'$ must be in the same connected component because they share some dual-vertex.) But by the construction of $\widetilde{c}$ and $\widetilde{c}'$, c and c' must have a common vertex, contradicting that all boundary cycles of any region are disjoint.

The second claim of the lemma follows from the following two facts: (i) every interior-cycle $\widetilde{c}$ of level ℓ has a dual-vertex $\widetilde{v}$ of level $\ell-1$ in its inside, and (ii) there should be $\mathrm{N}_k(\widetilde{b})$ witnessing the level of $\widetilde{v}$ and there should be a dual-path from $\widetilde{v}$ to the boss-vertex $\widetilde{b}$ that does not intersect with $\widetilde{c}$. □

Lemma 9.6. *Consider any boss-vertex $\widetilde{b} \in \widetilde{I}$, dual-vertex $\widetilde{u} \in \mathrm{Vr}(\widetilde{b})$, and the BFS dual-path $\widetilde{p}$ from $\widetilde{b}$ to $\widetilde{u}$. Let $(\widetilde{v}_1, \dots, \widetilde{v}_t)$ be the tail subpath of $\widetilde{p}$. where $\widetilde{v}_t = \widetilde{u}$, and $\widetilde{v}_1$ is the first dual-vertex on $\widetilde{p}$ that is not in $\mathrm{N}_k(\widetilde{b})$. For each i, $1 \leq i < t$, there exist an interior-cycle (resp., an exterior-cycle) that contains $\widetilde{v}_i$ and that has $\widetilde{b}$ in its one side and $\widetilde{v}_t$ in the other side.*

Proof. We fix any i, $1 \leq i < t$, and let C be a connected component of a sliced graph containing $\widetilde{v}_i$, and let ℓ be the level of $\widetilde{v}_i$ and C. We first show that $\widetilde{v}_i$ appears in one of the exterior-cycles of C that satisfies the condition of the lemma. (The argument for the interior-cycles of C is similar.)

Note that the level of $\widetilde{v}_{i+1}$ is $\ell + 1$; hence, an edge e_0 crossing the dual-edge $\{\widetilde{v}_i, \widetilde{v}_{i+1}\}$ is an edge of some exterior boundary cycle c of C. We focus on this cycle; see Figure 25. Consider the enumeration of dual-vertices adjacent to $\widetilde{v}_i$ anti-clockwise from $\widetilde{v}_{i+1}$ to $\widetilde{v}_{i-1}$. Here we note that the level of the dual-vertices in the enumeration is either $\ell-1$, ℓ, or $\ell+1$, and that $(*)$ the levels of every consecutive pair of dual-vertices differs at most one because they are connected since $\widetilde{G}$ is triangulated. Let $\widetilde{x}_{j-1}$ is the last dual-vertex in the enumeration that is of level $\ell+1$ and the edge e_k crossing $\widetilde{v}_i$ and $\widetilde{x}_{j-1}$ is an edge of the exterior boundary cycle c. Such $\widetilde{x}_{j-1}$ exists because (i) there exists a dual-vertex of level $\ell+1$ in the adjacent dual-vertex enumeration whose boundary edge between $\widetilde{v}_i$ is an edge of c because at least we have $\widetilde{v}_{j+1}$ that satisfies this condition, and (ii) there should be the last one from the fact $(*)$ stated above. Now from the property of $\widetilde{x}_{j-1}$ and also from $(*)$ it follows that the next one $\widetilde{x}_j$ in the adjacent dual-vertex enumeration is of level ℓ. Then the boundary edge e_{k+1} between $\widetilde{x}_{j-1}$ and $\widetilde{x}_j$ is a part of *some* exterior boundary cycle, which in fact must be the cycle c. This is because (i) both e_k and e_{k+1} must share the unique vertex w in the triangle $\widetilde{v}_i$, $\widetilde{x}_{j-1}$, and $\widetilde{x}_j$ as their end point, and (ii) the third edge e' incident to w is not an exterior boundary edge.

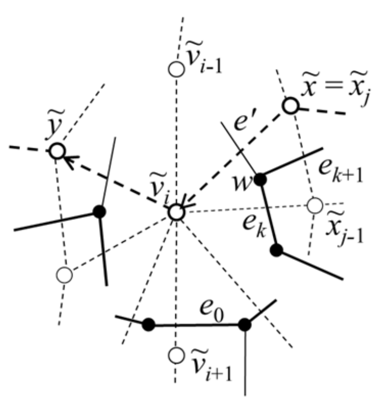
Below let us denote $\widetilde{x}_i$ simply by $\widetilde{x}$. We can symmetrically show the existence of a dual-vertex $\widetilde{y}$ with the same property in the clockwise enumeration of adjacent dual-vertices of $\widetilde{v}_i$ from $\widetilde{v}_{i+1}$ to $\widetilde{v}_{i-1}$.

Consider the m.d.-cycle $\widetilde{c}$ right adjacent to c. Note first that a directed dual-edge $(\widetilde{x}, \widetilde{v}_i)$ appears in $\widetilde{c}$. This is because (i) a consecutive pair of boundary edges e_{k+1} and e_k are shared by $\widetilde{x}$ and $\widetilde{v}_i$ respectively, and (ii) the direction is consistent with the requirement that c is located inside of $\widetilde{c}$. A similar argument shows that $(\widetilde{v}_i, \widetilde{y})$ appears in $\widetilde{c}$. Our goal is to show that this pair of dual-edges $(\widetilde{x}, \widetilde{v}_i)$ and $(\widetilde{v}_i, \widetilde{y})$ appear in one of the dual-cycles of $\widetilde{D}_{\widetilde{c}}$, that is, they are part of some exterior-cycle $\widetilde{c}'$ (see Definition 9.3 for $\widetilde{D}_{\widetilde{c}}$). This goal clearly suffices for the lemma because $\widetilde{v}_{i-1}$ (and hence $\widetilde{b}$) is in one side of $\widetilde{c}'$ while $\widetilde{v}_{i+1}$ (and hence $\widetilde{v}_t$) is in the other side. (Note that $\widetilde{v}_i$ is the unique point where the exterior-cycle $\widetilde{c}'$ crosses the BFS dual-path from $\widetilde{b}$ to $u = \widetilde{v}_t$.)

For our goal, it suffices to show that $(*)$ after $(\widetilde{v}_i, \widetilde{y})$ in $\widetilde{c}$, the dual-edge $(\widetilde{x}, \widetilde{v}_i)$ must be used when the m.d.-cycle $\widetilde{c}$ comes back to $\widetilde{v}_i$ next, that is, no other dual-edge to $\widetilde{v}_i$ is used. Once this is proved, it is clear that the subsequence of $\widetilde{c}$ from $(\widetilde{v}_i, \widetilde{y})$ to $(\widetilde{x}, \widetilde{v}_i)$ has a directed dual-cycle having both $(\widetilde{v}_i, \widetilde{y})$ and $(\widetilde{x}, \widetilde{v}_i)$ because $\widetilde{v}_i$ appears in only these two dual-edges in this subsequence.

Now we show $(*)$. For this, consider the m.d.-cycle $\widetilde{c}$ starting from $\widetilde{v}_i$ with dual-edge $(\widetilde{v}_i, \widetilde{y})$. Also we consider an upper part (i.e., the part from $(\widetilde{x}, \widetilde{v}_i)$ to $(\widetilde{v}_i, \widetilde{y})$ anti-clockwise) and a lower part (i.e., the part from $(\widetilde{v}_i, \widetilde{y})$ to $(\widetilde{x}, \widetilde{v}_i)$ anti-clockwise), and show that no dual-edge to $\widetilde{v}_i$ in these parts is used for coming back to $\widetilde{v}_i$ for the first time. For the upper part, we can in fact easily show that there is no directed dual-edge of $\widetilde{c}$ to $\widetilde{v}_i$ in this part. Suppose otherwise and that some directed dual-edge of $\widetilde{c}$ coming into $\widetilde{v}_i$. This should come from a dual-vertex adjacent to $\widetilde{v}_i$ that is of level ℓ sharing an edge of c that is a boundary edge to a dual-vertex of level $\ell+1$. Clearly, this contradicts to the choice of $\widetilde{x}$ and $\widetilde{y}$. Consider next the lower part. Suppose that the m.d.-cycle $\widetilde{c}$, after $(\widetilde{v}_i, \widetilde{y})$, comes back to

$\widetilde{v}_i$ for the first time from the lower part by some dual-edge $(\widetilde{z}, \widetilde{v}_i)$. Then after $(\widetilde{z}, \widetilde{v}_i)$, all dual-edges going out from $\widetilde{v}_i$ or coming back to $\widetilde{v}_i$ must be located between $(\widetilde{z}, \widetilde{v}_i)$ and $(\widetilde{v}_i, \widetilde{y})$ clockwise because the m.d.-cycle $\widetilde{c}$ is defined under the clockwise order; otherwise, some subsequence of $\widetilde{c}$ must cross the subsequence of $\widetilde{c}$ from $\widetilde{y}$ to $\widetilde{z}$, contradicting the noncrossing property of m.d.-cycle (Figure 11). Then $(\widetilde{x}, \widetilde{v}_i)$ cannot appear in $\widetilde{c}$, a contradiction. □



$\widetilde{v}_{i\text{-}1}$
$\widetilde{x} = \widetilde{x}_j$
$\widetilde{y}$
e'
e_{k+1}
w
$\widetilde{v}_i$
e_k
$\widetilde{x}_{j\text{-}1}$
e_0
$\widetilde{v}_{i+1}$

A bold line (with black nodes) indicates a part of the exterior boundary cycle c focused in the proof. In the proof, for the anti-clockwise enumeration of dual-vertices adjacent to $\widetilde{v}_i$ from $\widetilde{v}_{i+1}$ to $\widetilde{v}_{i-1}$, we consider $\widetilde{x}_{j-1}$, the last one in the enumeration that is of level $\ell + 1$ and whose boundary edge, i.e., e_k, is a part of c. We then show that directed dual-edges $(\widetilde{x}, \widetilde{v}_i)$ and $(\widetilde{v}_i, \widetilde{y})$ are part of the m.d.-cycle $\widetilde{c}$ right adjacent to c, and in fact they are part of the same dual-cycle component of $\widetilde{c}$, i.e., an exterior-cycle defined from $\widetilde{c}$.

Figure 25: An example for the proof of Lemma 9.6

Finally, we define the notion of neck, floor, and ceiling. Consider any interior- or exterior-cycle. It is called a *neck* if it has at most $\sqrt{k}$ dual-vertices, and an interior (resp., exterior) neck is a neck interior-cycle (resp., neck exterior-cycle). A neck is called *biased* if it has less than $\widetilde{n}/3$ dual-vertices in its inside. Note that if there is any neck that has at least $\widetilde{n}/3$ dual-vertices in both inside and outside, then we can use the neck itself as a 1/4-separator of $\widetilde{G}$ (when n is sufficiently large) from which we can construct a target separator for G_{org}. Therefore, in the following discussion, we assume that a neck is biased or it has less than $\widetilde{n}/3$ dual-vertices in its outside.

Roughly speaking, a *floor-cycle* (resp., *ceiling-cycle*) is a biased neck interior-cycle (resp., exterior-cycle). But some more conditions are needed. As explained in Outline section, we modify the frame graph $\widetilde{H}$ by covering each boss-vertex by a floor-cycle and a branch-triangle by a ceiling-cycle in order for reducing the length of each dual-path from a boss-vertex to (a dual-vertex of) a branch-triangle. Thus, we only need the “largest” one covering each boss-vertex or branch-triangle. Also we do not need an exterior-cycle that has no intersection with any branch-triangle as a ceiling-cycle. On the other hand, it may be the case where some boss-vertex is contained in no floor-cycle. For such boss-vertices, we would use their core-cycles as floor-cycles. From these considerations, we define the notion of floor- and ceiling-cycles as follows.

Definition 9.4 (Floor- and ceiling-cycles). *A* floor-cycle *is a biased neck interior-cycle that is not contained in the inside of any other biased neck interior-cycle. For any boss-vertex $\widetilde{b}$ that is contained in no floor-cycle, we regard the core-cycle of* Core($\widetilde{b}$) *also as a* floor-cycle. *A* ceiling-cycle *is a biased neck exterior-cycle that is not contained in the inside of any other biased neck cycle-cycle and that has at least one dual-vertex of some branch-triangle.*

Remark. *When a floor-cycle (resp., ceiling-cycle) is a special case of interior-cycles (resp., exterior-cycles), its inside/outside is the same as the corresponding interior-cycles (resp., exterior-cycles). Similarly, for a floor-cycle defined as a core-cycle, its inside/outside follow those of the core-cycle.*

We show that floor- and ceiling-cycles have desired properties. First we state the following size bounds, which are immediate from the definition.

Lemma 9.7. *Each floor- and ceiling-cycle has at most $\sqrt{k}$ dual-vertices and less than $\widetilde{n}/3$ dual-vertices in its inside.*

Next one is the key property of floor- and ceiling-cycles, which is the main reason of introducing these structures. (One can find a similar argument in the proof of, e.g., Lemma 3.3 of [10].)

Lemma 9.8. *Consider any pre-frame-cycle (after the preprocessing), and let $\widetilde{p}$ be any dual-path between its boss-vertex $\widetilde{b}$ and the "next" dual-vertex $\widetilde{u}$ of a branch-triangle used in the pre-frame-cycle* [17]*. There exist one floor-cycle $\widetilde{c}_{\mathrm{f}}$ and at most one ceiling-cycle $\widetilde{c}_{\mathrm{c}}$ crossing $\widetilde{p}$. Let $\widetilde{u}_{\mathrm{f}}$ and $\widetilde{u}_{\mathrm{c}}$ denote dual-vertices at the crossing point of respectively $\widetilde{p}$ and $\widetilde{c}_{\mathrm{f}}$, and $\widetilde{p}$ and $\widetilde{c}_{\mathrm{c}}$. Then $\widetilde{p}$ is separated into at most three parts $\widetilde{p}_1, \widetilde{p}_2$, and $\widetilde{p}_3$ by $\widetilde{c}_{\mathrm{f}}$ and $\widetilde{c}_{\mathrm{c}}$: that is, dual-path $\widetilde{p}_1$ between $\widetilde{b}$ and $\widetilde{u}_{\mathrm{f}}$, dual-path $\widetilde{p}_3$ between $\widetilde{u}$ and $\widetilde{u}_{\mathrm{c}}$, and the remaining dual-path $\widetilde{p}_2$. (In the case where $\widetilde{c}_{\mathrm{c}}$ does not exist, $\widetilde{p}_2$ is simply a dual-path defined by $\widetilde{p} \setminus \widetilde{p}_1$.) Furthermore, the length of $\widetilde{p}_2$ is less than $4\sqrt{k}$.*

Proof. Recall that $\widetilde{p}$ is a dual-path of the BFS dual-tree from $\widetilde{b}$. The existence of the unique floor-cycle $\widetilde{c}_{\mathrm{f}}$ and at most one ceiling-cycle $\widetilde{c}_{\mathrm{c}}$ follows from Lemmas 9.5 and 9.6, and Definition 9.4. Furthermore, it is easy to see that all dual-vertices of $\widetilde{p}$ between $\widetilde{b}$ and $\widetilde{u}_{\mathrm{f}}$ (except $\widetilde{u}_{\mathrm{f}}$ itself) are inside of the floor-cycle $\widetilde{c}_{\mathrm{f}}$ and that all dual-vertices of $\widetilde{p}$ between $\widetilde{u}_{\mathrm{c}}$ (except $\widetilde{u}_{\mathrm{c}}$ itself) and $\widetilde{u}$ (if they exist) are inside of the ceiling-cycle $\widetilde{c}_{\mathrm{f}}$. Thus, $\widetilde{p}$ is divided into at most three parts $\widetilde{p}_1, \widetilde{p}_2$, and $\widetilde{p}_3$ as stated in the lemma, and $\widetilde{p}_2$ is the part of $\widetilde{p}$ located outside of both $\widetilde{c}_{\mathrm{f}}$ and $\widetilde{c}_{\mathrm{c}}$.

We show that the length of $\widetilde{p}_2$ is less than $4\sqrt{k}$. For our discussion, let us regard $\widetilde{p}_2$ as a *directed* dual-path based on the distance from $\widetilde{b}$. Consider first the head part of $\widetilde{p}_2$ consisting of dual-vertices in $\mathrm{N}_k(\widetilde{b})$. This part exists only if the floor-cycle $\widetilde{c}_{\mathrm{f}}$ is defined by a core-cycle, and the length of this part is by definition at most $\mathrm{d}_{\mathrm{nb}}(\widetilde{b}) - \mathrm{d}_{\mathrm{core}}(\widetilde{b})$ that is bounded by $\sqrt{k}$ (Lemma 9.1). Thus, the length of this head part of $\widetilde{p}_2$ is either 0 or at most $\sqrt{k}$.

Now consider the remaining tail part of $\widetilde{p}_2$, and let $\widetilde{v}_1, \widetilde{v}_2, \ldots, \widetilde{v}_t$ be the enumeration of dual-vertices of this part following the direction of $\widetilde{p}_2$. For the lemma, it suffices to show that the length of this part of $\widetilde{p}_2$, i.e., t is less than $3\sqrt{k}$. Assume to the contrary that $t \geq 3\sqrt{k}$.

Considering the fact that $\widetilde{v}_1, \widetilde{v}_2, \ldots, \widetilde{v}_t$ are all in $\mathrm{Vr}(\widetilde{b}) \setminus \mathrm{N}_k(\widetilde{b})$ and also on one dual-path of the BFS dual-tree from $\widetilde{b}$, we have $\widetilde{v}_i \in \widetilde{\mathrm{L}}_{\mathrm{nb}}(\ell_1 + (i-1))$ for each i, $1 \leq i \leq t$ where ℓ_1 is the level of $\widetilde{v}_1$, which fact will be referred as $(*)$ later.

Consider any i, $1 \leq i \leq t$. Lemma 9.6 guarantees the existence of an interior-cycle $\widetilde{c}_i^{\mathrm{in}}$ and an exterior-cycle $\widetilde{c}_i^{\mathrm{ex}}$ containing $\widetilde{v}_i$ (where $\widetilde{c}_i^{\mathrm{in}}$ and $\widetilde{c}_i^{\mathrm{ex}}$ might be the same dual-cycle). We argue that either $\widetilde{c}_i^{\mathrm{in}}$ or $\widetilde{c}_i^{\mathrm{ex}}$ is not a neck; that is, $|\widetilde{c}_i^{\mathrm{in}}| > \sqrt{k}$ or $|\widetilde{c}_i^{\mathrm{ex}}| > \sqrt{k}$. Note first that $\widetilde{c}_i^{\mathrm{in}}$ (resp., $\widetilde{c}_i^{\mathrm{ex}}$) contains $\widetilde{c}_{\mathrm{f}}$ (resp., $\widetilde{c}_{\mathrm{c}}$ if it exists) in its inside (Lemmas 9.5 and 9.6). Hence, neither $\widetilde{c}_i^{\mathrm{in}}$ nor $\widetilde{c}_i^{\mathrm{ex}}$ is a biased neck (Definition 9.4). Next assume that both $\widetilde{c}_i^{\mathrm{in}}$ and $\widetilde{c}_i^{\mathrm{ex}}$ are necks. Since $\widetilde{c}_i^{\mathrm{in}}$ is not biased, $\widetilde{c}_i^{\mathrm{in}}$ would have less than $\widetilde{n}/3$ dual-vertices in its outside. On the other hand, the inside of $\widetilde{c}_i^{\mathrm{ex}}$ is contained in the outside of $\widetilde{c}_i^{\mathrm{in}}$ (Figure 24). Hence, the inside of $\widetilde{c}_i^{\mathrm{ex}}$ would have less than $\widetilde{n}/3$ dual-vertices, that is, $\widetilde{c}_i^{\mathrm{ex}}$ is biased; a contradiction. Therefore, either $\widetilde{c}_i^{\mathrm{in}}$ or $\widetilde{c}_i^{\mathrm{ex}}$ is not a neck; let $\widetilde{c}_i$ denote this non-neck dual-cycle. We note here that $|\widetilde{c}_i| > \sqrt{k}$.

Now consider any i, $1.5\sqrt{k} \leq i \leq 2.5\sqrt{k}$, and let $\widetilde{C}_i$ be a set of $\sqrt{k}+1$ dual-vertices of $\widetilde{c}_i$ that are nearest to $\widetilde{v}_i$ (including $\widetilde{v}_i$ itself). (Here we can use the fact that $|\widetilde{c}_i| > \sqrt{k}$ to ensure the existence of $\widetilde{C}_i$.) Let $\widetilde{C} = \cup_{1.5\sqrt{k} \leq i \leq 2.5\sqrt{k}} \widetilde{C}_i$. Then we have $|\widetilde{C}| > k$. Note also that the distance between any dual-vertex of $\widetilde{C}_i$ and $\widetilde{v}_i$ is at most $\sqrt{k}$. (Recall that $\widetilde{c}_i$ is a dual-cycle.) Here consider a set $\widetilde{B}$ of dual-vertices whose distance from $\widetilde{v}_{2\sqrt{k}}$ is at most $1.5\sqrt{k}$. Then we have $\widetilde{B} \supset \widetilde{C}$, and hence we have $|\widetilde{B}| > k$. Note here that $\mathrm{N}_k^+(\widetilde{b})$ is the nearest k-neighborhood$^+$ of $\widetilde{v}_{2\sqrt{k}}$ and the distance between $\widetilde{v}_{2\sqrt{k}}$ and $\mathrm{N}_k^+(\widetilde{b})$ is at least $2\sqrt{k}$ (from $(*)$ mentioned above). Hence, no dual-vertex of $\widetilde{B}$ belongs to $\mathrm{N}_k^+(\widetilde{b}')$ for any $\widetilde{b}' \in \widetilde{I}$; because otherwise, the distance of $\widetilde{v}_{2\sqrt{k}}$ from $\mathrm{N}_k^+(\widetilde{b}')$ would be at most $1.5\sqrt{k}$, contradicting that $\widetilde{b}$ is the boss-vertex of $\widetilde{v}_{2\sqrt{k}}$. That is, $\mathrm{N}_k^+(\widetilde{v}_{2\sqrt{k}})$ has no intersection with $\cup_{\widetilde{b}' \in \widetilde{I}} \mathrm{N}_k^+(\widetilde{b}')$, contradicting the choice of $\widetilde{I}$. □

Finally, we define the notion of "modified frame-graph" for our final graph to which Miller's algorithm is applicable.

Definition 9.5 (Modified frame-graph)**.** *Let $\widetilde{H}$ be the frame-graph defined from $\widetilde{G}$ in the previous section. Let $\widetilde{F}$ and $\widetilde{C}$ be respectively a set of floor-cycles and ceiling-cycles having at least one dual-vertex of $\widetilde{H}$ in their insides. We let $\widetilde{E}_1'$ be the set of dual-edges that appear in some cycle in $\widetilde{F} \cup \widetilde{C}$ and $\widetilde{E}_2'$ be the set of dual-edges of $\widetilde{H}$ that are not in the inside of any cycle of $\widetilde{F} \cup \widetilde{C}$. A graph $\widetilde{H}' = (\widetilde{U}', \widetilde{D}')$ is a* modified frame-graph*, where $\widetilde{D}' = \widetilde{E}_1' \cup \widetilde{E}_2'$ and $\widetilde{U}'$ is the set of all dual-vertices that are end points of dual-edges of $\widetilde{D}'$.*

[17]More precisely, the "next" dual-vertex $\widetilde{u}$ is the first/last dual-vertex of a branch-triangle from/to the boss-vertex appearing in the frame m.d.-cycle following its direction.

Note that faces of $\widetilde{H}'$ are defined by one of the following boundary dual-cycles: (i) a floor-cycle, (ii) a ceiling-cycle, (iii) a branch-triangle, or (iv) a frame-cycle of $\widetilde{H}$ modified by floor- and ceiling-cycles. We call these boundary dual-cycles *$\widetilde{H}'$-face-cycles.* Based on this observation, we define weights of faces of $\widetilde{H}'$ in the same way as $\widetilde{H}$.

Definition 9.6. *For each face of a modified frame-graph $\widetilde{H}'$, its weight is the number of dual-vertices of $\widetilde{G}$ located in the face (i.e., in the inside of its $\widetilde{H}'$-face-cycle) divided by the total number of dual-vertices removed from $\widetilde{G}$.*

We show that the modified frame graph $\widetilde{H}'$ satisfies the conditions (F1) $\sim$ (F4).

First the condition (F1) follows from the definition of a frame-graph and modified frame-graph. In particular, the condition that the weight of each face is less than 1/3 follows from the fact that $\widetilde{H}$ satisfies this condition. Next consider the condition (F3); that is, the number of faces is bounded by $O(\widetilde{n}/k)$. Again this follows easily from the fact that $\widetilde{H}$ satisfies this condition. As mentioned above, new faces introduced by our modification are those defined by floor- or ceiling-cycles. By definition, the number of these dual-cycles are bounded by either the number of boss-vertices or that of (dual-vertices of) branch-triangles, which is bounded by $O(\widetilde{n}/k)$. Note that a face defined a frame-cycle of $\widetilde{H}$ could be divided by ceiling-cycles; but it is easy to see that each face is divided at most some constant number of faces because the number of floor- and ceiling-cycles overlapping each frame-cycle is constant, say, at most six. From these observations, we can bound the number of faces of $\widetilde{H}'$ by $O(\widetilde{n}/k)$.

We argue that $\widetilde{H}'$ keeps the condition (F2) as follows.

Lemma 9.9. *$\widetilde{H}'$ is 2-connected.*

Proof. For any two distinct dual-vertices $\widetilde{u}$ and $\widetilde{v}$ in $\widetilde{U}'$, we show that there exist two vertex disjoint dual-paths between $\widetilde{u}$ and $\widetilde{v}$. If $\widetilde{u}$ and $\widetilde{v}$ are located on the same $\widetilde{H}'$-face-cycle, it is obvious. Assume that $\widetilde{u}$ and $\widetilde{v}$ are in $\widetilde{H}$ and they survived the modification; namely, they are on branch-triangles or frame-cycles of $\widetilde{H}$ remained after the modification of Definition 9.6. There are two vertex disjoint dual-paths $\widetilde{p}_{\mathrm{L}}$ and $\widetilde{p}_{\mathrm{R}}$ in $\widetilde{H}$ since $\widetilde{H}$ is 2-connected (Lemma 8.1). In $\widetilde{H}'$, some floor- and ceiling-cycles are introduced. When some of these dual-cycles touches only one of $\widetilde{p}_{\mathrm{L}}$ and $\widetilde{p}_{\mathrm{R}}$, then it is obvious that two disjoint dual-paths exist in $\widetilde{H}'$. Suppose that a floor- or ceiling-cycle $\widetilde{c}$ touches both $\widetilde{p}_{\mathrm{L}}$ and $\widetilde{p}_{\mathrm{R}}$. Let $\widetilde{u}_{\mathrm{L}}$ and $\widetilde{v}_{\mathrm{L}}$ (resp., $\widetilde{u}_{\mathrm{R}}$ and $\widetilde{v}_{\mathrm{R}}$) respectively denote the first and the last dual-vertices on $\widetilde{p}_{\mathrm{L}}$ (resp., $\widetilde{p}_{\mathrm{R}}$) on which $\widetilde{p}_{\mathrm{L}}$ (resp., $\widetilde{p}_{\mathrm{R}}$) intersects $\widetilde{c}$. These four dual-vertices $\widetilde{u}_{\mathrm{L}}$, $\widetilde{v}_{\mathrm{L}}$, $\widetilde{u}_{\mathrm{R}}$ and $\widetilde{v}_{\mathrm{R}}$ divide $\widetilde{c}$ into four subsequences. We can get two disjoint subsequences from $\widetilde{u}_{\mathrm{L}}$ to $\widetilde{v}_{\mathrm{L}}$ or $\widetilde{v}_{\mathrm{R}}$ and from $\widetilde{u}_{\mathrm{R}}$ to $\widetilde{v}_{\mathrm{R}}$ or $\widetilde{v}_{\mathrm{L}}$. If otherwise; for example, if $\widetilde{u}_{\mathrm{L}}$ is adjacent to $\widetilde{u}_{\mathrm{R}}$ and $\widetilde{v}_{\mathrm{L}}$, and $\widetilde{u}_{\mathrm{R}}$ is adjacent to $\widetilde{u}_{\mathrm{L}}$ and $\widetilde{v}_{\mathrm{L}}$ on $\widetilde{c}$, then $\widetilde{v}_{\mathrm{R}}$ never appears on $\widetilde{c}$, a contradiction. Therefore, there exist two disjoint dual-paths consisting of three subpaths; namely, the subpaths from $\widetilde{u}$ to $\widetilde{u}_{\mathrm{L}}$ (resp., $\widetilde{u}_{\mathrm{R}}$), the disjoint subsequences of $\widetilde{c}$ from $\widetilde{u}_{\mathrm{L}}$ (resp., $\widetilde{u}_{\mathrm{R}}$) to $\widetilde{v}_{\mathrm{L}}$ or $\widetilde{v}_{\mathrm{R}}$, and the subpath from $\widetilde{v}_{\mathrm{L}}$ (resp., $\widetilde{v}_{\mathrm{R}}$) to $\widetilde{v}$.

Finally, assume that $\widetilde{u}$ and $\widetilde{v}$ are on different floor- or ceiling-cycles $\widetilde{c}_{\widetilde{u}}$ and $\widetilde{c}_{\widetilde{v}}$. From the definition, each of $\widetilde{c}_{\widetilde{u}}$ and $\widetilde{c}_{\widetilde{v}}$ has at least one dual-vertex of $\widetilde{H}$ in its inside; let $\widetilde{w}_{\widetilde{u}}$ and $\widetilde{w}_{\widetilde{v}}$ denote them. There exist two disjoint dual-paths $\widetilde{p}_{\mathrm{L}}$ and $\widetilde{p}_{\mathrm{R}}$ between $\widetilde{w}_{\widetilde{u}}$ and $\widetilde{w}_{\widetilde{v}}$ in $\widetilde{H}$. Note that $\widetilde{p}_{\mathrm{L}}$ and $\widetilde{p}_{\mathrm{R}}$ have distinct intersections with $\widetilde{c}_{\widetilde{u}}$; let $\widetilde{u}_{\mathrm{L}}$ (resp., $\widetilde{u}_{\mathrm{R}}$) be the last intersection of $\widetilde{p}_{\mathrm{L}}$ (resp., $\widetilde{p}_{\mathrm{R}}$) and $\widetilde{c}_{\widetilde{u}}$ from $\widetilde{w}_{\widetilde{u}}$ to $\widetilde{w}_{\widetilde{v}}$. Similarly, let $\widetilde{v}_{\mathrm{L}}$ (resp., $\widetilde{v}_{\mathrm{R}}$) be the first intersection of $\widetilde{p}_{\mathrm{L}}$ (resp., $\widetilde{p}_{\mathrm{R}}$) and $\widetilde{c}_{\widetilde{v}}$. Clearly, there exist two disjoint dual-paths from $\widetilde{u}$ to $\widetilde{u}_{\mathrm{L}}$ and $\widetilde{u}_{\mathrm{R}}$ on $\widetilde{c}_{\widetilde{u}}$; one can find them by going along $\widetilde{c}_{\widetilde{u}}$ clockwise and anti-clockwise. Similarly, we have two disjoint dual-paths from $\widetilde{v}_L$ and $\widetilde{v}_{\mathrm{R}}$ to $\widetilde{v}$. By combining them, there exist two disjoint dual-paths from $\widetilde{u}$ to $\widetilde{v}$. Note that there may be some floor- or ceiling-cycle touching these disjoint dual-paths. Even in this case, by an argument similar to the above, we can show two disjoint dual-paths between $\widetilde{u}$ and $\widetilde{v}$. □

Finally, we confirm as a corollary of Lemma 9.7 that the condition (F4) holds.

Corollary 9.10. *The size of each face of $\widetilde{H}'$ is $O(\sqrt{k})$.*

Proof. As mentioned above, the faces of $\widetilde{H}'$ are defined by boundary dual-cycles of type (i) $\sim$ (iv); see the comment after Definition 9.6. For those defined by floor-, ceiling-cycles, and branch-triangles, the size bound of the lemma clearly holds by definition. Thus, we below consider faces defined by modified frame-cycles.

Consider any face defined by a modified frame-cycle, and let $(\widetilde{c})^-$ and $\widetilde{c}$ denote respectively the original frame-cycle and the pre-frame-cycle from which $(\widetilde{c})^-$ is defined (Definition 8.1). Consider any dual-path $\widetilde{p}$ of $\widetilde{c}$ connecting a boss-vertex of $\widetilde{c}$ and its "next" dual-vertex of a branch-triangle used in $\widetilde{c}$.

By our modification, a part $\widetilde{p}'$ of $\widetilde{p}$ that is in the outside of the corresponding floor-cycle and ceiling-cycle (if it exists) could be used as a component of the modified frame-cycle, and its length is bounded by $4\sqrt{k}$ by Lemma 9.8. Note that the floor-cycle may not be used in $\widetilde{H}'$ if it only intersects with the line part of $\widetilde{c}$ that is removed for defining $(\widetilde{c})^-$ (e.g., Figure 20 (a)). In this case, however, only a part of $\widetilde{p}'$ is used for the modified frame-cycle, which is even shorter. Thus, the modified frame-cycle consists of (at most) four such reduced dual-paths, (a part of) two floor-cycles, (a part of) four ceiling-cycles (i.e., could be two for each branch-triangle), and two dual-edges from two branch-triangles, and their total length is $O(\sqrt{k})$. □

Algorithms for Step 2.4

The task of this step is to compute information of the modified frame-graph $\widetilde{H}'$. We again explain as if each step receives the output of the previous step as input, and omit analysis of time complexity. Note that we can use the outputs of Step 2.3, i.e., the frame-graph $\widetilde{H}$, as input. We consider an algorithm that outputs

(a) a list of core-cycles;

(b) a list of interior- and exterior-cycles;

(c) a list of floor- and ceiling-cycles; and

(d) the complete weighted face information of $\widetilde{H}'$.

First we point out that a cycle, a sequence of vertices representing a (directed) cycle, is $\widetilde{O}(1)$-space computable provided we have a way to distinguish a set of edges of the cycle (*and*, more precisely, it is guaranteed that this set of edges indeed forms a cycle). Also for a given cycle c of G, the task of computing an m.d.-cycle consisting of dual-vertices left/right adjacent to c can be done locally and hence in $\widetilde{O}(1)$-space; see the proof of Lemma 9.2 and Lemma 9.4.

Once these algorithmic techniques are clear, what remains to show here for computing the above (a) and (b) is a way to identify core/interior/exterior boundary cycle edges.

For (a), we first compute all boundary cycles of $\mathrm{Core}(\widetilde{b})$ for all $\widetilde{b} \in \widetilde{I}$. For any $\widetilde{b} \in \widetilde{I}$, consider $\mathrm{Core}(\widetilde{b})$. Note that it is a subset of $\mathrm{N}_k(\widetilde{b})$; thus, we have an $\widetilde{O}(k)$-space algorithm that determines whether a given dual-vertex belongs to $\mathrm{Core}(\widetilde{b})$ or not by using the same BFS algorithm of Step 2.1. Then by using this procedure, we can identify edges of each boundary cycle of $\mathrm{Core}(\widetilde{b})$; basically, what we need is to check, for a given candidate edge of G, whether one of its two adjacent dual-vertices belongs to $\mathrm{Core}(\widetilde{b})$ and the other does not. Next we select *core* boundary cycles from all enumerated boundary cycles. For a given boundary cycle c, we can use Reingold's algorithm to determine whether a given dual-vertex is in the outside of c; hence, we can count the number of dual-vertices outside of c and determine whether c is a core boundary cycle. Thus, we can enumerate all core boundary cycles, from which we can enumerate all core-cycle as the output (a).

For (b), i.e., interior- and exterior-cycles, we first compute the level of all dual-vertices of $\widetilde{G}$. For each dual-vertex $\widetilde{v}$ and its boss-vertex $\widetilde{b}$, we have an algorithm that computes $\mathrm{dist}(\widetilde{v}, \widetilde{b})$ in $\widetilde{O}(k)$-space (Lemma 5.4). The level of all dual-vertices is computable by this procedure. Based on this level information, we can identify connected components of dual-vertices with the same level (outside of $\cup_{\widetilde{b}\in\widetilde{I}}\mathrm{N}_k(\widetilde{b})$). Again this procedure is enough for enumerating interior and exterior boundary cycles, and m.d.-cycles right-adjacent to them. Once we obtain a list of all m.d.-cycles right-adjacent to interior or exterior boundary cycles, the remaining task for computing the output (b) is to remove all line parts from these m.d.-cycles and decompose them into dual-cycles, which can be done in $\widetilde{O}(1)$-space.

By using the core-, interior- and exterior-cycles obtained above as input, it is easy to list floor- and ceiling-cycles; that is, the output (c). We compute the size of each cycle and check whether it is a neck, and count the number of dual-vertices in its inside and check whether it is biased. By these computations, biased neck interior- and exterior-cycles are obtained in $\widetilde{O}(1)$-space. Furthermore, we can check inclusion relations of these cycles and branch-triangles in $\widetilde{O}(1)$-space, from which we can obtain a list of floor- and ceiling-cycles, i.e., the output (c).

Now we can use $\widetilde{H}$ and the list of floor- and ceiling-cycles as input for computing the output (d). For this, we start with the list of frame-cycles (of $\widetilde{H}$) as an initial list of $\widetilde{H}'$-face-cycles. For each floor-

and ceiling-cycle, we check whether the dual-cycle has at least one dual-vertex of $\widetilde{H}$; if it does, then it is added to the list, and the corresponding frame-cycles of $\widetilde{H}$ in the list are modified appropriately (if necessary). In this way we can compute a list of all $\widetilde{H}'$-face-cycles in $\widetilde{O}(1)$-space. Once we have the list, the rest of the task is similar to the previous sub-steps.

Acknowledgements

A major part of this work was done during the project "Exploring the Limits of Computation (ELC)" supported by MEXT KAKENHI Grant No. 24106008. During the ELC project, many researchers have joined our discussion which helped us very much for finding some errors and improving the presentation of our idea of separator algorithms. In particular, we thank to Dr. Mohit Garg and Dr. Sebastian Kuhnert for their technical contributions. Pavan's and Vinodchandran's research are partly supported by NSF grants 1849053, 0916797,1849048, and 0916525.

References

[1] E. Allender and M. Mahajan, The complexity of planarity testing, *Information and Computation* 189(1): 117–134, 2004.

[2] T. Asano and B. Doerr, Memory-constrained algorithms for shortest path problem, in *Proc. of the 23rd Annual Canadian Conference on Comp. Geometry* (CCCG'11), 2011.

[3] T. Asano, D. Kirkpatrick, K. Nakagawa, and O. Watanabe, $O(\sqrt{n})$-space and polynomial-time algorithm for the planar directed graph reachability, in *Proc. of the 39th Int'l Sympos. on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 8635, 45–56, 2014.

[4] R. Ashida, S. Kuhnert, and O. Watanabe, A space-efficient separator algorithm for planar graphs, IEICE Trans. on Fundamentals, 2019, to appear.

[5] R. Ashida and K. Nakagawa, $\widetilde{O}(n^{1/3})$-space algorithm for the grid graph reachability problem, in *Proc. of the 34th Int'l Sympos. on Computational Geometry*, 5:1–5:13, 2018.

[6] G. Barnes, J.F. Buss, W.L. Ruzzo, and B. Schieber, A sublinear space, polynomial time algorithm for directed s-t connectivity, in *Proc. Structure in Complexity Theory Conference*, IEEE, 27–33, 1992.

[7] R. Diestel, *Graph Theory* (the 4th edition), Springer, 2010.

[8] H.N. Djidjev, On the problem of partitioning planar graphs, *SIAM Journal on Algebraic and Discrete Methods*, 3 (2):229–240, 1982.

[9] H.N. Djidjev and S.M. Venkatesan, Reduced constants for simple cycle graph separation, *Acta Informatica*, 34 (3):231–243, 1997.

[10] H. Gazit and G.L. Miller, A parallel algorithm for finding a separator in planer graphs, in *Proc. of the 28th Annual Symposium on Foundations of Computer Science* (FOCS'87), 238–248, 1987.

[11] M. Holzer, F. Schulz, D. Wagner, G. Prasinos, and C. Zaroliagis, Engineering planar separator algorithms, *Journal of Experimental Algorithmics*, 14 5:1.5–5:1.31, 2009.

[12] T. Imai, K. Nakagawa, A. Pavan, N.V. Vinodchandran, and O. Watanabe, An $O(n^{\frac{1}{2}+\epsilon})$-space and polynomial-time algorithm for directed planar reachability, in *Proc. of the 28th Conference on Computational Complexity* (CCC'13), IEEE, 277–286, 2013.

[13] G.L. Miller, Finding small simple cycle separators for 2-connected planar graphs, *Journal of Computer and System Sciences* 32(3):265–279, 1986.

[14] R.J. Lipton and R.E. Tarjan, A separator theorem for planar graphs, *SIAM Journal on Applied Mathematics* 36(2):177–189, 1979.

[15] O. Reingold, Undirected connectivity in log-space, *J. ACM*, 55(4), 2008.

[16] A. Wigderson, The complexity of graph connectivity, in *Proc. 17th Mathematical Foundations of Computer Science* (MFCS'92), Lecture Notes in Computer Science 629, 112–132, 1992.