Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 147 (2022)

Evaluating Monotone Circuits on Surfaces

Samir Datta
Chennai Mathematical Institute, Chennai, India

Chetan Gupta
Aalto University, Finland

Abstract

In this paper, we study the circuit value problem for monotone Boolean circuits (that is, circuits without negation gates) when the circuits are embedded on a surface of bounded genus, and all inputs to the circuits lie on at most constantly many faces. We show that this problem can be solved in LogDCFL thus by a result of Cook [6], yielding a space-efficient ($O(\log^2 n)$-space) and polynomial time algorithm for the problem. It also yields a highly parallel algorithm (simultaneously $O(\log n)$-time with polynomially many processors).

This generalises the previous bound of LogDCFL on one input face planar circuits [5]. More precisely, we show that if a monotone circuit is embedded on a surface of polylogarithmic genus g and has k faces on which all the inputs are present, then the circuit can be evaluated on a CROW-PRAM (concurrent read *owner* write parallel random access machine) in time $O(g(k+g)\log n)$ using $n^{O(1)}$ many processors.

Our main technical idea is a distance metric in single sink DAGs that can be computed in deterministic logarithmic space (L) and is useful in partitioning the circuit into subcircuits such that each one is a one-input face monotone planar circuit. We show that the partitioning procedure is in L. Thus we are able to side-step the barrier of computing the usual distance in bounded genus graphs, for which the best bound known is $\mathsf{UL} \cap \mathsf{coUL}$ [17, 30] and therefore not known to be contained in LogDCFL.

2012 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases Monotone Planar Circuit Value Problem, Bounded Genus, Parallel Algorithms, Bounded Space Complexity, LogDCFL, SC

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Acknowledgements We would like to thank Eric Allender for his detailed comments on a preliminary version of this manuscript. We would also like to thank anonymous reviewers for suggesting stylistic improvement. In particular, we would like to thank one of these anonymous reviewers for pointing out a mistake in the main theorem, which is fixed now.

1 Introduction

The Circuit Value Problem CVP – "Given a Boolean circuit consisting of AND ($\wedge$), OR ($\vee$), NOT ($\neg$)-gates and a Boolean assignment to its input values, what is the value output by the circuit?" – occupies an important place in complexity theory as the archetypal P-complete problem. Various relaxations of the problem are known to be P-complete such as *Monotone* Circuit Value Problem MCVP where there are no $\neg$ gates and *Planar* Circuit Value Problem PCVP where the circuit is itself embedded as a planar graph [12]. It is somewhat remarkable that a combination of the previous two problems *Monotone Planar* Circuit Value Problem MPCVP is parallelisable and therefore contained in NC [7, 32, 21] and hence is not expected to be P-complete. A series of papers have been devoted to refining the exact complexity of MPCVP and its restrictions [7, 22, 3, 19, 5]. Layering the circuit and restricting the outer face of the circuit to contain all the inputs are two specializations which dramatically improves the complexity bound on MPCVP.

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ISSN 1433-8092

MCVP can also be viewed as a generalization of reachability in a restricted class of graphs. This is so because reachability in a single sink DAG can be viewed as a circuit in which all gates are $\vee$-gates. Thus, MPCVP is a generalization of single sink planar DAG reachability. There has been considerable work on trying to pin down the exact complexity of planar reachability [4, 29] and of other topologically restricted reachability instances such as those embedded on a surface of bounded genus [17, 14]. Another thread of work has focused on DAG reachability with few sources/sinks in planar [1] and somewhat non-planar instances [1, 26, 27].

Overall there are three common restrictions on MCVP that make the problem efficiently parallelizable:

- Topological: Ensuring that the circuit is embedded on a plane or on a surface of bounded genus.
- Layering: Ensuring that the gates are partitioned into layers such that edges are between adjacent layers only.
- Single input face: Ensuring that there is a face containing all the inputs of the circuit.

The first set of papers [13, 9] on the topic imposed all three kinds of restrictions and gave the $\mathsf{LogCFL} \subseteq \mathsf{NC}^2$ bound on the problem. Subsequent work [22, 21] gave parallel algorithms for the problems which were processor efficient (i.e. used linearly many processors) but had weaker bounds on the running time with and without the layering constraint. In an independent thread of work [7] used the single input face constraint and used it to solve the (node bimodal) MPCVP problem in NC. Notice that they assumed that each gate has only two inputs by expanding gates with larger fan-in as a tree – this is directly possible only if the inputs and outputs of a gate do not intersperse i.e. as a directed graph, the each node is bimodal. The bound on upward planar, layered and single input face MPCVP was optimized to LogDCFL in [3]. Later, [19] removed the upward planar and layering restrictions but kept the single input face restriction to prove a bound using both LogDCFL and planar longest path in DAGs, the latter problem is known to be in $\mathsf{UL} \subseteq \mathsf{NL}$ (by [18, 4, 29]). In [5] this dependence on finding a planar longest path was removed. Also, in [19] the topological restrictions were relaxed from planar to toroidal yielding a $\mathsf{L}(\mathsf{LogCFL}) = \mathsf{SAC}^2 \subseteq \mathsf{NC}^3$ bound for the monotone toroidal circuit value problem with no restriction on number of input faces.

In this work, we dispense completely with the layering restriction and show a smooth tradeoff between parameterized versions of the other two restrictions on the one hand and the complexity of the MCVP problem on the other. This provides a common generalization of the MPCVP results and of the DAG-reachability results mentioned above.

1.1 Our Results

The primary problem of interest for us is what we call kMgCVP – this stands for monotone circuit value problem for genus g circuits such that there are k faces of the embedded circuit that contain all the inputs. Thus the case with $k = 1, g = 0$ is the usual single input face MPCVP that was studied in [5] building on [3] and using insights from [19]. In this work, building on [3, 19, 5] we show that kMgCVP is solvable in $\mathsf{CROW}[g(k+g)\log n]$ where $\mathsf{CROW}[t(n)]$ is the class of languages accepted by a Concurrent Read Owner Write PRAM (or CROW PRAM) machine in (parallel) time $O(t(n))$ and with $n^{O(1)}$ processors.

Notice that Dymond and Ruzzo [10] proved that $\mathsf{CROW}[\log n]$ is precisely LogDCFL, i.e. recognizable by a deterministic logspace machine equipped additionally with a polynomial height stack. Our main results can be summarised as follows:

▶ **Theorem 1.** *The following are true:*

1. kMgCVP *problem when the number of input faces* $k = O(1)$ *and the genus* $g = O(1)$ *can be solved in* LogDCFL.
2. *The* MGCVP *problem with the number of input faces* $k = (\log n)^{O(1)}$ *and with the genus* $g = (\log n)^{O(1)}$ *is in* NC.

Notice that the length of the longest path was used in [19] and previous papers to layer the graph so that the LogDCFL algorithm of [3] can be used. Since this is in UL ∩ coUL for planar DAGs they got a complexity bound larger than this. In [5], layering was done by using a grid embedding and so the complexity of layering was reduced to L. Some cut-and-paste surgery was also required, which was the main technical content of [5]. Generalizing this to a larger genus seems difficult because working with a grid embedding on a surface is hard. Even if we could work with a grid-embedded fundamental polygon of the surface – we would need to do major surgery as in [17]. We are not sure how to do this while preserving the grid embedding. We take a different approach as outlined below. The main idea behind

Restrictions			Bound
Topological	Layering	#input faces	
Upward Planar	✓	1	DSPACE$[\log^2 n]$ [13]
Upward Planar	✓	1	LogCFL[13]
Upward Planar	✓	1	LogDCFL = CROW$[\log n]$ [3]
Planar ($g = 0$)	✓	∞	EREW$[\log^2 n]$ [21]
Planar ($g = 0$)	✗	∞	CRCW$[\log^4 n]$ [7]
Planar ($g = 0$)	✗	∞	EREW$[\log^6 n]$ [22]
Planar ($g = 0$)	✗	1	LogDCFL ⊕ (UL ∩ coUL) [19]
Planar ($g = 0$)	✗	1	LogDCFL [5]
Toroidal ($g = 1$)	✗	∞	AC^1(LogCFL) = SAC^2 [19]
$g = O(1)$	✗	$O(1)$	$\boxed{\text{LogDCFL}}$
$g = (\log n)^{O(1)}$	✗	$(\log n)^{O(1)}$	$\boxed{\text{NC}}$

Table 1 Previously known and $\boxed{\text{new}}$ results (∞ refers to unbounded number of input faces)

our results is the introduction of a new measure of the "distance" of a node in a graph to the unique sink t that is conceptually simple as well as computable in L. This allows us to chop up a planar circuit into annuli consisting of vertices spanning a range of distances to t such that the number of input faces in each annulus is at most half of those in the parent graph. Thus we can, in $O(k)$ number of steps (where k is the number of input faces in the original planar graph) reach the base case of one input face. That can be solved via [5] in LogDCFL and by the equivalence of LogDCFL and CROW$[\log n]$ [10] by an Owner PRAM in logarithmic time. Composing the computation of the individual PRAMs we get an $O(k \log n)$ parallel time algorithm for planar i.e. genus zero graphs. Moving on to circuits embedded on a genus g surface we chop them into $O(g)$ subcircuits each of which is either planar with $O(g + k)$ number of input faces or of constant depth (even though it may be embedded on a high genus surface) and hence can be evaluated in CROW$[(g + k) \log n]$ in both the cases. Composing the functions computed by the $O(g)$ pieces, we are able to get a bound of CROW$(g(g + k) \log n))$.

Our notion of "distance" is based on the number N_v of nodes that can reach a node v. Thus, distance between two nodes u and v, $d_{\#}(u, v)$ can be defined as $N_v - N_u$ if u can reach v and ∞ (alternatively, undefined) otherwise. Notice that if reachability in a class of DAGs is in $\mathcal{C}$ then $d_{\#}(., .)$ is computable in $\mathsf{L}^{\mathcal{C}}$. In our case $\mathcal{C}$ is almost invariably L ensuring that $d_{\#}(., .)$ is also computable in L.

1.2 Organization of the Paper

The rest of the paper is organized as follows. In Section 2, we state some previous results, talk briefly about some complexity classes and among them and define some necessary notation that we use in the paper. In the first part of Section 3, we prove our result for planar circuits and then in the second part we generalize it to bounded genus circuits. Finally, in Section 4, we conclude our result and leave some open questions for future work.

2 Preliminaries

A Boolean circuit is a DAG (directed acyclic graph) which contains three types of nodes: *source nodes, a sink node* and *internal nodes*. Each internal node is labelled by an AND, OR or NOT gate. Edges in the graph represent the connection (wires) between two nodes (gates). In a Boolean circuit, source nodes are fed with an input binary string, and the circuit produces an output on the sink node by applying the sequence of Boolean operations represented by internal nodes. A circuit is called *n-input* circuit if the number of source nodes in the circuit is n. Given a *n-input* circuit along with a binary string of length n, problem of evaluating the output of the circuit (value at sink node) is called *circuit-value problem* (CVP). A Boolean circuit which does not have any NOT gate is called a *monotone* circuit. A circuit that can be embedded on a plane without crossing its edges is called a planar circuit and the problem of evaluating those monotone circuits is called monotone planar circuit value problem (MPCVP). Circuits that can be embedded on surfaces are a natural extension of planar circuits. For a circuit embedded on a surface such that all the source nodes of the circuit lie on k-faces, we call it k-input-face circuit. We denote the problem of evaluating a k-input faces monotone circuit that can be embedded on a genus g surface as kMgCVP. Which is the problem we are focusing on in this paper. We use the following result proved for one-input-face planar circuits.

▶ **Lemma 2.** *[5] One-input-face monotone planar circuit value problem can be solved in* LogDCFL*.*

As we mentioned that a circuit could be represented by a DAG; thus from now on we will identify a circuit as a DAG and write everything in terms of graphs. If a node in the graph does not have any incoming edges of the graph incident on it, we call it a source node or input node. Similarly, a node with no outgoing edges is called a sink node of the graph.

2.1 Graph Theory

Most of the directed graphs that we use are Directed Acyclic Graphs or DAGs. A DAG is said to be connected when the underlying undirected graph is connected. In general, when we refer to digraphs as graphs we are referring to the underlying undirected graph.

Graphs that can be embedded on surfaces form an important class for us and we proceed to introduce them. See [8, Appendix B] for a more detailed exposition. A g-genus surface is a sphere with g-many handles on it. A graph is called a g-genus graph if g is the minimum integer such that the graph can be embedded on a g-genus surface without intersecting its edges. A 2-cell embedding of a graph is an embedding in which every face of the graphs is homeomorphic to an open disk. A graph of genus g always has a 2-cell embedding on a surface of genus g. For a graph G and surface S, we will use $\mathsf{g}(G)$ and $\mathsf{g}(S)$ to denote the genus of G and S respectively. We know that if G is embedded on S then $\mathsf{g}(G) \leq \mathsf{g}(S)$. Cycles in a surface embedded graph can be divided into two categories, *surface separating* cycles and *surface non-separating* cycles. As the name suggests, surface separating cycles are those cycle such that cutting the surface along those cycles divides the surface into at least two disjoint surfaces. Surface non-separating cycles are those cycles such that cutting the surface along these cycles does not separate the surface but reduces the genus of the surface. We will use the following lemmas related to surface separating and surface non-separating cycles in surface embedded graphs. We will also use the following lemmas in our result (Lemma B.4 and Lemma B.5 from [8]).

▶ **Lemma 3** ([8]). *Let C be a surface separating cycle in a surface S, and S' and S'' be the surfaces obtained from S by cutting along C and capping the holes. Then* $\mathsf{g}(S) = \mathsf{g}(S') + \mathsf{g}(S'')$.

▶ **Lemma 4** ([8]). *Let C is a surface non-separating cycle in a surface S, and S' be the surface obtained from S by cutting along C and capping the holes. Then* $\mathsf{g}(S') = \mathsf{g}(S) - 1$.

We will also use the following lemmas about the deterministic logarithmic space (L) computable properties of graphs (a crucial final step in their proof is the equivalece of SL and L [23])

▶ **Lemma 5** ([2]). *Given a graph G, we can check if G is planar or not in* L.

▶ **Lemma 6** ([27]). *Given a directed acyclic graph embedded on a surface of genus $2^{O(\sqrt{\log n})}$ with $2^{O(\sqrt{\log n})}$ source nodes, we can check whether there is a directed path from a node u to another node v in* L.

Our algorithm requires an embedding that has simultaneously bounded genus and bounded number of input faces. While it is possible to test for the embeddability of a graph on a surface of bounded genus and even obtain the embedding in L [11] – this does not suffice for our purpose since we do not know how to concurrently ensure that the embedding has a bounded number of input faces.

$V(G), E(G)$ and $F(G)$ to represent the set of nodes, set of edges and set faces in a graph G, respectively. Although, when we write that $v \in G$ (or $e \in G$, $f \in G$), we mean $v \in V(G)$ (respectively $e \in E(G), f \in F(G)$). We refer to the number of vertices, faces and edges in a graph G by $\#v(G), \#f(G), \#e(G)$ respectively.

2.2 Complexity classes

Classically, the common parallel computation model is the PRAM or Parallel Random Access Machine [16] where many processors communicate via shared memory. A problem is said to be parallelisable if it can be solved in the PRAM model using $\log^{O(1)} n$ time using polynomial number of processors. PRAM models can further be distinguished on the basis of how they resolve read and write conflicts. Thus the weakest model is the EREW PRAM model or the exclusive read exclusive write model that stipulates that there are no concurrent writes or reads for any memory location. On the other extreme is the CRCW PRAM where concurrent reads and writes are both permitted with several ways of resolving conflicts, most of which are shown to be equivalent (see [16]). Intermediate between these two types of PRAM models are the CREW PRAM models where the writes are exclusive but reads can be concurrent. While the fine-grained mapping between CREW, EREW PRAM models and Turing machine models or circuit models is not precise, CRCW PRAM models correspond naturally to both alternating Turing machines [24] and unbounded fan-in circuits [25]. There is a variant of the CREW PRAM model namely, the CROW PRAM model – where writes are not only exclusive but can be made only by a designated owner processor for a particular memory cell – that

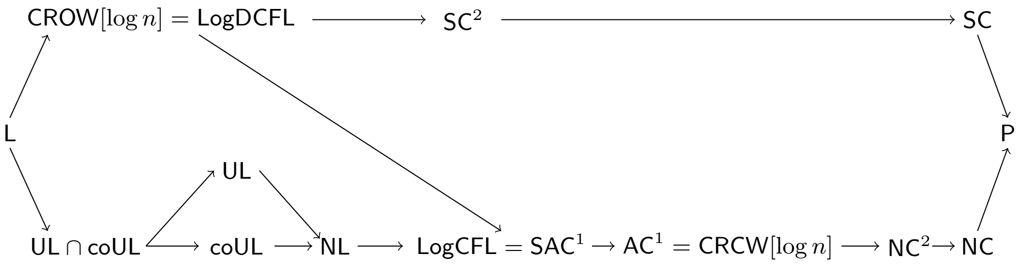

Figure 1 Relevant complexity classes and relation among them.

yields a complexity class corresponding to a natural Turing machine class. This is our main protagonist amongst complexity classes. We put it in perspective below.

- LogDCFL : class of languages reducible to deterministic context-free languages using logspace reductions [28]. Alternatively, they are languages accepted by deterministic Aux-PDAs with a pushdown stack in polynomial time (colloquially "logspace with polynomial stack") [28]. Cook proved that LogDCFL is contained in the class of problems that can be solved simultaneously in polylogarithmic space and polynomial time SC [6, 31]. They are also known to be contained in LogCFL—the class languages that are logspace reducible to context-free languages. Alternatively, LogCFLs are the uniform version of the circuit class SAC^1, which is a class of languages that are accepted by a circuit family of depth $O(\log n)$ and size polynomial in the number of inputs (like AC^1) but only OR-gates have polynomial fan-out while the AND-gates have bounded fan-in. LogCFL is known to be contained in NC^2 (by just replacing large fan-in OR-gates by trees of fan-in 2) thus, so is LogDCFL though LogCFL is not known to be contained in SC.
  Dymond and Ruzzo [10] showed a PRAM characterisation of LogDCFL in terms of Owner writes (any subset of processors can read from a memory location but only the designated owner of a memory location can write to it – a notion weaker than the CREW PRAM model) in a model known as CROW-PRAM. They showed that LogDCFL is precisely the class of languages that can be solved by polynomially many processors in $O(\log n)$ time which form a concurrent read owner write PRAM or equivalently $\mathsf{CROW}[\log n]$. Further [20] show other circuit based characterisations of LogDCFL, LogCFL.
- UL or unambiguous logspace is the class of languages accepted by a nondeterministic Turing machine that is unambiguous i.e. has at most one accepting path on any input. This class is clearly contained in $\mathsf{NL} \subseteq \mathsf{LogCFL} \subseteq \mathsf{NC}^2$ but like LogCFL is not known to be contained in SC. This class has achieved some prominence because planar restrictions of important problems like reachability [4, 29] and distance [30] are known to be contained here (or indeed in the slightly smaller class $\mathsf{UL} \cap \mathsf{coUL}$).
  These unambiguous classes are the main villains for us and we show how to circumvent these by instead using the logspace algorithms for single sink planar DAG reachability from [1] and similar reachability for DAGs with a single sink embedded on a surface of bounded genus [26, 27].

3 Evaluating Monotone Circuits

Let us first define the notion of *distance* that we are using in this paper. If G is a directed acyclic graph and v is in a node in G then N_v represents the number of nodes in G which can reach v, i.e. nodes that have a directed path to v. We define distance $d_{\#}(u, v)$ between

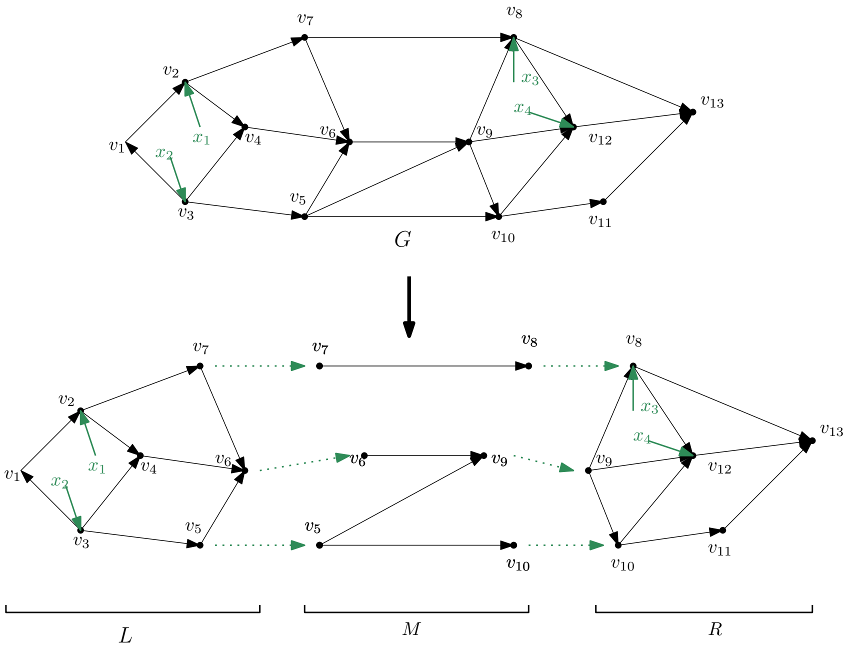

Figure 2 Graph G with two input faces $f_1 = \{v_1, v_2, v_3, v_4\}$ and $f_2 = \{v_8, v_9, v_{12}\}$, edge partitioned into the graphs L, M and R such that f_1 is contained in L and f_2 is contained in R. Output from L (i.e. v_5, v_6, v_7) is input for M and output of M is input for R. After edge partition the outer face $\{v_8, v_9, v_{10}, v_{11}, v_{13}\}$ of R becomes a new input face in R, receiving inputs at v_8, v_9 and v_{10}.

two nodes u and v as follows[1]:

$$d_{\#}(u,v) = \begin{cases} N_v - N_u, & \text{if there is directed path from } u \text{ to } v \\ \infty, & \text{otherwise.} \end{cases}$$

▶ **Observation 7.** *If there is a directed path from a node u to node v in G such that $u \neq v$ then $d_{\#}(u,w) > d_{\#}(v,w)$, for all nodes w reachable from both u and v.*

Since G is DAG and there is path from u to v, there will not be a path from v to u. Thus we know that $N_v > N_u$, which implies that $d_{\#}(u,w) > d_{\#}(v,w)$. If G is a DAG with only one sink node t then we define $G^{(i)}$ to be the subgraph of G induced by the vertices v such that $d_{\#}(v,t) \leq i$. This means $G^{(0)} = t$ (the sink node), and $G^{(n)} = G$.

▶ **Lemma 8.** *For any connected directed acyclic graph G, $G^{(i)}$ is a connected subgraph of G for all $i \in [n]$.*

Proof. The claim trivially holds for when $G^{(i)} = t$. For the sake of contradiction, assume the graph $G^{(i)}$ contains more than one node and is disconnected i.e., there is a node u such that u and t are in different components of $G^{(i)}$. Since G is a DAG with only one sink node t, we know that there is a directed path P from u to t in G. Now assume that u and t are in different components of $G^{(i)}$. We can say that there exists a node v in P such that v does not belong to $G^{(i)}$. If v is not a node in $G^{(i)}$, this means $d_{\#}(v,t) > i$. On the other hand u is a node in $G^{(i)}$ therefore $d_{\#}(u,t) \leq i$. By Observation 7, it must hold that $d_{\#}(u,t) > d_{\#}(v,t)$, which is a contradiction. Thus we can conclude that $G^{(i)}$ is a connected subgraph of G. ◀

[1] $d_{\#}(.,.)$ is a *quasimetric* that satisfies axioms of metric except symmetry. The fact that we can compute this in L may be of independent interest (see Lemma 9).

▶ **Lemma 9.** *In a bounded genus directed acyclic graph with one sink, we can compute $d_{\#}(u, v)$ for any pair of nodes u and v in the graph in* L.

Above lemma follows simply from Lemma 6. We can reverse the direction of all the edges of the single sink DAG so that the resulting graph becomes a single source DAG and then we can use Lemma 6 to check reachability.

In the following sections, we begin with a graph (planar or bounded genus) and divide it into multiple subgraphs. At any point, for any two nodes u and v of the graph, no matter whether u and v remain in the same subgraph or different after division, $d_{\#}(u, v)$ is always computed with respect to the initial graph.

CROW-**Transducers:** A CROW-PRAM accepts a language but we can as well use CROW-PRAMs to define functions that take a sequence of bits and output a polynomially bounded sequence of bits. In particular, we say that a function family $f_n : \{0,1\}^n \to \{0,1\}^m$, where ($m = m(n)$ is polynomially bounded in n) is computable by a $\mathsf{CROW}[\log n]$ transducer if the map $f^{(i)} : x \mapsto (f(x))_i$ is [2] in $\mathsf{CROW}[\log n]$ for every $i \in \{1, \ldots, m\}$.

We define functional composition of functional families with polynomially bounded outputs is the usual way: let, $f_n : \{0,1\}^n \to \{0,1\}^{m(n)}$ and and $f'_n : \{0,1\}^n \to \{0,1\}^{m'(n)}$ be two function families where $m(n), m'(n) = n^{O(1)}$. Define the function family $g = f' \circ f$ where, $g_n : \{0,1\}^n \to \{0,1\}^{m'(m(n))}$ maps $g_n : x \mapsto f'(f(x))$. The following lemma is what what makes the equivalence of $\mathsf{LogDCFL}$ and $\mathsf{CROW}[\log n]$ useful for us:

▶ **Lemma 10.** *If f, f' are functional families with polynomially bounded outputs computable in $\mathsf{CROW}[\log n]$ then so is their composition $f' \circ f$.*

The proof is straightforward and we supply it only for the sake of completion.

Proof. Given a string $x \in \{0,1\}^n$, each of the n^c bits of $f_n(x)$ can be computed in $e \log n$ time using n^d processors in via a CROW-PRAM. Here c, d, e are constants. Thus with n^{c+d} processors we can compute all bits of the outputof f Let, the corresponding constants for f' be c', d', e' Then the time to compute all the bits of $f' \circ f$ on input x is $(e + e') \log n$ and the number of processors required is $n^{c+d} + n^{(c+d)(c'+d')}$.

Moreover there are no concurrent writes in the composition since we can assign the same owners to memory locations as those in the computation of each bit of $f(x), f'(y)$ where $y = f(x)$. ◀

3.1 Monotone Circuit Value Problem in Planar Graphs

In this section, we will prove that a monotone planar circuit such that all its inputs lie on at most k-faces, can be evaluated in $\mathsf{CROW}[\log k \log n]$. The high-level idea is as follows. We show that if G is the planar DAG that represents the given circuit such that it has at most k-input faces, we edge-partition G using a logspace procedure into three subgraphs L, M and R such that (i) L is a collection of DAGs each containing $(\frac{k}{2})$ input faces, (ii) R is a DAG that contains at most $(\frac{k}{2})$ input faces and, (iii) M is a 2-layered graph – a graph with two layers of nodes such that all the edges go from one layer to another. Then we recursively apply the same procedure on graphs L and R. We repeat this procedure for $O(\log k)$ steps. In each recursive call, we decrease the number of input faces in each graph by half. Therefore, after $O(\log k)$ steps, we partition the graph into smaller graphs such that each graph that

[2] For a string $s \in \{0,1\}^n$, the i-th bit of s is represented by s_i for $i \in \{1, \ldots, n\}$.

we obtain either has one-input face or is a 2-layered graph. A graph may be partitioned into multiple graphs in one recursive call, but the number of graphs that can be obtained is at most n (the number of nodes in G). A 2-layered CVP can be solved trivially in $\mathsf{CROW}[\log n]$ and we also know that one-input face MPCVP can be solved in $\mathsf{CROW}[\log n]$ [5]. Therefore, we can combine them and obtain $\mathsf{CROW}[\log k \log n]$ bound for evaluating G. Below we give a formal description of this idea.

First, we show how we edge-partition G into L, M and R. Note that when we partition G into many subgraphs, some faces of G may remain faces in one of the subgraphs, and some faces may get divided in this partition. More precisely, when we edge-partition the graph G, edges of a face f of G may appear in different subgraphs of G after the partition. Suppose that G is partitioned into subgraphs and H is one such subgraph. We say that a face of G is *contained* in H (or H *contains* a face of G), if H contains all the edges of the face. Similarly, we say that a face of G is incident on H if H contains some of the edges of that face. Note that a face of G that is contained in H is also incident on H, but a face of G that is incident on H might not be contained in H. Let F' be the set of input faces of G. Graph $G^{(n)}$ (which is nothing but G) contains all the input faces from F' and graph $G^{(0)}$ has a single node (sink t), i.e. contains no input faces of G. Thus, we can say that there exists a positive integer i such that $G^{(i)}$ contains at least $(\frac{k}{2} - 1)$ faces from F'. Let r be the largest integer such that $G^{(r)}$ contains at most $(\frac{k}{2} - 1)$ faces from F'. We define graphs L, M and R as follows.

- We define R to be the graph $G^{(r)}$.
- L is the graph induced by the nodes $V(L) \coloneqq V(G) - V(G^{(r)})$.
- M is the graph induced by the edges: $E(M) \coloneqq \{(u, v) \mid u \notin G^{(r)} \text{ and } v \in G^{(r)}\}$.

We also define two sets of vertices V_1 and V_2 as follows: $V_1 = \{u \mid (u, v) \in E(M)\}$ and $V_2 = \{v \mid (u, v) \in E(M)\}$. Note that $V_1 = V(L) \cap V(M)$, $V_2 = V(M) \cap V(R)$ and $V_1 \cup V_2 = V(M)$.

▶ **Lemma 11.** *Edges of graph M form an edge-cut for graph G.*

Proof. To prove this lemma, it is sufficient to prove that there is no edge $(u, v) \in G$ such that $u \in R$ and $v \in L$. Because all the edges that go from L to R, are already in M. We know that for each node $x \in L$, $d_{\#}(x, t) > r$. Now assume that there edge (u, v) such that u is node in R and v is a node in L. If v is node in L, we know that $d_{\#}(v, t) > r$. From Observation 7, we can say that $d_{\#}(u, t) > d_{\#}(v, t)$. However, we know that for each node y in R, $d_{\#}(y, t) \leq r$. This implies that $d_{\#}(u, t) \leq r \implies d_{\#}(v, t) < r$, which is a contradiction. ◀

From Lemma 11, we can say the edges of M form a cut for G, such that removing the edges of M from G divides it into graphs L and R. Faces in F' can now be divided into three classes: (i) faces which are contained in L, (ii) faces which are contained in R and (iii) faces which are incident to M. Faces of F', which are contained in L and R, remain input faces in L and R, respectively. However, some new input faces may appear on these graphs (for example, see Figure 2). Let us first prove that R has a total of at most $(\frac{k}{2})$ input faces (note that this includes faces from F' as well as the new input faces that appear after partition). From Lemma 8, we know that R is a connected graph. We also know that R contains at most $(\frac{k}{2} - 1)$ faces from F'. Note that the edges of M are incident on the outer face of R. Thus sink nodes of M – nodes in the set V_2 are source nodes of R, and all these nodes lie on the outer face of R. Therefore the outer face of R will also become an input face in R. Therefore, we can conclude that R has at most $(\frac{k}{2})$ input faces. Let H be the graph defined as $H = M \cup R$. We will now prove that each graph in L also has at most $(\frac{k}{2})$ input faces. Before that, we prove the following lemma.

▶ **Lemma 12.** *Graph $G^{(r+1)}$ is a subgraph of the graph H.*

Proof. For the sake of contradiction, assume that $G^{(r+1)}$ is not a subgraph of H. This implies that there exists a node u in $G^{(r+1)}$ such that $u \notin H$. We know that there is a directed path in G from u to the sink node t. Since the edges of M form a cut for graph G, we can say that there exists a node $w \in V_1$ such that this path goes via node w. We know that for each node $v \in V_1$, $d_{\#}(v, t) > r$. Also by Observation 7, we know that $d_{\#}(u, t) > d_{\#}(w, t)$. This implies that $d(w, t) > r$. Hence we can say that $d_{\#}(u, t) > r + 1$. which is contradiction because u is node in $G^{(r+1)}$. ◀

From Lemma 12, we can say that H contains more than $(\frac{k}{2} - 1)$ input faces of G. Since L and $M \cup R$ are edge-disjoint subgraphs, there are at most $\frac{k}{2}$ faces from F' incident on L. In Lemma 11 we proved that there are no edges in G that go from R to L. Thus any source of L is already a source in G and so belongs to a face of F'. We can say that the (number of input faces in L) $\leq$ (number of faces from F' incident to L). We know that the number faces from F' incident to L are at most $\frac{k}{2}$. Therefore the number of input faces in L are at most $\frac{k}{2}$.

Now that we have divided the graph G into three subgraphs L, M and R, such that L and R are planar DAG with at most $\frac{k}{2}$ input faces and M is a 2-layered graph. We can recursively apply the same procedure in L and R. The only problem is that, unlike R, L may have many sink nodes and the algorithm that we have described works with graphs that have only one sink node. Assume that $t_1, t_2, \ldots$ are sink nodes in L. Using Lemma 6 we can obtain graphs $L_1, L_2, \ldots$ such that L_i is the graph induced by the nodes that can reach t_i, in logspace. Then we can apply the same algorithm in each L_i recursively. At the end all the graphs that we obtain are one input face planar DAG or 2-layered. Since there are k input faces in the graph initially, the total number of graphs that we obtain are k. We know that we can evaluate 2-layered circuits in $\mathsf{AC}^0 (\subseteq \mathsf{LogDCFL})$ and one input face MPCVP in $\mathsf{LogDCFL} = \mathsf{CROW}[\log n]$. Notice that these circuits cannot be evaluated independently in parallel because output from one circuit may be an input for another. Therefore we will have to evaluate them sequentially. In order to evaluate the entire circuit represented by G, we need to sequentially compose the evaluations these k circuit. This can be done in $\mathsf{CROW}[k \log n]$ using Lemma 10.

3.2 Monotone Circuit Value Problem in Bounded Genus Graphs

In this section, we will prove that a monotone circuit embedded on a g genus surface such that all the inputs lie on at most k-faces, can be evaluated by CROW-PRAMS in $O(g \log k \log n)$ time using $(gkn)^{O(1)}$ many processors. The approach that we use in this section is similar to the one that we used in the previous section. Given a DAG G representing the monotone circuit embedded on a g-genus surface, we divide it into three subgraphs L, M and R such that L has genus at most $(g - 1)$, M is a 2-layered graph, and R is a planar graph with at most $(g + \frac{3k-1}{2})$ input faces. Now we recursively apply the same procedure with L. In each recursive step, the genus of the resulting graphs decreases at least by one. Thus after g many iterations, we divide the graph G into subgraphs such that each of them is either a planar graph with at most $(g + \frac{3k-1}{2})$ input faces or a 2-layered graph. We know that we can solve the circuit value problem represented by a 2-layered graph trivially in $\mathsf{CROW}[\log n]$. From Section 3.1, we know that we can solve circuit value problem represented by a planar DAG with $(g + \frac{3k-1}{2})$-input faces in $\mathsf{CROW}[\log(g + k) \log n]$. Thus by combining them, we obtain a $\mathsf{CROW}[g \log(g + k) \log n]$ algorithm for evaluating G. We describe the idea formally as follows.

Similar to Section 3.1, given a DAG G of g-genus with one sink node t, we first compute the distance $d_{\#}(v,t)$ for each node in G using Lemma 6 (since we are interested in $(\log n)^{O(1)}$ genus graphs, we can use Lemma 6). Notice that $G^{(n)}$ (which is nothing but G) is a g-genus graph and $G^{(0)}$ contains a single node t therefore is a planar graph. Thus we can say that there exists an integer $i \in [n]$ such that $G^{(i)}$ is a nonplanar graph. Let r be the largest integer such that $G^{(r)}$ is a planar graph. Similar to Section 3.1, we define graphs L, M and R as follows:

- R is the graph $G^{(r)}$.
- L is the graph induced by the nodes $V(L) \coloneqq V(G) - V(G^{(r)})$.
- M is the graph induced by the edges: $E(M) \coloneqq \{(u,v) \mid u \notin G^{(r)} \text{ and } v \in G^{(r)}\}$.

V_1 and V_2 are defined similarly: $V_1 = \{u \mid (u,v) \in E(M)\}$ and $V_2 = \{v \mid (u,v) \in E(M)\}$. Note that $V_1 = V(L) \cap V(M)$, $V_2 = V(M) \cap V(R)$ and $V_1 \cup V_2 = V(M)$. By our assumption, we know that $G^{(r+1)}$ is nonplanar subgraph of G and similar to Lemma 12, we can prove that $G^{(r+1)}$ is a subset of $M \cup R$. Let $H = M \cup R$.

▶ **Lemma 13.** *Graph R has $(g + \frac{3k-1}{2})$ input faces and L has genus at most $(g-1)$.*

We will first prove that L has genus at most $(g-1)$. Let us assume that G is embedded on a surface S of genus g such that the embedding is a 2-cell embedding (we do not need such an embedding explicitly, we just assume that such an embedding exists and use that embedding to prove that L has genus less than g). Let F' be the set of the k-input faces of G. We modify the graph G as follows: we split each node $u \in V_1$ into two nodes u' and u'' such that all the edges of L and $M \cup R$ which were incident on u will now be incident on u' and u'' respectively. Let $V_1' = \{u' \mid u \in V_1\}$ and $V_1'' = \{u'' \mid u \in V_1\}$. If there was edge $\{u,v\}$ in G such that $u, v \in V_1$, we add an edge $\{u', v'\}$ (note that we do not add an edge between nodes u'' and v''). We also add a dummy edge between u' and u'' for all $u \in V_1$. L, M and R still represent the same subgraphs as earlier (except now L and H use node sets V_1' and V_1'' respectively instead of V_1). Let f be a face that contains nodes from both sets V_1' and V_1''. We split f into faces $f_1, f_2, \dots$ by adding dummy edges of $\{u', v''\}$ where $u' \in V_1'$ and $v'' \in V_1''$, inside f such that each f_i contains exactly two nodes of V_1''. Let E_1 be set of all edges $\{u', v''\}$ such that $u' \in V_1'$ and $v'' \in V_1''$. Remember that we are doing all these constructions so that we can prove that L has genus $(g-1)$. We do not do these constructions as a part of our final algorithm. Let G_m be this new graph. We can see that the embedding of G_m is also a 2-cell embedding on S. Before splitting the nodes of V_1, we had that $V_1 = V(L) \cap V(H)$. Therefore, we can say that the set of edges in E_1 forms a cut for G_m after the split.

Now consider the dual graph G_m^* of G_m with respect to its embedding on surface S. Each face in G_m becomes a node in G^* and vice-versa. Also, there is a one-to-one correspondence between the edges of G_m and G_m^*. Let E_1^* be the set of dual edges corresponding to edges in E_1 and F_1 be the set of faces in G_m that contain edges of E_1'. Let $f \in F_1$ be a face and f^* be the corresponding nodes in G_m^*. By our construction, we know that f contains exactly two edges of E_1. Thus f^* has exactly two edges of E_1^* incident on it. Therefore, we can say that the subgraph of G_m^* induced by edges in E_1^* must be a collection of node-disjoint cycles. Let $\mathcal{C}^*$ be the set of these dual cycles. The following lemma is standard. However, for the sake of completion we provide a proof here.

▶ **Lemma 14.** *Cutting the surface S along cycles in $\mathcal{C}^*$ divides the surface into two or more surfaces.*

Proof. (Extracted from [15]) We assume that the graph is 2-cell embedded that is all faces of G are topological disks. After cutting along C^*, each face is a topological disk bounded

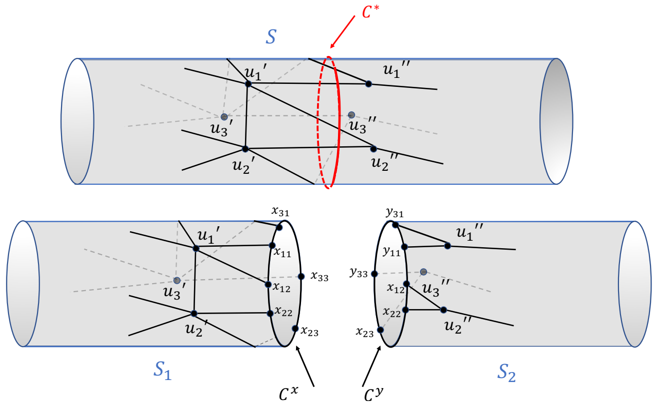

Figure 3 Surface S is divided into two surfaces S_1 and S_2, by cutting it along the dual cycle $C^* \in \mathcal{C}^*$ (in red). C^x and C^y are the corresponding primal cycles created by this cutting that become facial cycles in the respective surfaces.

by either a cycle of G, or a cycle that consists of two arcs (one on G and one on C^*) with common endpoints (at the intersection between edges of C and their duals), where the arc on C^* lies on the boundary of the cut surface. In particular, the boundary of any face intersects G in a single component.

Now, assume for a contradiction that the surface after cutting is still connected, and consider an edge $(u, v) \in C$. Then there exists a path π on the cut surface connecting u and v. By the above property of faces, we can snap π to a path on $G - C$, showing that u and v lie in the same component of $G - C$. Therefore, $C - (u, v)$ is still an edge cut for G, contradicting minimality of C and hence connectedness of the cut surface. ◀

We know that there is a one-to-one correspondence between the edges of G_m and G_m^*. Let C^* be a cycle in $\mathcal{C}^*$ that contains dual edges $e_1^*, e_2^*, \dots e_t^*$ in some cyclic order. Let $e_i = \{u_i', u_i''\}$ be the respective primal edges in G_m, in the same cyclic order. Note that when we cut the surface S along a dual edge e_i^*, the respective primal edge gets divided in two edges. Let us assume that edge $\{u_i', u_j''\}$ is divided into two edges $\{u_i', x_{ij}\}$ and $\{y_{ij}, u_i''\}$ when we cut S along C^* (see Figure 3). Let C^x be the primal cycle that is obtained by adding dummy edges among the nodes x_{ij} and C^y be the the primal cycle that is obtained by adding the edges among the nodes y_{ij}, corresponding to cycle C^*. We cap the holes that appear on the surface (or the surfaces if C^* is a separating cycle) after we cut S along C^*. We do this for all the cycles in $\mathcal{C}^*$. Let L' be the graph that contains all the edges of L along with the edges $\{u_i', x_{ij}\}$ and the edges of C^x, for all $C^* \in \mathcal{C}^*$. Similarly, let H' be the graph that contains all the edges of H along with the edges $\{y_{ij}, u_i''\}$ and the edges of C^y, for all $C^* \in \mathcal{C}^*$.

We know that cutting the surface along the cycles in $\mathcal{C}^*$ divides the surface into two or more surfaces. Let $S_1, S_2, \dots$ be the surfaces obtained after cutting S along $\mathcal{C}^*$ and capping the holes. Since H' is a connected graph, it remains embedded on one of the surfaces, say S_1. However, L' may contain many connected components that are embedded on different surfaces.

▶ **Lemma 15.** *Graph L' has genus at most $(g-1)$.*

Proof. We divide the analysis into the following two cases:

- $\mathcal{C}^*$ contains only surface separating cycles: By Lemma 3, we know that if we cut the surface S along a surface separating cycle, then the sum of the genus of the surfaces obtained by cutting S along the cycle (and capping the holes) must be equal to the genus of S. If all the cycles in $\mathcal{C}^*$ are surface separating then we can say that there exists a cycle $C^* \in \mathcal{C}^*$ such that cutting S along C^* separates S_1 from the rest of the surface. Since H' is a nonplanar graph that is embedded on S_1, we can say that genus of $S_1 \geq 1$. Therefore, the genus of the remaining surface (on which L' is embedded) after cutting S along C^* and capping holes is at most $(g-1)$. Hence in this case genus of L' is at most $(g-1)$.
- $\mathcal{C}^*$ contains a surface non-separating cycle: If it contains a surface non-separating cycle C^* then from Lemma 4 we know that surfaces obtained by cutting surface S along C^* and capping the holes will have genus smaller than that of S. Therefore in this case also, L' has genus most $(g-1)$.

◀

Since graph L is a subgraph of L', we can conclude that L has genus at most $(g-1)$. Now we turn to prove the second part of Lemma 13 that R has at most $(g + \frac{3k-1}{2})$ input faces. We know that unlike H', L' may have many connected components. Nevertheless, we know that each connected component of L' must contain at least one of the initial k input faces (because for a node w in L', every predecessor of w in G must be in the same connected component of L' as w). Therefore we can say that L' has $l \leq k$ connected components. We will first prove the following lemma.

▶ **Lemma 16.** *$\mathcal{C}^*$ contains at most $g + \frac{(k-1)}{2}$ cycles.*

Proof. By Euler's formula for surface embedded graphs, we know that

$$\#v(H') - \#e(H') + \#f(H') = 2 - 2\mathrm{g}(H') \tag{1}$$
$$\#v(L') - \#e(L') + \#f(L') = 1 + l - 2\mathrm{g}(L'). \tag{2}$$

For each dual cycle $C^* \in \mathcal{C}^*$, we create two cycles C^x and C^y. Let n_{C^x} and n_{C^y} be the total number of nodes in cycle C^x and C^y. Let us define $n_C := n_{C^x} = n_{C^y}$. Therefore,

$$\#v(H') + \#v(L') \quad = \#v(G) + \textstyle\sum_{C^* \in \mathcal{C}^*} 2n_C, \tag{3}$$

also, while cutting the surface along C^*, we destroy some edges of G and some create new edges. For each destroyed edge $\{u'_i, u''_j\}$we create four new edges $\{u'_i, x_{ij}\}, \{y_{ij}, u''_j\}, \{x_{ij}, x_{pq}\}$, and $\{y_{ij}, y_{st}\}$, for some nodes x_{pq} and y_{st} that lie on cycles C^x and C^y, respectively (see Figure 3. Therefore,

$$\#e(H') + \#v(L') \quad = \#e(G) + \textstyle\sum_{C^* \in \mathcal{C}^*} 3n_C. \tag{4}$$

Note that cycles C^x and C^y become faces in L' and H', respectively. In addition, while cutting along C^* we destroy n_C faces of G and create n_C faces in both H' and L'. Therefore,

$$\#f(H') + \#f(L') = \#f(G) + |\{C^x : C^* \in \mathcal{C}^*\}| + |\{C^y : C_j \in \mathcal{C}^*\}| + \sum_{C^* \in \mathcal{C}^*} n_C$$
$$\Rightarrow \qquad \#f(H') + \#f(L') = \#f(G) + 2|\mathcal{C}^*| + \sum_{C^* \in \mathcal{C}^*} n_C \tag{5}$$

From these equations, we can conclude that,

$$\begin{aligned} & 2 - 2\mathrm{g}(H') + 1 + l - 2\mathrm{g}(L') = \#v(G) - \#e(G) + \#f(G) + 2|\mathcal{C}^*| \\ \Rightarrow \quad & 2 - 2\mathrm{g}(H') + 1 + l - 2\mathrm{g}(L') = 2 - 2\mathrm{g}(G) + 2|\mathcal{C}^*| \\ \Rightarrow \quad & 2\mathrm{g} - 2(\mathrm{g}(H') + \mathrm{g}(L')) + l - 1 = 2|\mathcal{C}^*| \\ \Rightarrow \quad & |\mathcal{C}^*| \leq g + \frac{(k-1)}{2} \end{aligned}$$

◀

Proof of Lemma 13. Graph R may have at most k initial input faces of G i.e. faces from the set F'. Also, there may appear at most one input face in R with respect to each cycle C^y (see Figure 3) such that $C^* \in \mathcal{C}^*$. This implies that R can have at most $(g + \frac{3k-1}{2})$ input faces. This along with Lemma 15 completes the proof of Lemma 13. ◀

Overall Evaluation: Since R is a planar DAG with only one sink node, from Section 3.1 we know that it can be evaluated in $\mathsf{CROW}[\log(k+g)\log n]$. Notice that L can have multiple sink nodes. Let $t_1, t_2, \ldots t_l$ be sinks in L. For each t_i we can compute the graph consisting of nodes that can reach t_i say L_i using Lemma 6 in L. Using the same argument that we use in Section 3.1, we can say that no new input face appears in L after partition. Therefore, each L_i has at most k input faces and one sink node. We can recursively apply the same procedure discussed in this section to further decompose each L_i until each obtained subgraph is planar. To summarise, we begin with a graph with genus g and with each recursive step, the graphs we obtain are either graphs of genus at most $(g-1)$ with at most $(g + \frac{3k-1}{2})$ input faces or 2-layered. Therefore, after g many recursive steps all the subgraphs that we obtain are planar with at most $(g + \frac{3k-1}{2})$ input faces or 2-layered. We can evaluate each such graph using CROW-PRAMS in time $O((k+g)\log n)$ with $(n)^{O(1)}$ many processors. Since depth of the recursion is g, overall evaluation can be done in $\mathsf{CROW}[g(k+g)\log n]$. Notice that we assume that g is polylogarithmically bounded. In particular, for bounded g and k, we can evaluate the circuit in $\mathsf{CROW}[\log n]$, equivalently in $\mathsf{LogDCFL}$.

4 Conclusion and Open Questions

We show that the $\mathsf{LogDCFL}$ bound of [5] for single input face monotone circuit value problem, that builds on [19, 3], can be extended to monotone circuits with constantly many input faces and embedded on a surface of bounded genus. Further, for a monotone circuit with polylogarithmically many input faces that is embedded on a surface of polylogarithmic genus we show an NC bound. We leave a further improvement of this bound as our main open question.

In this work we have dealt with only orientable surfaces. Extending the results to circuits embedded on non-orientable surfaces is yet another open question.

It is known from [5] that single input face MPCVP is L-hard. Improving this lower bound to $\mathsf{LogDCFL}$ is our other open question.

References

1. Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and grid graph reachability problems. *Theory Comput. Syst.*, 45(4):675–723, 2009. doi:10.1007/s00224-009-9172-z.
2. Eric Allender and Meena Mahajan. The complexity of planarity testing. *Inf. Comput.*, 189(1):117–134, 2004. doi:10.1016/j.ic.2003.09.002.

3 David A. Mix Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. On monotone planar circuits. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, page 24, 1999. doi:10.1109/CCC.1999.766259.

4 Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Trans. Comput. Theory*, 1(1):4:1–4:17, 2009. doi:10.1145/1490270.1490274.

5 Tanmoy Chakraborty and Samir Datta. One-input-face MPCVP is hard for L, but in LogDCFL. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*, volume 4337 of *Lecture Notes in Computer Science*, pages 57–68. Springer, 2006. doi:10.1007/11944836_8.

6 Stephen A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 338–345, 1979. doi:10.1145/800135.804426.

7 Arthur L. Delcher and S. Rao Kosaraju. An NC algorithm for evaluating monotone planar circuits. *SIAM J. Comput.*, 24(2):369–375, 1995. doi:10.1137/S0097539792226278.

8 Reinhard Diestel. *Graph Theory*. Springer Publishing Company, Incorporated, 5th edition, 2017.

9 Patrick W. Dymond and Stephen A. Cook. Complexity theory of parallel time and hardware. *Inf. Comput.*, 80(3):205–226, 1989. doi:10.1016/0890-5401(89)90009-6.

10 Patrick W. Dymond and Walter L. Ruzzo. Parallel rams with owned global memory and deterministic context-free language recognition. *J. ACM*, 47(1):16–45, 2000. doi:10.1145/331605.331607.

11 Michael Elberfeld and Ken-ichi Kawarabayashi. Embedding and canonizing graphs of bounded genus in logspace. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 383–392, 2014. doi:10.1145/2591796.2591865.

12 Leslie M. Goldschlager. The monotone and planar circuit value problems are log space complete for P. *SIGACT News*, 9(2):25–29, 1977. doi:10.1145/1008354.1008356.

13 Leslie M. Goldschlager. A space efficient algorithm for the monotone planar circuit value problem. *Inf. Process. Lett.*, 10(1):25–27, 1980. doi:10.1016/0020-0190(80)90117-9.

14 Chetan Gupta, Vimal Raj Sharma, and Raghunath Tewari. Reachability in $O(\log n)$ genus graphs is in unambiguous logspace. In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, pages 34:1–34:13, 2019. doi:10.4230/LIPIcs.STACS.2019.34.

15 Tim (https://cstheory.stackexchange.com/users/27113/tim). Dual of cut of embedded graph disconnects surface. Theoretical Computer Science Stack Exchange. URL:https://cstheory.stackexchange.com/q/51936 (version: 2022-09-28).

16 Richard M. Karp and Vijaya Ramachandran. Parallel algorithms for shared-memory machines. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 869–942. 1990.

17 Jan Kyncl and Tomás Vyskocil. Logspace reduction of directed reachability for bounded genus graphs to the planar case. *ACM Trans. Comput. Theory*, 1(3):8:1–8:11, 2010. doi:10.1145/1714450.1714451.

18 Nutan Limaye, Meena Mahajan, and Prajakta Nimbhorkar. Longest paths in planar DAGs in unambiguous log-space. *Chic. J. Theor. Comput. Sci.*, 2010, 2010. URL: http://cjtcs.cs.uchicago.edu/articles/CATS2009/5/contents.html.

19 Nutan Limaye, Meena Mahajan, and Jayalal Sarma. Upper bounds for monotone planar circuit value and variants. *Comput. Complex.*, 18(3):377–412, 2009. doi:10.1007/s00037-009-0265-5.

20 Pierre McKenzie, Klaus Reinhardt, and V. Vinay. Circuits and context-free languages. In *Computing and Combinatorics, 5th Annual International Conference, COCOON '99, Tokyo, Japan, July 26-28, 1999, Proceedings*, pages 194–203, 1999. doi:10.1007/3-540-48686-0_19.

21 Vijaya Ramachandran and Honghua Yang. An efficient parallel algorithm for the general planar monotone circuit value problem. *SIAM J. Comput.*, 25(2):312–339, 1996. doi:10.1137/S0097539793260775.

22 Vijaya Ramachandran and Honghua Yang. An efficient parallel algorithm for the layered planar monotone circuit value problem. *Algorithmica*, 18(3):384–404, 1997. doi:10.1007/PL00009162.

23 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, 2008. doi:10.1145/1391289.1391291.

24 Walter L. Ruzzo. On uniform circuit complexity. *J. Comput. Syst. Sci.*, 22(3):365–383, 1981. doi:10.1016/0022-0000(81)90038-6.

25 Larry J. Stockmeyer and Uzi Vishkin. Simulation of parallel random access machines by circuits. *SIAM J. Comput.*, 13(2):409–422, 1984. doi:10.1137/0213027.

26 Derrick Stolee, Chris Bourke, and N. V. Vinodchandran. A log-space algorithm for reachability in planar acyclic digraphs with few sources. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 131–138, 2010. doi:10.1109/CCC.2010.36.

27 Derrick Stolee and N. V. Vinodchandran. Space-efficient algorithms for reachability in surface-embedded graphs. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 326–333, 2012. doi:10.1109/CCC.2012.15.

28 Ivan Hal Sudborough. On the tape complexity of deterministic context-free languages. *J. ACM*, 25(3):405–414, 1978. doi:10.1145/322077.322083.

29 Raghunath Tewari and N. V. Vinodchandran. Green's theorem and isolation in planar graphs. *Inf. Comput.*, 215:1–7, 2012. doi:10.1016/j.ic.2012.03.002.

30 Thomas Thierauf and Fabian Wagner. The isomorphism problem for planar 3-connected graphs is in unambiguous logspace. *Theory Comput. Syst.*, 47(3):655–673, 2010. doi:10.1007/s00224-009-9188-4.

31 Burchard von Braunmühl, Stephen A. Cook, Kurt Mehlhorn, and Rutger Verbeek. The recognition of deterministic CFL's in small time and space. *Inf. Control.*, 56(1/2):34–51, 1983. doi:10.1016/S0019-9958(83)80049-7.

32 Honghua Yang. An NC algorithm for the general planar monotone circuit value problem. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing, SPDP 1991, 2-5 December 1991, Dallas, Texas, USA*, pages 196–203, 1991. doi:10.1109/SPDP.1991.218279.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il