The Computational Complexity Column

by

V. Arvind

Institute of Mathematical Sciences, CIT Campus, Taramani
Chennai 600113, India
arvind@imsc.res.in
http://www.imsc.res.in/~arvind

The design of efficient algorithms for planar graphs, as a field of research, is over forty year old and continues to be an exciting area. There are several new efficient algorithms for a variety of graph optimization problems that exploit planarity and, in general, the structure of low genus graphs and graphs with excluded minors.

Efficient *space-bounded* computations, on the other hand, involve a different genre of algorithmic ideas and exploit the planar graph structure somewhat differently. There has been a flurry of interesting research on this topic in last decade. In this timely and well-written article, Samir Datta and Raghav Kulkarni focus on planar graphs in the context of space complexity. They discuss several properties of planar graphs and their precise role in the design of space efficient algorithms.

Space complexity: What makes planar graphs special?

Samir Datta* Raghav Kulkarni†

1 Introduction

The purpose of this article is to survey several useful properties of planar graphs that can be exploited specifically in the context of space bounded computation to obtain efficient algorithms. For completeness we also point out some situations where planar restrictions remain computationally as hard as general graphs.

The classes we will encounter in this survey are mostly variants of deterministic logarithmic space, L. They include non-deterministic logarithmic space, NL, Unambiguous Log-space UL, Stoic Probabilistic Log-space SPL and counting classes like $\mathsf{GapL}, \oplus\mathsf{L}$. For a definition of these classes see e.g. [32]. We will also have occasion to compute *functions* in Log-space. We say that a function is computable in Log-space if there is a Log-space transducer, i.e. a deterministic logarithmic space machine with a write-only one-way output tape on which it writes the function value.

Definition 1. *A graph is said to be planar if there is a mapping of the vertices on the plane such that the edges can be drawn as curves in the plane, which do not intersect except at their end points.*

We are going to make repeated use of the following algorithmic version of planarity:

Theorem 2 (Allender,Mahajan[3],see also [15]). *There is a Log-space algorithm for constructing an embedding of a planar graph on the plane.*

We will also make use of the notion of *grid* embedding of a graph. This is a map which identifies a graph with a subgraph of a complete rectangular grid (i.e. a rectangular array of nodes in which a node is connected to the vertices immediately next to it horizontally and vertically). This is done by a degree reduction step, (by replacing a vertex with a binary tree or a cycle) followed by embedding the vertices so that that the edges of the degree reduced graph get elongated into paths. The crucial point is that this preserves some property of the digraph or graph e.g. reachability [1] or number of perfect matchings [10] and runs in Log-space.

In the rest of the paper, in every section, we mention a property (or a collection of properties) of planar graphs and then give a brief description of how the property helps us develop a bounded space algorithm or hardness result. We should mention at the outset

*Chennai Mathematical Institute, India email:sdatta@cmi.ac.in
†Centre for Quantum Technologies, Singapore email:kulraghav@gmail.com

that this survey does not aim at completeness of any sort - it aims at providing a taste of the planarity magic through a number of examples drawn according to our preferences and experience.

2 Unique Embedding

The first property of planar graphs that we will exploit is the property of unique embeddability of 3-connected planar graphs, in the plane. This remarkable property which was first shown by Whitney, way back in 1933, allows us to give a bounded space algorithm *canonical description* for such highly connected planar graphs. We should point out it is notoriously difficult to prove good upper bounds *or* lower bounds for the general graph isomorphism problem. In contrast, planarity enables us to prove matching upper and lower space complexity bounds.

The first bounded space algorithm that made use of Whitney's theorem for planar isomorphism was by Thierauf, Wagner [29] based on distance computation in planar graphs. This was soon followed up by an algorithm [13] (which we describe below) that achieved the optimal Log-space bound by replacing the distance computation by an exploration from Reingold's seminal paper[26]. The result was later extended [14] to arbitrary planar graphs through a careful application of Lindell's Log-space Tree Canonization algorithm to the tree decomposition of the given planar graph into its 2-connected and 3-connected components.

Whitney's Unique Embedability Theorem

We will make crucial use of the following theorem by Whitney

Theorem 3. *(Whitney[33]) A 3-connected*[1] *planar graph has exactly one combinatorial embedding on the plane (up to reflection).*

The theorem can be rephrased as: in any planar drawing - the cyclic order of the edges leaving a vertex in, say, the clockwise order, is identical (except, of course that we may reflect the graph and get another one in which the order of edges is reversed). As an example consider the 3-connected planar graph Figure 2 (a) with its two embeddings. Figure 2 (b) illustrates the necessity of 3-connectivity for unique embeddability.

Unique Embedding and 3-connected Planar Graph Isomorphism

Given a planar graph, we will compute, using a Log-space transducer, a binary string that is independent of the presentation of the graph (in terms of the vertex and edge names) and uniquely characterizes the graph. Thus, if $can(G)$ denotes the canonical string for$G$, $G_1 \cong G_2$ if and only if, $can(G_1) = can(G_2)$. Testing for isomorphism is then easy - just go through the canonical descriptions bit by bit and verify that they match.

The rough idea behind the algorithm is to construct a canonical ordered spanning tree - a rooted tree that does not depend on the presentation of the graph and where the

[1] removing any two vertices from the graph does not disconnect it

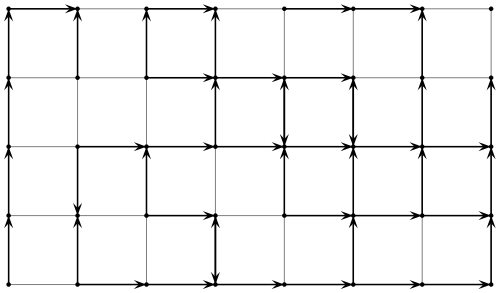
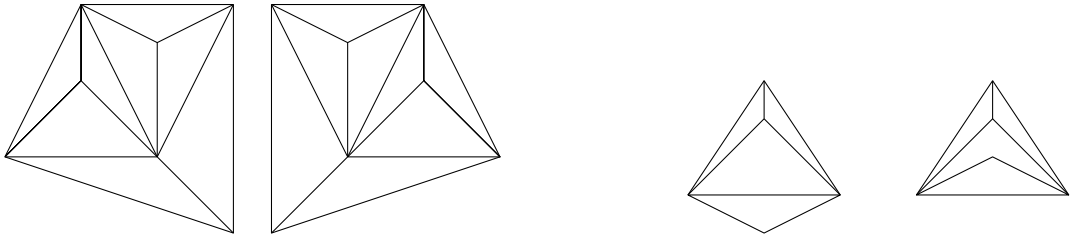


Figure 1: A Directed Grid Graph



(a) A 3-connected graph and its reflection (b) Distinct embeddings of a 2-connected graph

Figure 2: Embeddings of planar graphs

children of a node are ordered. Now it is easy to describe the graph in terms of the tree. First the vertices of the graph can be canonically labeled by the pre-order numbering of the tree. Next the edges can be listed in lexicographic order of the numbers of their endpoints (with,say, the smaller endpoint first).

Thus we have to show how to produce a canonical spanning tree. We show something slightly weaker, that is, we produce a collection of spanning trees which depends only on the choice of the "root edge" and the choice of one of the two embeddings. We can "canonize" the graph relative to each of the trees as described above and then pick the minimum description as our final canon.

For simplicity assume that the graph has degree exactly 3 - this is easy to ensure by a simple degree reduction step.

We will need to introduce the concept of universal exploration sequence UXS to describe how to obtain the tree above. A UXS σ_n is a string (of length polynomial in n) from $\{0,1,2\}^*$ which encodes a way to explore a cubic graph which is equipped with a cyclic ordering on the neighbours of each vertex. We will explore the graph one edge at a time. At the i^{th}-step we will be sitting on an (directed) edge (u,v), having just visited u and being about to visit v. The next edge will be some (v,w) depending on the letter $\sigma_n[i]$ which tells us the *offset* of the next edge relative to the current one : an offset of zero means $w = u$ and offsets $1,2$ indicate the the first/second neighbours of v after u in the prescribed cyclic order. Somewhat mysteriously the universality of σ_n implies that given any cubic graph of size n with an ordering on its vertices, an exploration guided by σ_n will lead us to visit every vertex in the graph. It is easy to construct UXS's using randomization. The magic of Reingold's construction [26] is that

- a UXS can in fact be constructed in Log-space!

A planar embedding of a cubic graph naturally induces a cyclic ordering on the neighbours of a vertex - say the clockwise order. Whitney's theorem tells us there are exactly two such cyclic orders. The magic of planarity meets that of Reingold, at this point. Thus, given a fixed UXS churned out by Reingold's algorithm and two choices:

- the directed edge to start the exploration from, and
- one of the two possible embeddings,

there is a fixed way to visit each vertex in the graph. By considering the edge of first visit to a vertex we can produce a spanning tree from this exploration. It is important to note that:

- the tree produced is canonical relative to the two choices above (and a fixed UXS)
- it can be computed in Log-space since a transducer can output the parent edge of each vertex by repeating the exploration multiple times

This is all that is needed to prove the following:

Theorem 4. *Isomorphism in* 3*-connected planar graphs can be tested in* L.

3 Duality and Flows in Planar Graphs

The next property which we will consider is duality. For this we need to define the dual of a graph G. Informally, the nodes of the dual consist of the faces of G and two nodes are joined by an edge whenever the corresponding faces share an edge. We use this correspondence crucially in our next application viz. computing flows in planar networks.

We consider the problem of determining if there is a feasible flow in a network where there are both demands at nodes (i.e. the amount of flow that the vertex needs - this may be positive or negative) and capacity constraints at edges (i.e. the maximum amount of flow possible through an edge). Notice that the graph is bidirected i.e. there is an edge in either direction for every undirected edge and each of these edges have capacities of their own.

Many graph theory problems can be recast in this framework, e.g. bipartite matching. We will of course restrict our attention to the case when the network is planar. The approach we are going to delineate is based on a paper by Miller and Naor [23] which proposes a parallel algorithm for the problem.

Cut-Cycle duality in Planar Graphs and Miller-Naor's Algorithm

We first reduce the problem with edge capacities and vertex demands to one in which the vertex demands are all zero and edge capacities are appropriately modified. Zero demands imply that the flow across any cut is zero. Thus if the sum of capacities of edges in any directed cut are strictly negative we have a contradiction to the actual flow being zero i.e. the flow is infeasible. It is not too hard to prove that this sufficient condition for

infeasibility is also necessary. But finding if a graph has negative cuts seems to be a hard problem. We can of course reduce the problem to finding a minimal cut[2] that is negative.

This is where the planarity magic comes into play:

- Minimal cuts in the primal graph correspond to cycles in the dual.

Thus with appropriately weighed dual edges, the problem reduces to detecting a negative cycle in the dual. But there exists a negative cycle in a graph if and only if there exists a negative closed walk. And detecting the existence of a negative closed walk is easy in NL - just start from a vertex keeping and non-deterministically walk along the graph keeping track of only the current vertex and the sum of weights encountered so far (along with the initial vertex). If the initial vertex is reached with a negative sum within polynomial number of steps then accept else reject.

The idea can be refined to find the flow if indeed the dual graph has no negative cycles. The intuitive idea being that since there are no negative cycles shortest distance between any pair of dual nodes is well defined. Miller and Naor [23] show these dual shortest distances can in fact be converted to a primal feasible flow.

Space Complexity of Miller-Naor's algorithm

The basic ingredients needed in the proof are:

- a planar embedding of the graph
- a spanning tree algorithm (for reduction to the zero-demand case)
- an algorithm to detect a negative cycle in a (planar) digraph
- a directed shortest path algorithm in a (planar) digraph with no negative cycles

The first two are known to be in deterministic Log-space through non-trivial algorithms [3, 15, 26] while the latter two are easily seen to be reducible to reachability i.e. in NL.

4 Duality and Deterministic Isolation in Planar Graphs

Mulmuley, Vazirani and Vazirani's isolation lemma informally states that a uniform random assignment of small weights to the elements of a universe makes the weight of the minimum weight set of any given set system, unique. Derandomizing the lemma in general is not possible [2] but for several set systems arising from planar graphs, there exists a simple deterministic weighing scheme that isolates the min weight set [1, 7, 12]. This deterministic isolation can then be translated to a space efficient algorithm for finding the min-weight set using known extraction procedures like [25, 4]. Our focus in this article will be on the isolation rather than the extraction part of the algorithm, since it is the former that makes crucial use of planarity.

In describing these algorithms it is often convenient for the purposes of visualization to first embed the graph in a a 2-d rectangular grid, after a degree reduction step. This

[2] a minimal set of edges removing which disconnects the graph

is how the algorithms [7, 12] historically came about. But it is possible to work directly on the graph (and its dual) rather than invoke grid embedding [28, 8]. In this survey we will stick to grid embedding for the ease of exposition.

Deterministic Isolation in Planar Structures

We will consider two important set systems in this article:

- the system of directed s,t-paths in a grid-digraph G, where s,t are vertices of G
- the system of perfect matchings in a grid-graph G.

A path or matching will be regarded as a set of edges.

We want to prescribe a weighing scheme for the grid edges such that the min-weight set is unique. For technical reasons we will insist that the weighing scheme in the first case is skew-symmetric i.e. reversing an edge just flips the sign of the weight.

By definition, if we have two distinct min-weight sets S_1, S_2 then their weights must be equal. Let us first consider the case of s,t-reachability. If we reverse the edges of,say S_2 we get a closed walk of zero weight. An easy substitution argument shows that every simple cycle C contained in this closed walk must have zero total weight.

We follow a similar argument in the case of perfect matchings. The symmetric difference $S_1 \oplus S_2$, of two perfect matchings is exactly a collection of disjoint cycles. Thus an easy substitution argument again shows that the difference $w(S_1 \cap C) - w(S_2 \cap C) = 0$ for any of these cycles C. Because there is strict alternation between edges of S_1, S_2 in C this is equivalent to saying that the alternating sum of each cycle C is zero.

As a notational convenience we have the following definitions of *circulation* for reachability and matching. Let circulation of a simple cycle be

- the sum of weights of the (directed) edges of the cycle in case of reachability
- the alternating sum (which is well-defined up to its sign) of weights of an undirected simple cycle in the case of matching

Thus the two paragraphs prior to the definition can be summarized as: distinct min-weight sets imply zero circulation for some simple cycle. Contrapositively, if we can produce a weighing scheme such that all simple cycles in the *complete grid* have non-zero circulations then the min-weight set for *any grid graph* will be unique. We now proceed to construct such a weighing scheme.

In order to do so we again invoke planarity magic and ensure that:

- the circulation of a (simple) cycle equal in magnitude to the area enclosed by the cycle.

This is easy to argue for reachability, suppose we could assign skew symmetric weights to the edges of a unit square such that they sum to 1 in the clockwise orientation. Then if we consider two adjacent unit squares both oriented clockwise, skew symmetry ensures that their circulations adds up to 2 since the common edge cancels out. Inductively, it is easy to see that any clockwise cycle will get a circulation equal to its area (and counter-clockwise cycle will have a circulation equal to the negative of its area).

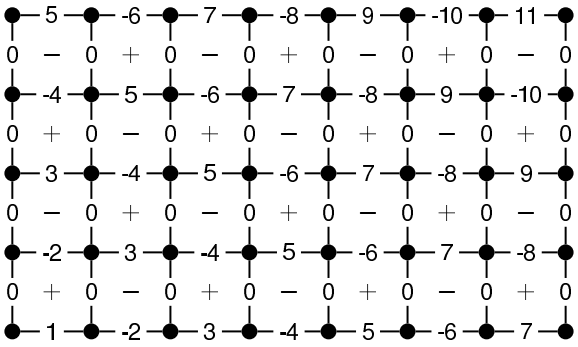


Figure 3: The weighing scheme and signs for isolating perfect matching in grid graphs

By a somewhat more complicated argument we can ensure that the (matching) circulation of a cycle in the grid equals the *signed* sum of circulations of the squares enclosed by it. The sign of square being a quantity that flips between adjacent squares and the bipartiteness of the dual grid permits us such a function. Now we need to make sure that the circulation of every square equals its sign which ensures that the circulation of a simple cycle equals its signed area.

Two simple sets of linear equations reveal what assignments of weights will work - we illustrate this for perfect matchings in Figure 4.

Isolation and Extraction in Planar Graphs

We have sketched the proof of the following result by Bourke, Tewari and Vinodchandran [7]: Let G be a sub-digraph of the $n \times n$ square grid. There is a constant c and a Log-space computable weighing function $w : E \to [-cn, cn]$ under which the min-weight s,t-path is unique (if it exists) for any two vertices $s, t \in V(G)$. As a consequence, due to the extraction procedure in Reinhardt and Allender [25], s,t-reachability in planar digraphs is in $\mathsf{UL} \cap \mathsf{co} - \mathsf{UL}$.

The following result by Datta, Kulkarni and Roy [12] mirrors the previous one for planar bipartite perfect matchings: Let G be a sub-graph of the $n \times n$ square grid. There is a constant c' Log-space computable weighing function $w' : E \to [-c'n, c'n]$ under which the min-weight perfect matching of G is unique (if it exists). As a consequence, due to the extraction procedure in Allender, Reinhardt and Zhou [4], finding a perfect matching in planar bipartite graphs is in SPL.

5 Bounded Local Tree Width

The concept of *tree-width* of graphs has turned out to be very useful for designing efficient algorithms. See for instance [17]. Although the tree-width of planar graphs could be arbitrarily large, they enjoy a weaker property that is equally useful, namely that of having bounded *local* tree width, i.e., if we do breadth first search (BFS) on a planar graph starting from any given vertex then the tree-width of any constant number of

BFS layers is bounded above by a constant. This weaker property suffices to obtain fast algorithms for several problems.

Baker's Algorithm

In a well-known paper [5], Baker first exploited the bounded local tree width property of planar graphs to give an approximation scheme ($(1+\epsilon)$ approximation algorithm for every $\epsilon > 0$) for a variety of problems (for instance Max-Cut) that are hard to approximate in general graphs. The basic idea of Baker's algorithm is as follows:

- (Step 1) Remove some edges from the graph so that:
  - the optimum solution is not affected by much: say it is affected only by ϵ fraction
  - the rest of the edges can be partitioned into disjoint union of graphs whose tree-width is a constant that depends on the approximation guarantee, i.e., ϵ
- (Step 2) Solve the problem on the bounded tree width graphs using dynamic programming and combine the solution on the disjoint pieces to obtain a solution in the original graph.

Space Complexity of Baker's Algorithm

In fact, for Step 1, Baker observes that the vertices at any fixed distance from the root form an outer-planar graph. Thus any k consecutive layers constitute k-outer-planar graphs, which happen to have tree-width at most $O(k)$.

Thus: the efficiency of Baker's algorithm relies mainly on the following two components:

- (Component 1) distance computation in planar graphs,
- (Component 2) exact solution on bounded tree-width graphs.

The first component is crucial for obtaining the BFS layers, which in turn help in obtaining the desired decomposition of the planar graph into graphs of bounded tree width. The second component gives an efficient algorithm for solving the problem exactly on the bounded tree width graphs. This is achieved via a dynamic programming approach. The bounded tree-width property guarantees the efficiency of the dynamic programming.

The exact solution on bounded tree-width graphs more or less directly combine to give an approximate solution in the original graph. In the end, one has to argue that the solution obtained after Step 2 is good enough, i.e., the removed edges do not affect the optimal solution by much. This is guaranteed by the delicate choice of edges in Step 1.

In the context of the space complexity, Datta and Kulkarni [9] observe that both these components are less expensive than their counter-parts in general graphs!

For the first component, we can use an elegant result by Bourke, Tewari, and Vinodchandran [7] showing that the directed planar reachability is in $\mathsf{UL} \cap \mathsf{co} - \mathsf{UL}$ as opposed to the directed reachability in general graphs being NL-complete. As a consequence, BFS

in planar graphs can be performed in $\mathsf{UL} \cap \mathsf{co}-\mathsf{UL}$ The exact space complexity of BFS in planar graphs is still not known. It is still possible that in fact BFS in planar graphs could be performed as efficiently as Log-space! The current best known upper bound in this context is (deterministic) $O(\log^2 n)$ space, which follows from the famous Savitch's Theorem [27] that the class NL is contained in $O(\log^2 n)$ space. This means that the decomposition of planar graphs into bounded tree width graphs can performed in a space efficient way.

As for the second component in Baker's algorithm, a recent theorem of Elberfeld et. al. [17] comes to our help. They show that any optimization problem *expressible in monadic second order logic* can be solved in Log-space on bounded tree width graphs. It turns out that many optimization problems (e.g. Max-Cut) can be expressed in MSO. Thus the second component of Baker's algorithm can be made space efficient for problems like Max-Cut.

This simple combination gives space efficient algorithms for some optimization problems in planar graphs. The best bound on space complexity that one can currently obtain via this approach is $O(\log^2 n)$ space. It is an interesting open question whether this space bound can be further improved possibly using some other type of decomposition. For instance: does Max-Cut in planar graph have a Log-space approximation scheme ?

6 Algebraic Properties

The adjacency matrix of an undirected graph G on n vertices is an $n \times n$ symmetric matrix A, i.e., $A(i,j) = A(j,i)$ and whose entries are $A(i,j) = 1$ if and only if (i,j) is an edge in G; $A(i,j) = 0$ otherwise. Similarly one can define the adjacency matrix of a weighted as well as directed graphs. Let D denote the diagonal matrix whose i^{th} diagonal entry is the degree of the vertex i. The Laplacian matrix of an undirected graph is $L := D - A$. Suppose we have an undirected graph G together with a fixed orientation of its edges, i.e., every undirected edge $\{i,j\}$ of G is oriented either (i,j) or (j,i). The oriented matrix M of G with respect to a fixed orientation is a skew symmetric matrix, i.e., $M(i,j) = -M(j,i)$ such that $M(i,j) = 0$ if $\{i,j\}$ is not an edge in G; otherwise it is equal to 1 if (i,j) is the orientation of the edge; otherwise it is equal to -1. The matrices A, L, and M hold some crucial information about graphs. In the context of planar graphs the algebraic properties of these matrices can be exploited to obtain space efficient algorithms.

Counting Perfect Matchings in Planar Graphs

An orientation of an undirected graph is called *Pfaffian* if the determinant of the skew symmetric matrix M with respect to the orientation is exactly equal to the square of the number of perfect matchings in the graph. Thus immediately suggests the following approach for counting perfect matchings in a graph that has a *Pfaffian Orientation*.

- obtain a *Pfaffian Orientation* via an efficient algorithm
- compute the determinant of the skew symmetric matrix M associated to the *Pfaffian Orientation*; the square root of this determinant is the # perfect matchings.

In this context, planar graphs have been extremely lucky. Around 1965, Kastelyn [21] discovered that the planar graphs are endowed with an amazing property:

Theorem 5 (Kastelyn)**.** *Every planar graph has a* Pfaffian Orientation.

In fact, Kastelyn discovered a simple and elegant criteria for a planar graph to have a Pfaffian Orientation. This criteria has lead to efficient algorithms for obtaining a Pfaffian Orientation. We need a definition for further explanation. We call a cycle *oddly oriented* if we traverse the cycle in the clockwise direction then we encounter odd number of edges oriented in the direction of traversal. Kastelyn showed the following:

- an orientation of a planar graph is *Pfaffian* if every simple cycle is oddly oriented.
- furthermore: there is an oddly oriented simple cycle if and only if there is an oddly oriented *face* of the planar graph, i.e., facial cycle

Thus to construct a Pfaffian orientation it suffices to give the orientation to the edges so that every face is oddly oriented. This *local* criteria for constructing the *Pfaffian Orientation* turns out to be useful in the context of space complexity. In fact, Mahajan and Vinay [24] observe that using *parity tree evaluation* techniques one can obtain a Log-space algorithm for computing such an orientation! A consequence of having a Log-space computable *Pfaffian Orientation* is that one can compute the number of perfect matchings in planar graph in the class NC, and hence in poly-logarithmic space. To note the contrast, recall that the problem of counting perfect matchings in general graphs is known to be $\#P$-hard by the classic result of Valiant [30].

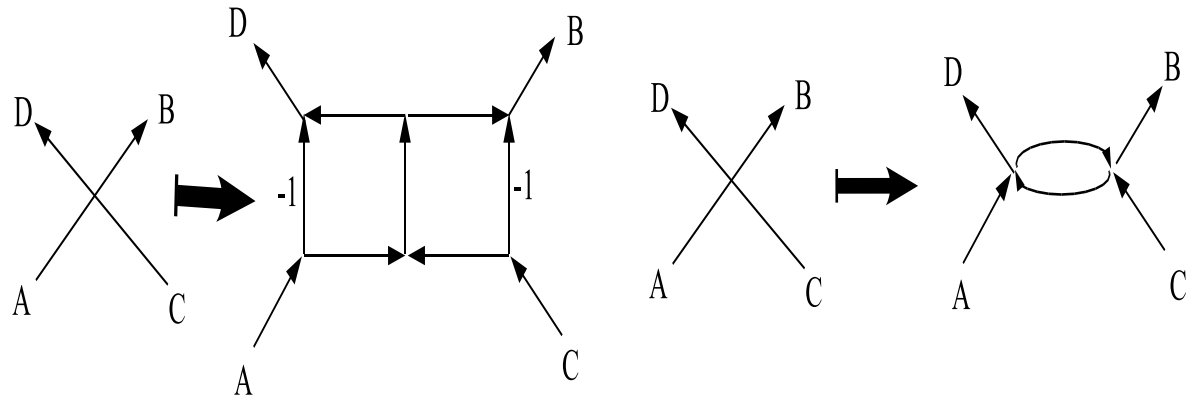
Counting Spanning Trees modulo 2^k in Planar Graphs

Another setting where planarity works like magic is that of modular counting of spanning trees. In particular, Braverman, Kulkarni, and Roy [6] show that for any fixed k, computing the number of spanning trees modulo 2^k in planar graphs can be done in Log-space! To show the contrast, we note that same problem in general graphs is known to be $\oplus$L-complete [6], i.e., as hard as computing the determinant of an integer matrix modulo 2.

This result uses two interesting connections.

- connection of certain type of knots to the co-rank of Laplacian matrix (mod 2)
- the Matrix-Tree Theorem that connects # spanning trees to the determinant of a minor of the Laplacian

First we give some explanation on the first connection: There are certain type of knots that are called *left-right walks* (cf. Godsil and Royle [20]) that one can associate to a planar graph. These knots can be constructed in Log-space. The number of these knots is exactly equal to the co-rank of the Laplacian matrix (modulo 2) of the planar graph. Counting the number of these knots can be done by a simple reduction to undirected connectivity problem, which is known to be in Log-space; thanks to the celebrated result of Reingold [26]. Now the magic is handed over to the Kirchoff's Matrix-Tree Theorem:



(a) Planarizing Gadget for Determinant (b) Planarizing Gadget for Permanent

Theorem 6 (Kirchoff). *Let L be the Laplacian matrix of graph G and let L' denote the matrix obtained from L by deleting the first row and the first column. Then: determinant of L' is exactly equal to the number of spanning trees in G.*

Thus the number of spanning trees is odd if and only if determinant of L' modulo 2 is odd. This happens if and only if the co-rank of L is at most 1. This immediately gives a Log-space algorithm for counting (modulo 2) the spanning trees in planar graphs. In fact, Braverman, Kulkarni, and Roy extend this result to mod 2^k setting for any fixed k using somewhat sophisticated techniques from Algebraic Topology.

7 Planarizing Gadgets

Every graph can be drawn on plane. One can represent the nodes of the graph by some points in the plane and one can represent the edges of the graph by curves joining the points. We say that two edges cross if the curves representing the two edges intersect in a point other than the end points of the edges. A graph is planar if and only if one can draw the graph on plane so that no two edges cross. Hence the only (and the big) difference between planar graphs and arbitrary graphs is that planar graph has no crossings while arbitrary graph might have many crossings.

A planarizing gadgets is loosely speaking a local replacement for crossings to obtain a planar graph. The replacement is done in order to maintain certain properties. Such gadgets can be used in two ways:

- to obtain an efficient algorithm for a problem in general graphs
- to obtain a hardness result for problems planar graphs.

Below we illustrate the two with some examples.

Upper Bounds using Planarizing Gadgets

A perfect example of the power of planarizing gadgets for obtaining upper bounds is given by the framework of so called *Holographic Algorithms*. This framework was introduced by

Valiant [31] and was further developed by Cai et. al. [34]. It exploits the fact that perfect matchings in planar graphs can be computed efficiently, to obtain non-trivial polynomial time algorithms for several seemingly hard problems. The key to this framework is the construction of so called *Match-Gates*, which are the planarizing gadgets satisfying certain properties. It is a beautiful piece of art to construct such Match-gates, see for instance a survey by Cai [34].

The framework works as follows:

- model a counting problem in general graphs in terms of *Match-Gate* circuits
- design appropriate *Match-Gates* and replace the crossings by these *Match-Gates*

In fact the only computationally expensive step in these algorithms is the computation of determinant. This step can be performed in NC, and hence in poly-logarithmic space. So all these algorithms can be considered space efficient.

Hardness Results using Planarizing Gadgets

One of the main uses of planarizing gadgets is to obtain hardness results. The basic strategy for proving hardness results for problems in planar graphs is as follows:

- start with a hard instance of the problem in general graphs
- replace each crossing by a suitable planar graph such that the solution is *preserved* under this replacement

Classically several hardness results are known for planar graphs via such *gadgets*: for instance Hamiltonian Cycle in planar graph is NP-hard [18]. Oftentimes the gadgets are simple once they are discovered but could be challenging to construct. In the context of space bounded computation (and this article) we would like to mention the following:

Datta, Kulkarni, Limaye, and Mahajan [10] study the Determinant and Permanent when restricted to adjacency matrices of planar graphs. They show that planar restrictions of Determinant and Permanent remain as hard as general graph.

Using an interesting use of the gadgets developed by Datta, Kulkarni, Limaye, and Mahajan together with some additional gadgets, Kulkarni [22] shows that the extension of Datta, Kulkarni, and Roy's [11] method of isolation from bipartite to non-bipartite planar graphs is hard; in the sense that such an extension would imply several strong results such as Bipartite Matching is in NC, $\mathsf{NL} \subseteq \oplus\mathsf{L}$, and $\mathsf{NP} \subseteq \oplus\mathsf{P}$.

Using the result of Kulkarni [22], recently Datta and Kulkarni [9] show that Max-Cut in planar graphs is NL hard.

Recently there have been some results on proving that certain type of planarizing gadgets do not exist (see for instance [19]).

8 Conclusion and Open Ends

We have shown several examples where studying the space complexity of problems in planar graphs could be fruitful. Several useful properties of planar graphs naturally lead to space efficient algorithms. We list some open questions that could be further explored:

Space Complexity of Recognizing Bounded Genus Graphs

It is known that one can recognize planar graph and in fact construct a combinatorial embedding of it in Log-space [3]. It is still open whether or not one can generalize it for constant-genus graphs. The best known bound is NC [16].

Space Complexity of Reachability in Planar Graphs

Is reachability in directed planar graphs in Log-space ? A related question is whether or not the distance computation in undirected planar graphs is in Log-space?

Isolation in Non-bipartite Planar Graphs

Can one extend the method of non-vanishing circulation of Datta, Kulkarni, and Roy [11] to non-bipartite planar graphs ? Such an extension would have non-trivial consequences such as $\mathsf{NL} \subseteq \oplus\mathsf{L}$ [22].

Space Complexity of Max-Cut in Planar Graphs

We have observed that Max-Cut in planar graphs admits $UL \cap co - UL$ approximation scheme. Furthermore: we have shown that Max-Cut in planar graphs is NL hard. A natural question is whether or not there is a Log-space approximation scheme for Max-Cut in planar graphs.

Space Complexity of Matching in Planar Graphs

Perfect Matching in bipartite planar graphs is known to have space efficient algorithms [11] but non-bipartite case remains open for long time. Kulkarni [22] shows that Minimum Weight Perfect Matching in planar graphs is NL hard even when the weights are 0 or 1. Can one show NL hardness for Perfect Matching or Maximum Matching in planar graphs? Does Perfect Matching in planar graphs have poly-log space algorithm?

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.

[2] Manindra Agrawal. Rings and integer lattices in computer science. In *Proceedings of Barbados Workshop on Computational Complexity*, 2007.

[3] Eric Allender and Meena Mahajan. The complexity of planarity testing. *Information and Computation*, 189:117–134, 2004.

[4] Eric Allender, Klaus Reinhardt, and Shiyu Zhou. Isolation, matching, and counting: Uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[5] Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

[6] Mark Braverman, Raghav Kulkarni, and Sambuddha Roy. Space-efficient counting in graphs on surfaces. *Computational Complexity*, 18(4):601–649, 2009.

[7] Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Trans. Comput. Theory*, 1(1):1–17, 2009.

[8] Samir Datta, Arjun Gopalan, Raghav Kulkarni, and Raghunath Tewari. Improved bounds for bipartite matching on surfaces. In *STACS*, pages 254–265, 2012.

[9] Samir Datta and Raghav Kulkarni. Space complexity of optimization problems in planar graphs. *Manuscript*, 2013.

[10] Samir Datta, Raghav Kulkarni, Nutan Limaye, and Meena Mahajan. Planarity, Determinants, Permanents, and (Unique) Matchings. In *CSR*, pages 115–126, 2007.

[11] Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically Isolating a Perfect Matching in Bipartite Planar Graphs. In *STACS*, pages 229–240, 2008.

[12] Samir Datta, Raghav Kulkarni, and Sambuddha Roy. Deterministically isolating a perfect matching in bipartite planar graphs. *Theory of Computing Systems*, 47:737–757, 2010. 10.1007/s00224-009-9204-8.

[13] Samir Datta, Nutan Limaye, and Prajakta Nimbhorkar. 3-connected Planar Graph Isomorphism is in Log-space. In *FSTTCS*, pages 155–162, 2008.

[14] Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar Graph Isomorphism is in Log-Space. In *IEEE Conference on Computational Complexity*, pages 203–214, 2009.

[15] Samir Datta and Gautam Prakriya. Planarity testing revisited. In *TAMC*, pages 540–551, 2011.

[16] Hristo Djidjev and John H. Reif. An efficient algorithm for the genus problem with explicit construction of forbidden subgraphs. In *STOC*, pages 337–347, 1991.

[17] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. In *FOCS*, pages 143–152, 2010.

[18] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[19] Rohit Gurjar, Arpita Korwar, Jochen Messner, Simon Straub, and Thomas Thierauf. Planarizing gadgets for perfect matching do not exist. In *MFCS*, pages 478–490, 2012.

[20] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. Springer Verlag, New York 1st ed, 2001.

[21] P W Kastelyn. Graph theory and crystal physics. In F Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

[22] Raghav Kulkarni. On the power of isolation in planar graphs. *TOCT*, 3(1):2, 2011.

[23] Gary L. Miller and Joseph Naor. Flow in planar graphs with multiple sources and sinks. *SIAM J. Comput.*, 24(5):1002–1017, 1995.

[24] Meena Mahajan and Kasturi R. Varadarajan. A new nc-algorithm for finding a perfect matching in bipartite planar and small genus graphs (extended abstract). In *STOC*, pages 351–357, 2000.

[25] Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM Journal of Computing*, 29:1118–1131, 2000. An earlier version appeared in FOCS 1997, pp. 244–253.

[26] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):1–24, 2008.

[27] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[28] Raghunath Tewari and N. V. Vinodchandran. Green's theorem and isolation in planar graphs. Technical Report TR10-151, Electronic Colloquium on Computational Complexity, 2010.

[29] Thomas Thierauf and Fabin Wagner. Reachability in $K_{3,3}$-free graphs and K_5-free graphs is in unambiguous log-space. In *17th International Conference on Foundations of Computation Theory (FCT)*, Lecture Notes in Computer Science 5699, pages 323–334. Springer-Verlag, 2009.

[30] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[31] Leslie G. Valiant. Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008.

[32] Heribert Vollmer. *Introduction to circuit complexity - a uniform approach.* Texts in theoretical computer science. Springer, 1999.

[33] Hassler Whitney. A set of topological invariants for graphs. *American Journal of Mathematics*, 55:235–321, 1933.

[34] Jin yi Cai. Holographic algorithms: guest column. *SIGACT News*, 39(2):51–81, 2008.