I/O-Efficient Planar Separators and Applications

Norbert Zeh

May 30, 2001

Abstract

We present a new algorithm to compute a subset S of vertices of a planar graph G whose removal partitions G into $O(N/h)$ subgraphs of size $O(h)$ and with boundary size $O(\sqrt{h})$ each. The size of S is $O(N/\sqrt{h})$. Computing S takes $O(\text{sort}(N))$ I/Os and linear space, provided that $M \geq 56h\log^2 B$. Together with recent reducibility results, this leads to $O(\text{sort}(N))$ I/O algorithms for breadth-first search (BFS), depth-first search (DFS), and single source shortest paths (SSSP) on undirected embedded planar graphs. Our separator algorithm does not need a BFS tree or an embedding of G to be given as part of the input. Instead we argue that "local embeddings" of subgraphs of G are enough.

1 Introduction

I/O-efficient graph algorithms have received considerable attention lately because massive graphs arise naturally in many applications. Recent web crawls, for example, produce graphs of on the order of 200 million nodes and 2 billion edges. Recent work in web modeling uses depth-first search, breadth-first search, shortest paths, and connected components as primitive operations for investigating the structure of the web [8]. Massive graphs are also often manipulated in Geographic Information Systems (GIS), where many fundamental problems can be formulated as basic graph problems. The graphs arising in GIS applications are often planar. Yet another example of massive graphs is AT&T's 20TB phone call graph [9]. When working with such large data sets, the transfer of data between internal and external memory, and not the internal memory computation, is often the bottleneck. Thus, I/O-efficient algorithms can lead to considerable run-time improvements.

Breadth-first search (BFS) and depth-first search (DFS) are the two most fundamental graph searching strategies. They are extensively used in internal memory algorithms, as they are easy to perform in linear time; yet they provide valuable information about the structure of the given graph. Unfortunately, no I/O-efficient algorithms for BFS and DFS in arbitrary sparse graphs are known, while existing algorithms perform reasonably well on dense graphs.

In this paper, we develop a new algorithm for computing small separators of planar graphs. Together with recent results on single-source shortest paths (SSSP) and DFS, our algorithm leads to I/O-efficient algorithms for SSSP, BFS, and DFS on undirected embedded planar graphs.

1.1 Model of Computation

The algorithms in this paper are designed and analyzed in the Parallel Disk Model (PDM) [27]. In this model, D identical disks of unlimited size are attached to a machine with an internal memory capable of holding M data items. These disks constitute the external memory of the machine. Initially, all data is stored on disk. Each disk is partitioned into blocks of B data items each. An I/O-operation is the transfer of up to D blocks, at most one per disk, to or from internal memory from or to external memory. The complexity of an algorithm in the PDM is the number of I/O-operations it performs.

Sorting, permuting, and scanning an array of N consecutive data items are primitive operations often used in external memory algorithms. Their I/O-complexities are $\text{sort}(N) = \Theta((N/DB)\log_{M/B}(N/B))$, $\text{perm}(N) = \Theta(\min(N, \text{sort}(N)))$, and $\text{scan}(N) = O(N/DB)$, respectively [27].

1.2 Previous Results

Many authors have studied I/O-efficient graph algorithms [1, 2, 5, 6, 10, 18, 19, 22, 23, 25, 26]. We only discuss results on BFS, DFS, SSSP, and graph separators here. The best SSSP algorithm for arbitrary undirected graphs takes $O(|V| + (|E|/B)\log_2|E|)$ I/Os [19]. The best BFS algorithm for arbitrary undirected graphs takes $O(|V| + \text{sort}(|E|))$ I/Os [26]. Recently a BFS algorithm for graphs of bounded degree has been presented in [25]. If d is the maximum vertex degree in the graph, the algorithm takes $O(|V|/(\gamma\log_d B) + \text{sort}(B^\gamma|V|))$ I/Os using $O(|V|/B^{1-\gamma})$ blocks of external memory, for $0 < \gamma \leq \frac{1}{2}$.

In [18], an $O(\text{sort}(N))$ I/O algorithm for computing a 2/3-separator of size $O(\sqrt{N})$ for an embedded planar graph G is given, provided that a BFS-tree of G is part of the input. In [5], this idea has been extended to obtain an $O(\text{sort}(N))$ I/O algorithm to compute a small ε-separator of an embedded planar graph, provided that a BFS-tree of the graph is given. Using the computed separator, the SSSP problem can then be solved in $O(\text{sort}(N))$ I/Os for the given graph [5]. In a recent paper [6], two DFS algorithms for embedded planar graphs are given. The first one takes $O(\text{sort}(N)\log N)$ I/Os. The second one takes $O(I(N))$ I/Os, where $I(N)$ is the number of I/Os required to compute a BFS-tree of an embedded planar graph. This seems to suggest that BFS is the core problem to be solved on planar graphs, in order to obtain efficient algorithms for all other problems discussed here.

For undirected outerplanar graphs, $O(\text{sort}(N))$ I/O algorithms for BFS, DFS, and finding 2/3-separators of size 2 are presented in [22]. Together with the algorithm of [5], this also gives an $O(\text{sort}(N))$ I/O SSSP algorithm for undirected outerplanar graphs. In [23] it is shown how to solve the SSSP problem in $O(\text{sort}(N))$ I/Os on graphs of bounded treewidth. It is shown in [22] that BFS, DFS, and SSSP require at least $\Omega(\text{perm}(N))$ I/Os, even on outerplanar graphs.

In internal memory, the problem of computing graph separators is well studied. We mention the most relevant results here. In their classic paper [21], Lipton and Tarjan show that a $\frac{2}{3}$-separator of size $O(\sqrt{N})$ can be computed in linear time for a given embedded planar graph G of size N. The required embedding can be computed in linear time [17, 7, 15, 20, 11, 24]. In [16], the algorithm of [21] is applied recursively to compute in $O(N\log N)$ time a set S of $O(N/h)$ vertices whose removal partitions G into $O(N/h)$ (possibly disconnected) subgraphs of size $O(h)$, each of which is adjacent to at most $O(\sqrt{h})$ vertices in S. In [3] it is shown how to compute a set S of $O(\sqrt{(g+1/\varepsilon)N})$ vertices whose removal partitions an embedded graph G of genus g into subgraphs of size at most εN. Other results include results on edge separators [12], separators for graphs with multiple vertex weights [14], and separators of low cost if a cost function of the vertices of G is given [13].

1.3 Our Results

In this paper we exploit the fact that BFS, DFS, and SSSP in embedded planar graphs can all be reduced to computing small separators. Using the result of [5], SSSP can be solved in $O(\text{sort}(N))$ I/Os, given a small separator of G. Giving all edges in the graph unit weight, this immediately gives a BFS-algorithm for the given graph. Using the reduction of [6], DFS can be reduced to BFS in $O(\text{sort}(N))$ I/Os. Thus, if we can compute a small separator of an embedded planar graph G in $O(\text{sort}(N))$ I/Os, we can solve all these problems in $O(\text{sort}(N))$ I/Os on embedded planar graphs.

We show how to find a planar separator S of size $O(N/\sqrt{h})$ whose removal partitions the given graph into $O(N/h)$ subgraphs of size at most h so that each subgraph is adjacent to at most $\sqrt{h}$ separator vertices. Our algorithm takes $O(\text{sort}(N))$ I/Os, provided that $M \geq 56h\log^2 B$. Our algorithm does not make any

assumptions about the input, except that the given graph G is planar. In particular, we do not require an embedding or a BFS-tree of G. This is the main improvement over previous algorithms.

1.4 Preliminaries

An *undirected graph* $G = (V,E)$ is an ordered pair of two sets V and E. The elements of V are called the *vertices* of G; the elements of E are the *edges* of G and are unordered pairs $\{v,w\}$, $v,w \in V$. For an edge $\{v,w\} \in E$, vertices v and w are the *endpoints* of edge $\{v,w\}$. Also, v and w are said to be *adjacent*. Edge $\{v,w\}$ is *incident* to vertices v and w. The notion of adjacency can be extended to vertex sets and subgraphs of G. Two vertex sets $V_1 \subseteq V$ and $V_2 \subseteq V$, $V_1 \cap V_2 = \emptyset$, are adjacent if there are two adjacent vertices $v \in V_1$ and $w \in V_2$. Two subgraphs $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ of G are adjacent if V_1 and V_2 are adjacent. A graph G is said to be *planar* if it can be drawn in the plane so that the edges of G do not intersect, except at their endpoints. We call such a drawing of G a *(planar) embedding* and denote it by $\hat{G}$. Given such an embedding $\hat{G}$, we call the connected regions of $\mathbb{R}^2 \setminus \hat{G}$ the *faces* of $\hat{G}$. Let F denote the set of faces of $\hat{G}$. Then Euler's formula says that $|V| + |F| - |E| = 2$. In particular, this implies that $|E| \leq 3|V| - 6$, for every planar graph G. We define the *size* $|G|$ of a planar graph G as the number $|V|$ of vertices in G. As $|E| = O(|V|)$, $|V| + |E| = O(|G|)$.

Given a set $S \subseteq V$ of vertices of G and a subgraph $H \subseteq G - S$, ∂H is the set of vertices in S adjacent to H. We call ∂H the *boundary* of H. The following two results will be applied in our separator algorithm.

Theorem 1 [3] *Given a planar graph $G = (V,E)$ and an integer $h > 0$, it takes $O(N)$ time to compute a set $S \subseteq V$ of $O(N/\sqrt{h})$ vertices so that no connected component of $G - S$ has size exceeding h.*

Theorem 2 [16] *Given a planar graph $G = (V,E)$ and a set $S \subseteq V$ of vertices whose removal partitions G into $O(N/h)$ subgraphs $H_1, \ldots, H_q$ such that $|H_i| \leq h$ and $\sum_{i=1}^{q} |\partial H_i| = O(N/\sqrt{h})$, it takes $O(N \log N)$ time to compute a set S' of $O(N/\sqrt{h})$ vertices, $S \subseteq S' \subseteq V$, whose removal partitions G into $O(N/h)$ subgraphs $H'_1, \ldots, H'_{q'}$ such that $|H'_i| \leq h$ and $|\partial H'_i| \leq c\sqrt{h}$, for $1 \leq i \leq q$ and some constant $c \geq 0$.*

A graph $G = (V,E)$ is *bipartite* if the vertex set V can be partitioned into two sets V_1 and V_2 such that $v \in V_1$ and $w \in V_2$, for every edge $\{v,w\} \in E$. In this case we write $G = (V_1,V_2,E)$. We will need the following two technical results.

Lemma 1 *Let $G = (V_1,V_2,E)$ be a bipartite planar graph such that the vertices in V_2 have degree at least three each. Then $|V_2| \leq 2|V_1|$.*

Proof. Consider an embedding $\hat{G}$ of G. As G is bipartite, every face of $\hat{G}$ has size at least 4. Thus, $|F| \leq |E|/2$. By Euler's formula $|V| + |F| - |E| = 2$. That is,

$$\begin{aligned} 2 &= |V| + |F| - |E| \\ &\leq |V| - |E|/2 \\ |E| &\leq 2|V|. \end{aligned}$$

On the other hand, $|E| \geq 3|V_2|$, so that

$$\begin{aligned} 3|V_2| &\leq 2|V| \\ &= 2(|V_1| + |V_2|) \\ |V_2| &\leq 2|V_1|. \end{aligned}$$

□

Corollary 1 *Let $G = (V_1, V_2, E)$ be a bipartite planar graph. Let the vertices in V_2 be partitioned into equivalence classes $C_1, \ldots, C_q$, where two vertices v and w are equivalent if they are adjacent to the same set of vertices in V_1. Then $q \leq 6|V_1|$.*

Proof. Every vertex $v \in V_1$ defines one class C_v such that the vertices in C_v are adjacent only to v. Thus, there are at most $|V_1|$ such classes C_v. Now consider a pair $\{v,w\}$ of vertices in V_1. Let $C_{\{v,w\}}$ be the set of vertices in V_2 adjacent only to v and w. Then we choose one representative $r_{\{v,w\}} \in C_{\{v,w\}}$ for each such class of vertices. Let H_1 be the bipartite subgraph of G induced by all edges incident to such representatives $r_{\{v,w\}}$. As H_1 is a subgraph of G, H_1 is planar. We now remove representatives $r_{\{v,w\}}$ from H_1 and replace edges $\{v, r_{\{v,w\}}\}$ and $\{r_{\{v,w\}}, w\}$ by a single edge $\{v,w\}$. As every representative $r_{\{v,w\}}$ has degree 2, the resulting graph H_2 is a planar graph whose vertex set is a subset of V_1, and whose edges are in one-to-one correspondence to the classes $C_{\{v,w\}}$. Thus, by Euler's formula, there can be at most $3|V_1|$ such classes $C_{\{v,w\}}$. Finally, let H_3 be the subgraph of G induced by all edges incident to vertices of degree at least 3 in V_2. Graph H_3 is a bipartite planar graph $H_3 = (V_1', V_2', E')$ with $V_1' \subseteq V_1$, $V_2' \subseteq V_2$, and $E' \subseteq E$. All vertices in V_2' have degree at least 3, so that $|V_2'| \leq 2|V_1'| \leq 2|V_1|$, by Lemma 1. Each vertex in V_2' gives rise to at most one category C_i, so that there are at most $6|V_1|$ categories C_i in total. □

Given a graph $G = (V,E)$ and an edge $\{v,w\} \in E$, the *contraction* of edge $\{v,w\}$ is the operation of removing w from G and making all edges $\{u,w\} \in G$, $u \neq v$, incident to v. If the vertices of G have weights, then the weight of v is increased by the weight of w. It is known that contracting edges of a planar graph G preserves the planarity of G.

A *matching* of a graph $G = (V,E)$ is a subset $\mathcal{M} \subseteq E$ of edges so that no two edges in $\mathcal{M}$ share an endpoint. The matching $\mathcal{M}$ is *maximal* if every edge in $E \setminus \mathcal{M}$ shares an endpoint with an edge in $\mathcal{M}$.

2 Separators for Planar Graphs

In this section we present a new separator algorithm for planar graphs. In particular, we show how to compute in $O(\mathrm{sort}(N))$ I/Os and linear space a subset S of $O(N/\sqrt{h})$ vertices of a given planar graph G whose removal partitions G into $O(N/h)$ subgraphs $H_1, \ldots, H_q$ such that $|H_i| \leq h$ and $|\partial H_i| \leq \sqrt{h}$, for $1 \leq i \leq q$, provided that $M \geq 56h\log^2 B$.

The main idea of our algorithm is to construct a hierarchy of $\log_2 B$ graphs $G_0, \ldots, G_r$ such that $G = G_0$, $|G_{i+1}| \leq \frac{1}{2}|G_i|$, and every vertex in G_{i+1} represents a small subgraph of G_i and thus a subgraph of G. Given this hierarchy, we start by computing a small separator S_r partitioning G_r into relatively coarse subgraphs such that the number of vertices in G corresponding to the vertices in S_r is small. Then we iteratively refine graph G_r to graphs $G_{r-1}, G_{r-2}, \ldots, G_0$ as well as the partitions of graphs G_i until we obtain a separator of size $O(N/\sqrt{h})$ partitioning G into subgraphs of size at most $h\log^2 B$. Given such a partition, we load each subgraph into internal memory and refine it so that no subgraph has size exceeding h or boundary size exceeding $\sqrt{h}$. This can be done using Theorems 1 and 2 and introduces at most $O(N/\sqrt{h})$ additional separator vertices. Summing up, we construct a separator of size $O(N/\sqrt{h})$ partitioning G into subgraphs of size at most h and with boundary size at most $\sqrt{h}$. If we can implement every recursive step in $O(\mathrm{sort}(|G_i|))$ I/Os, we obtain an algorithm that takes $O(\mathrm{sort}(N))$ I/Os to compute the desired separator.

Our algorithm proceeds in two phases. The first phase is only concerned with generating subgraphs of size at most $h\log^2 B$ while keeping the total separator size small. We are not concerned about the boundary size of each subgraph at this point. The second phase partitions each subgraph further, so that every subgraph has size at most h and boundary size at most $\sqrt{h}$. We now describe these phases in detail.

2.1 The Graph Hierarchy

The first step is to compute the sequence of graphs $G_0, \ldots, G_r$. We first show how to construct graph G_{i+1} from graph G_i and then prove a number of useful properties of graphs $G_0, \ldots, G_r$.

Given a graph G_i, every vertex $v \in G_i$ has a *weight* $\omega(v)$, which is the number of vertices in G represented by v. That is, $\omega(v) = 1$, for each vertex $v \in G_0$, as $G_0 = G$. For every vertex $v \in G_i$, its *size* $\sigma(v)$ is the number of vertices in G_{i-1} represented by v. We define a series of weight thresholds $\rho_i = 2^{i+1}$, for $0 \leq i \leq r$. Given G_i, we initially define a graph $G'_{i+1} = G_i$ with $\sigma(v) = 1$, for all $v \in G'_{i+1}$. The weight of a vertex in G'_{i+1} is the same as its weight in G_i. We inspect the edges of G'_{i+1} in an arbitrary order. As long as there is an edge $\{v, w\} \in G'_{i+1}$ such that $\omega(v) + \omega(w) \leq \rho_{i+1}$ and $\sigma(v) + \sigma(w) \leq 56$, we contract edge $\{v, w\}$ and repeat. Let G''_{i+1} be the resulting graph which does not allow any further edge contractions.

We call a vertex v in G''_{i+1} *heavy* if $\omega(v) \geq \rho_{i+1}/2$ or $\sigma(v) \geq 28$, and *light* otherwise. Observe that for every edge $\{v, w\} \in G''_{i+1}$, either $\omega(v) + \omega(w) > \rho_{i+1}$ or $\sigma(v) + \sigma(w) > 56$, so that at least one of v and w is heavy. Thus, no two light vertices are adjacent. We partition the light vertices of degree at most 2 in G''_{i+1} into classes $C_1, \ldots, C_q$ such that the vertices in each class have the same set of (heavy) neighbors. We partition each class C_j into subclasses $C_{j,1}, \ldots, C_{j,k_j}$, so that the total weight of each subclass $C_{j,l}$, $1 \leq l \leq k_j$, is at most ρ_i and its total size is at most 56. For $1 \leq l < k_j$, either the total weight of $C_{j,l}$ is at least $\rho_i/2$ or its total size is at least 28. The last class C_{j,k_j} may have weight less than $\rho_{i+1}/2$ and size less than 28.

Graph G_{i+1} is defined as follows: The vertex set of G_{i+1} consists of all heavy vertices of G''_{i+1}, all light vertices of degree at least 3, as well as one vertex $v_{j,l}$, for each class $C_{j,l}$ of light vertices of degree at most 2. For every heavy vertex $v \in G''_{i+1}$, the weight of v in G_{i+1} is the same as in G''_{i+1}. The same is true for all light vertices of degree at least 3. For every vertex $v_{j,l}$, we define $\omega(v_{j,l}) = \sum_{v \in C_{j,l}} \omega(v)$. There is an edge between two vertices v and w, where v and w are either heavy or light and of degree at least 3, if there is an edge between v and w in G''_{i+1}. There is an edge $\{v, v_{j,l}\} \in G_{i+1}$, where v is a heavy vertex and $v_{j,l}$ corresponds to class $C_{j,l}$, if the vertices in $C_{j,l}$ are adjacent to v in G''_{i+1}. The following lemma provides important properties of graphs $G_0, \ldots, G_r$, which are crucial to guarantee an upper bound on the size of the separator we compute in the next section.

Lemma 2 *Let G be planar, and let $G = G_0, \ldots, G_r$ be the graphs as defined above. Every graph G_i, $0 \leq i \leq r$, has the following properties:*

(i) Graph G_i is planar,

(ii) $\omega(v) \leq \rho_i$, for all vertices $v \in G_i$,

(iii) Every vertex $v \in G_i$ corresponds to at most 56 vertices in G_{i-1}, for $i > 0$, and

(iv) $|G_i| \leq 28N/\rho_i$.

Proof. To prove Property (i), we inductively show that each graph G_i, $0 \leq i \leq r$, has a planar embedding $\hat{G}_i$. For $i = 0$, $G_0 = G$, so that an embedding $\hat{G}_0$ of G_0 exists by the planarity of G. So assume that we are given an embedding $\hat{G}_{i-1}$ of graph G_{i-1}. We show how to obtain a planar embedding $\hat{G}_i$ of G_i. First we construct a planar embedding of G''_i. Such an embedding is easily obtained from $\hat{G}_{i-1}$, as G''_i is constructed from G_{i-1} using a series of edge contractions. The adjacencies between heavy vertices and light vertices of degree at least 3 do not change from G''_i to G_i. For a class $C_{i,l}$, vertex $v_{i,l}$ is adjacent to the same heavy vertices in G_i as all members of $C_{i,l}$ in G''_i. Thus, we rename an arbitrary member of $C_{i,l}$ to $v_{i,l}$ and remove all other members of $C_{i,l}$ from G''_i. The result is the desired embedding $\hat{G}_i$.

Property (ii) is easily shown by induction. In particular, $\omega(v) = 1 \leq \rho_0$, for all vertices $v \in G_0$. Given that all vertices in G_i have weight at most ρ_i, the weight of a vertex in G_{i+1} can exceed ρ_{i+1} only by merging

two or more vertices into a single vertex. But we merge vertices only if their total weight does not exceed ρ_{i+1}. Property (iii) is explicitly guaranteed by our construction.

It remains to show Property (iv). Let h_i be the number of heavy vertices in G_i. It follows from the above construction and Corollary 1 that $|G_i| \leq 7h_i$. Thus, Property (iv) follows if we can show that $h_i \leq 4N/\rho_i$.

We prove this claim by induction. We partition the heavy vertices into two categories. Heavy vertices of type I are heavy because their weight is at least $\rho_i/2$. Type-II vertices are heavy because their size is at least 28. In general, graph G_i contains at most $2N/\rho_i$ type-I vertices and at most $|G_{i-1}|/28$ type-II vertices. That is $h_i \leq \frac{2N}{\rho_i} + \frac{|G_{i-1}|}{28}$.

For $i = 1$, we obtain $h_1 \leq \frac{2N}{\rho_1} + \frac{N}{28} < \frac{4N}{\rho_1}$ because $\rho_1 = 4$. For $i > 1$, we obtain

$$h_i \leq \frac{2N}{\rho_i} + \frac{|G_{i-1}|}{28} \tag{1}$$

$$\leq \frac{2N}{\rho_i} + \frac{h_{i-1}}{4} \tag{2}$$

$$\leq \frac{2N}{\rho_i} + \frac{N}{\rho_{i-1}} \tag{3}$$

$$= \frac{4N}{\rho_i}. \tag{4}$$

Line (2) follows from Line (1) because $|G_{i-1}| \leq 7h_{i-1}$, as shown above. Line (3) follows from Line (2) by the induction hypothesis. Line (4) follows from Line (3) using the fact that $\rho_i = 2\rho_{i-1}$. □

Next we show how to compute graphs $G_0, \ldots, G_r$ I/O-efficiently. Graph G_0 is already given. So assume that we have computed graphs $G_0, \ldots, G_{i-1}$. We compute graph G_i as follows. Initially let $G'_i = G_{i-1}$. An edge $\{v,w\}$ of G'_i is called *contractible* if $\omega(v) + \omega(w) \leq \rho_i$ and $\sigma(v) + \sigma(w) \leq 56$. The contractible subgraph H_0 of G'_i is the graph induced by all contractible edges in G'_i. H_0 can easily be extracted from G'_i in $O(\text{sort}(|G_{i-1}|))$ I/Os. We compute a maximal matching of H_0, which takes $O(\text{sort}(|H_0|))$ I/Os [23]. We contract all edges in the matching, leading to a graph H'_0. We call a vertex v of H'_0 *matched* if it is the result of contracting a matching edge in H_0. Otherwise, we call v *unmatched*. First observe that two unmatched vertices cannot be adjacent because otherwise the matching we have computed is not maximal. Our goal is to construct a graph H''_0 such that no unmatched vertex has an incident edge which is contractible.

In order to do that we compute the subgraph $\tilde{H}$ of H'_0 induced by all edges incident to unmatched vertices. Graph $\tilde{H}$ is bipartite. Let V_u be the set of unmatched vertices and V_m be the set of matched vertices in $\tilde{H}$. We number the vertices in V_u and V_m in their order of appearance. For every vertex $v \in V_m$, we store the set of vertices in V_u adjacent to v, sorted by increasing indices in the numbering. Let w be a vertex adjacent to v, $v_1 < v_2 < \cdots < v_k$ be the vertices adjacent to w, and $v = v_j$. Then we store the index of vertex v_{j+1} with w in the adjacency list of v. If v_{j+1} does not exist, we mark v as being the last vertex adjacent to w. Also, we mark v as being the first vertex adjacent to w if $v = v_1$. This representation of $\tilde{H}$ can easily be constructed in $O(\text{sort}(|H_0|)$ I/Os from the edge list of H'_0.

Now we use a priority queue Q to contract the contractible edges in $\tilde{H}$. We inspect the vertices in V_m in their order of appearance. For every vertex v, let $w_1 < w_1 < \cdots < w_l$ be the vertices adjacent to v. For every vertex w_j we perform the following action. If v is the first vertex adjacent to w_j or if the minimum entry in Q is entry (v, w_j), we test whether $\omega(v) + \omega(w_j) \leq \rho_i$ and $\sigma(v) + \sigma(w_j) \leq 56$. If this is the case, we contract edge $\{v, w_j\}$ and increase the weight of v by $\omega(w_j)$ and the size of v by $\sigma(w_j)$. Otherwise, there are two possibilities. If v is the last vertex adjacent to w_j, none of the edges incident to w_j is contractible. If v is not the last vertex, let u be the vertex stored with w_j in the adjacency list w_j. Then we add the pair (u, w_j) to Q. If v is not the first vertex adjacent to w_j and the first entry in Q is not (v, w_j), then vertex w_j has been contracted into another heavy vertex already, and we proceed to the next vertex adjacent to v.

It is obvious that this algorithm achieves the desired goal. In particular, every edge in $\tilde{H}$ is being inspected by the algorithm, and the algorithm contracts the edge unless the edge is not contractible or the unmatched endpoint of the edge has been contracted into another matched vertex, so that the edge now joins two matched vertices. Thus, none of the remaining edges that are incident to unmatched vertices are contractible. This procedure takes $O(\mathrm{sort}(|H_0|))$ I/Os [4], as $\tilde{H}$ is planar and its size is bounded from above by the size of H_0.

Let H_0'' be the graph obtained from H_0' by contracting the edges in $\tilde{H}$ using the above algorithm. We extract the contractible subgraph H_1 of H_0'' and repeat the whole process. We stop as soon as the contractible subgraph H_{j+1} of graph H_j'' is empty. From the above discussion it follows that each iteration takes $O(\mathrm{sort}(|H_j|))$ I/Os. The following lemma shows that we repeat this compression procedure at most $\lfloor 2\log 56 \rfloor = 11$ times and that the total I/O-complexity is $O(\mathrm{sort}(|G_i|))$, as the sizes of graphs $H_0,\ldots,H_s$ are geometrically decreasing.

Lemma 3 *For every vertex $v \in H_j$, $\sigma(v) \geq 2^j$.*

Proof. The proof is by induction. For $j = 0$, the claim holds because every vertex in H_0 has size $\sigma(v) = 1$. So assume that the claim holds for $j < k$. Then every vertex in H_{k-1} has size at least 2^{k-1}. As every matched vertex v in H_{k-1}'' is the result of contracting at least one edge in H_{k-1}, its size is at least $\sigma(v) = 2 \cdot 2^{k-1} = 2^k$. Also, no unmatched vertex in H_{k-1}'' has an incident edge which is contractible. Thus, only the matched vertices in H_{k-1}'' remain in H_k. □

It is easy to extract the final vertex and edge sets of G_i' obtained after finishing all contractions. Then it takes sorting and scanning to partition the light vertices of degree at most two in G_i' into classes $C_1,\ldots,C_q$ and partition these classes into subclasses $C_{j,l}$, as described above. Thus, G_i can be constructed in $O(\mathrm{sort}(|G_{i-1}|)$ I/Os from G_{i-1}. As $|G_0| = N$, and the sizes of subgraphs G_i are geometrically decreasing, we obtain the following lemma.

Lemma 4 *The sequence $G_0,\ldots,G_r$ of graphs, as described above, can be constructed in $O(\mathrm{sort}(N))$ I/Os using $O(N/B)$ blocks of external memory.*

2.2 The Separator Hierarchy

Having constructed graphs $G_0,\ldots,G_r$ we use them to construct a relatively coarse separator of G. In particular, we show how to construct a separator S of size $O(N/\sqrt{h})$ whose removal partitions G into connected subgraphs of size at most $h\log^2 B$.

We start by computing a partition of G_r into subgraphs of size at most $h\log^2 B$. To do this, we use an arbitrary linear-time algorithm to compute a planar embedding of G_r and apply Theorem 1 to compute the desired partition. As $|G_r| = O(N/B)$, this takes $O(N/B)$ I/Os. Let $S_r = S_r''$ be the computed separator, whose size is $|S_r| \leq c|G_r|/(\sqrt{h}\log B)$, for some constant c defined in [3]. Given a separator S_{j+1} for graph G_{j+1}, we construct a separator S_j for graph G_j as follows: First we construct the set S_j' of vertices represented by the vertices in S_{j+1}. S_j' is a separator of G_j whose removal partitions G_j into subgraphs of size at most $56h\log^2 B$. This is true because every subgraph H of $G_{j+1} - S_{j+1}$ has size at most $h\log^2 B$, and every vertex in G_{j+1} represents at most 56 vertices in G_j, by Lemma 2. We load every subgraph H of $G_j - S_j$ whose size exceeds $h\log^2 B$ into internal memory, compute an embedding of H, and partition it into subgraphs of size at most $h\log^2 B$, again applying Theorem 1. Let S_j'' be the separator obtained by partitioning all heavy subgraphs of $G_j - S_j'$ this way. $|S_j''| \leq c|G_j|/(\sqrt{h}\log B)$, for the same constant c as defined above. Now $S_j = S_j' \cup S_j''$. We continue this procedure until we obtain a separator S_0 of $G_0 = G$.

Lemma 5 *The separator S_0 of G computed by the above procedure has size $O(N/\sqrt{h})$. The connected components of $G - S_0$ have size at most $h \log^2 B$.*

Proof. It follows from the above construction and Lemma 2 that

$$\begin{aligned}
|S_0| &\leq \sum_{i=0}^{r} \rho_i |S_i''| \\
&\leq c \sum_{i=0}^{r} \rho_i \frac{|G_i|}{\sqrt{h}\log B} \\
&\leq c \sum_{i=0}^{r} \rho_i \frac{28N}{\rho_i \sqrt{h}\log B} \\
&= c \sum_{i=0}^{r} \frac{28N}{\sqrt{h}\log B} \\
&= \frac{28cN}{\sqrt{h}}.
\end{aligned}$$

The bound on the size of the connected components of $G - S_0$ is explicitly ensured by our construction. □

Given separator S_{i+1}, it takes $O(\text{sort}(|G_i|))$ I/Os to construct S_i' and compute the connected components of $G_i - S_i'$ [10]. Then it takes $O(\text{scan}(|G_i|))$ I/Os to load each connected component into internal memory and perform the above partition. As the sizes of graphs $G_0, \ldots, G_r$ are geometrically decreasing, the total I/O-complexity of the algorithm is $O(\text{sort}(N))$.

2.3 Computing the Final Separator

In order to finish the computation of our separator algorithm, we have to partition each connected component of $G - S_0$ into smaller subgraphs of size at most h and boundary size at most $\sqrt{h}$. We do this by applying Theorems 1 and 2.

We use Theorem 1 first to partition the subgraphs of $G - S_0$ into subgraphs of size at most h each. In particular, we compute the connected components of $G - S_0$ and load them into internal memory one by one. For each such component Q, we compute an embedding $\hat{Q}$ and apply Theorem 1 to partition Q into subgraphs of size at most h. Let S' be the total number of separator vertices introduced by partitioning the connected components of $G - S_0$ this way. By Theorem 1, $|S'| = O(N/\sqrt{h})$. Let $S_0' = S_0 \cup S'$. Then $|S_0'| = O(N/\sqrt{h})$.

The final step is now to apply Theorem 2 in order to compute a separator $S \supseteq S_0'$ whose removal partitions G into $O(N/h)$ subgraphs whose boundary sizes do not exceed $\sqrt{h}$. There are two obstacles preventing us from applying Theorem 2 immediately. Firstly, the total boundary size of the connected components of $G - S_0'$ may be $\omega(N/\sqrt{h})$, even though the number of separator vertices is $O(N/\sqrt{h})$. Secondly, the number of connected components of $G - S_0'$ may be as large as $\Omega(N)$. In order to apply Theorem 2, we need to obtain a partition of $G - S_0'$ into $O(N/h)$ subgraphs $\bar{H}_1, \ldots, \bar{H}_s$ such that $|\bar{H}_i| \leq h$, for $1 \leq i \leq s$, and $\sum_{i=1}^{s} |\partial \bar{H}_i| = O(N/\sqrt{h})$.

We satisfy these two requirements using ideas similar to the compression step in the construction of the graph hierarchy in Section 2.1, but separately. First we ensure that the total boundary size is $O(N/\sqrt{h})$. Once this is done, we merge small subgraphs, in order to reduce the number of graphs to $O(N/h)$. A more straightforward procedure may produce the correct result; but our analysis uses the fact that at any time, we can represent the adjacencies between subgraphs and separator vertices using a planar graph. This would not be true if for instance we exchanged the two steps (i.e., first reduced the number of regions and then tried to argue about the total boundary size).

In order to reduce the total boundary size, we build a planar graph $\tilde{G}$ containing all separator vertices in S'_0 as well as one *region vertex* per connected component of $G - S'_0$. $\tilde{G}$ contains the same edges between vertices in S'_0 as G. There is an edge between a separator vertex v and a region vertex w representing component Q if v is adjacent to Q in G. We now consider all region vertices of degree at most two in $\tilde{G}$. We partition these vertices into equivalence classes so that the vertices in the same equivalence class have the same set of neighbors. For every region vertex w corresponding to a connected component Q, let its weight $\omega(w)$ be defined as $\omega(w) = |Q|$. A region vertex is *heavy* if its weight is at least $h/2$. We now merge region vertices in each equivalence class C until at most one light vertex is left in each class, while maintaining the property that no vertex has weight exceeding h.

As a result, there are $O(N/h)$ heavy region vertices and $O(N/\sqrt{h})$ light region vertices in $\tilde{G}$, by Corollary 1. Every vertex v in $\tilde{G}$ now represents a (possibly disconnected) subgraph H_i of $G - S'_0$. The total boundary size $\sum_{i=1}^{q} |\partial H_i|$ of these subgraphs is equal to the number of edges in $\tilde{G}$. However, $\tilde{G}$ has $O(N/\sqrt{h})$ vertices and is planar. Thus, $\sum_{i=1}^{q} |\partial H_i| = O(N/\sqrt{h})$, as desired.

Given $\tilde{G}$, we use it to further merge subgraphs $H_1, \ldots, H_q$, in order to reduce the number of subgraphs to $O(N/h)$. The crucial observation is that merging subgraphs cannot increase the total boundary size. We now give every separator vertex $v \in S'_0$ weight $\omega(v) = 0$. Region vertices retain their weights from the previous computation.

We compress $\tilde{G}$ further by choosing an arbitrary incident edge $\{v, w\}$ for each separator vertex $v \in \tilde{G}$ and contracting it. This preserves the planarity of $\tilde{G}$ and reduces the vertex set of $\tilde{G}$ so that it contains only region vertices. We apply our edge-contraction algorithm of Section 2.1 in order to further compress $\tilde{G}$ so that no vertex in $\tilde{G}$ has weight exceeding h and there is no edge $\{v, w\} \in G'$ such that $\omega(v) + \omega(w) < h$. A vertex in $\tilde{G}$ is heavy if its weight is at least $h/2$, and light otherwise. We now partition the light vertices of degree at most two in $\tilde{G}$ into equivalence classes so that the vertices in a class C have the same set of neighbors. We keep merging vertices in the same class until no class C contains more than one light vertex. As shown in Section 2.1, this procedure produces a graph $\bar{G}$ with $O(N/h)$ vertices. Each such vertex v represents a subgraph of $G - S'_0$ whose size is equal to the weight of v. Thus, we obtain a partition of $G - S'_0$ into $O(N/h)$ subgraphs $\bar{H}_1, \ldots, \bar{H}_s$ of size at most h such that $\sum_{i=1}^{s} |\partial \bar{H}_i| = O(N/\sqrt{h})$.

We now load each subgraph $\bar{H}_i$ into internal memory, compute an embedding of $\bar{H}_i$, and partition it into subgraphs of size at most h and boundary size at most $\sqrt{h}$. If we had used the algorithm in [16] to compute S'_0, this would be straightforward, as in fact every subgraph H_i *including its boundary* would have size at most h. Our algorithm on the other hand does not guarantee a bound on the size of $H_i \cup \partial H_i$ better than $O(N/\sqrt{h})$.

In order to work around this problem, we represent each subgraph $\bar{H}_i$ and its boundary by another graph $\tilde{H}_i$, which is obtained by compressing $\partial \bar{H}_i$ to some new set $\partial \tilde{H}_i$ whose size is at most $6h$. We first define $\tilde{H}_i$ and prove that its size is at most $7h$. Then we show how to derive the desired separator of $\bar{H}_i$ from a separator of $\tilde{H}_i$.

In order to compute $\tilde{H}_i$ from $\bar{H}_i$, we consider the bipartite graph induced by edges $\{v, w\}$, $v \in \bar{H}_i, w \in \partial \bar{H}_i$. The compression technique we apply is the one we have applied already a number of times in our algorithm. In particular, we partition the separator vertices in $\partial \bar{H}_i$ into classes according to the sets of adjacent vertices in $\bar{H}_i$. Let the *degree* of such a class C be the number of vertices in $\bar{H}_i$ adjacent to the vertices in C. For each class C of degree at most two, we replace all vertices in C by a single vertex v whose weight we define to be $\omega(v) = |C|$. For all other vertices $v \in \partial \bar{H}_i$, $\omega(v) = 1$. For all vertices in $\bar{H}_i$, $\omega(v) = 0$. By Corollary 1, $|\tilde{H}_i| \leq 7|\bar{H}_i| \leq 7h$.

We now apply Theorem 2 to compute a partition of $\tilde{H}_i$ into subgraphs $\tilde{H}_{i,1}, \ldots, \tilde{H}_{i,k_i}$ of weight at most $\sqrt{h}/2$ each. By Theorem 2, the total size of the separators computed for all graphs $\tilde{H}_1, \ldots, \tilde{H}_s$ is $O(N/\sqrt{h})$, and each subgraph $\tilde{H}_{i,j}$ of $\tilde{H}_i$ is adjacent to at most $\sqrt{h}/2$ separator vertices. Unfortunately, some of these vertices may have a large weight; that is they correspond to many separator vertices in $\partial \bar{H}_i$. Thus, even

though a subgraph $\tilde{H}_{i,j}$ is adjacent to at most $\sqrt{h}/2$ separator vertices, the corresponding subgraph $\bar{H}_{i,j} = \bar{H}_i \cap \tilde{H}_{i,j}$ of $\bar{H}_i$ may be adjacent to many separator vertices. However, each separator vertex of large weight is the result of compressing a number of separator vertices in $\partial\bar{H}_i$ into a single vertex. As we do this only for vertices adjacent to at most two vertices in $\bar{H}_i$, every separator vertex of large weight is adjacent to at most two vertices in $\tilde{H}_{i,j}$. Thus, we modify the obtained separator as follows. For every separator vertex of weight two or greater adjacent to a vertex $v \in \tilde{H}_{i,j}$, we add v to the separator computed for $\tilde{H}_i$. As a result we obtain a separator whose size is at most twice the size of the separator computed for $\tilde{H}_i$; every subgraph $\bar{H}_{i,j}$ of $\bar{H}_i$ is now adjacent to at most $\sqrt{h}$ separator vertices.

Let S'' be the set of all separator vertices introduced in this step, and let $S = S'_0 \cup S''$ be the final separator. By Theorem 2, $|S| = O(N/\sqrt{h})$, and the number of subgraphs $\bar{H}_{i,j}$ obtained by removing the vertices in S from G is $O(N/h)$, so that we obtain the main result of our paper.

Theorem 3 *Given a planar graph G and an integer $h > 0$, it takes $O(\mathrm{sort}(N))$ I/Os and $O(N/B)$ blocks of external memory to compute a separator S of size $O(N/\sqrt{h})$ whose removal partitions G into $O(N/h)$ subgraphs of size at most h and boundary size at most $\sqrt{h}$, provided that $M \geq h \log^2 B$.*

3 Applications

In this section we show how to use Theorem 3 to solve the following three fundamental problems on embedded planar graphs I/O-efficiently:

Breadth-first search: Given an undirected graph $G = (V,E)$, compute a spanning tree T of G rooted at some source vertex s so that for every edge $\{v,w\} \in E$, the distances from s to v and from s to w in T differ by at most one, where the distance between two vertices is the number of edges in the path between the two vertices.

Depth-first search: Given an undirected graph $G = (V,E)$, compute a spanning tree T of G rooted at some source vertex s so that for every edge $\{v,w\} \in E$, v is an ancestor of w in T or vice versa.

Single source shortest paths: Given an undirected graph $G = (V,E)$ and an assignment $w : E \to \mathbb{R}_0^+$ of non-negative weights to the edges of G, compute a spanning tree T of G rooted at some source vertex s so that T is the union of the shortest paths from s to all vertices in G, where the shortest path from s to some vertex v is the path whose edges have minimum total weight among all paths from s to v.

We prove the following theorem.

Theorem 4 *It takes $O(\mathrm{sort}(N))$ I/Os and $O(N/B)$ blocks of external memory to compute BFS and DFS trees of an embedded planar graph G of size N and to solve the single source shortest-path problem on G, provided that $M \geq B^2 \log^2 B$.*

We show that SSSP can be solved in $O(\mathrm{sort}(N))$ I/Os. The result on BFS then follows from the fact that BFS is the single source shortest path problem with $w(v) = 1$, for all $v \in G$. The result on DFS follows from the $O(\mathrm{sort}(N))$ I/O reduction of DFS to BFS shown in [6].

We use the algorithm of [5] to solve the SSSP problem given a small separator of an embedded planar graph. This algorithm uses the notion of boundary sets, as introduced in [16], in order to achieve I/O-efficiency. Given a separator S whose removal partitions a planar graph G into subgraphs $H_1, \ldots, H_q$, the vertices in S can be partitioned into maximal subsets $\mathcal{B}_1, \ldots, \mathcal{B}_t$ such that the vertices in each set $\mathcal{B}_i$ are adjacent to the same set of subgraphs H_i. We call $\mathcal{B}_1, \ldots, \mathcal{B}_t$ the *boundary sets* defined by the separator S and the subgraphs $H_1, \ldots, H_q$. The following theorem provides the tool for solving the SSSP problem on embedded planar graphs.

Theorem 5 [5] *Let G be an embedded planar graph of size N and bounded degree, and S be a set of $O(N/B)$ vertices $O(N/B)$ whose removal partitions G into $O(N/B^2)$ subgraphs $H_1,\ldots,H_q$ with the following properties:*

(i) $|H_i| \leq B^2$, $1 \leq i \leq q$,

(ii) $|\partial H_i| \leq B$, $1 \leq i \leq q$,

(iii) The separator S and subgraphs $H_1,\ldots,H_q$ define $O(N/B)$ boundary sets $\mathcal{B}_1,\ldots,\mathcal{B}_t$.

Then the single source shortest path problem in G can be solved in $O(\mathrm{sort}(N))$ I/Os using $O(N/B)$ blocks of external memory.

In order to solve the SSSP problem for any embedded planar graph G, we first transform it to an embedded planar graph G' of size $O(|G|)$ whose vertices have degree at most 3. We will ensure that the weighted distances between the vertices in G and the corresponding vertices in G' are preserved. Given G', we apply Theorem 3 with parameter $h = B^2$ to G' to obtain a separator S whose removal partitions G' into $O(N/B^2)$ subgraphs of size at most B^2 and boundary size at most B. Finally we regroup the connected components of $G' - S$ in a manner that reduces the number of boundary sets to $O(N/B^2)$.

The construction of G' from G replaces every vertex v of G whose degree d_v exceeds 3 by a cycle $v_0,\ldots,v_{d_v-1}$ of vertices. Let $w_0,\ldots,w_{d_v-1}$ be the neighbors of v in G in clockwise order around v. Then we replace edges $\{v,w_i\}$, $0 \leq i < d_v$, by edges $\{v_i,w_i\}$, $0 \leq i < d_v$. For every edge $e = \{v_i, v_{(i+1) \bmod d_v}\}$ in the cycle representing vertex v, we define its weight $w(e) = 0$. For edges $\{v_i,w_i\}$, $0 \leq i < d_v$, we define $w(\{v_i,w_i\}) = w(\{v,w_i\})$. Then it is easily verified that G' is planar, has size $O(|G|)$, and that for any two vertices $v,w \in G$, $\mathrm{dist}_G(v,w) = \mathrm{dist}_{G'}(v_i,w_j)$, $0 \leq i < d_v$, $0 \leq j < d_w$, where $v_0,\ldots,v_{d_v-1}$ and $w_0,\ldots,w_{d_w-1}$ are the vertices in G' representing v and w, respectively. Given the embedding of G, the construction of G' takes $O(\mathrm{sort}(N))$ I/Os.

In order to reduce the number of boundary sets of $G' - S$, let $Q_1,\ldots,Q_r$ be the connected components of $G' - S$. We construct a graph $\tilde{G}$ containing one vertex v_i per connected component Q_i and an edge between two vertices v_i and v_j if $\partial Q_i \cap \partial Q_j \neq \emptyset$. Since the vertices in G' have degree at most 3, $\tilde{G}$ is planar, and $\sum_{i=1}^{q} |\partial Q_i| = O(|S|) = O(N/B)$. We now assign weights $\omega(v_i) = |Q_i|$ and $\gamma(v_i) = |\partial Q_i|$ to vertices $v_1,\ldots,v_r$, and apply our, by now standard, contraction procedure to $\tilde{G}$. That is, as long as there is an edge $\{v,w\}$ such that $\omega(v) + \omega(w) \leq B^2$ and $\gamma(v) + \gamma(w) \leq B$, we contract this edge and repeat. We call a vertex in the resulting compressed version of $\tilde{G}$ *heavy* if either $\omega(v) \geq B^2/2$ or $\gamma(v) \geq B/2$, and light otherwise. We partition the light vertices of degree at most two into equivalence classes such that the vertices in the same equivalence class are adjacent to the same set of (heavy) neighbors. Then we merge vertices in the same equivalence class until no class contains more than one light vertex. As there are $O(N/B^2)$ heavy vertices, the total size of the graph $\bar{G}$ obtained after these contractions is $O(N/B^2)$, by Corollary 1. Every vertex $v \in \bar{G}$ represents a subgraph H_i of $G' - S$ consisting of a number of connected components Q_j. It is easily shown that graphs $R_i = H_i \cup \partial H_i$ are of two possible types:

(i) Either R_i is connected or

(ii) R_i shares vertices with at most two other graphs R_j and R_k, which are connected.

It now follows from arguments found in [16] that the separator S and graphs $H_1,\ldots,H_q$ define $O(N/B^2)$ boundary sets. As it takes $O(\mathrm{sort}(N))$ I/Os to construct G' from G, compute the separator S for G', and group the connected components of $G' - S$ into the desired subgraphs $H_1,\ldots,H_q$, this proves Theorem 4.

4 Conclusions

We have provided a fairly simple and practical algorithm for computing separators of planar graphs which leads to $O(\text{sort}(N))$ I/O solutions for BFS, DFS, and SSSP on embedded planar graphs, provided that $M \geq B^2 \log^2 B$. Due to the constraints of the SSSP algorithm of [5], reducing the memory requirement for our separator algorithm below B^2 does not lead to improved memory requirements for BFS, DFS, and SSSP. It is a challenging open problem to design a separator algorithm that takes $O(\text{sort}(N))$ I/Os for $M \leq B^2$. If the number of disks is $D = 1$, we may increase the size of the separator to $O(\text{sort}(N))$ instead of $O(N/B)$ in our algorithm. This still leads to $O(\text{sort}(N))$ I/O algorithms for SSSP, BFS, and DFS and reduces the memory requirements of our algorithm to $M \geq \frac{B^2 \log^2 B}{\log^2_{M/B} N}$. For $\log N \geq \log^2 B$, this implies that our algorithm requires only $M \geq B^2$. In practice, however, this constraint is never satisfied.

Although our separator algorithm does not require an embedding of the planar graph to be given as part of the input, the degree reduction of the graph applied in Section 3 requires an embedding, as does the reduction from DFS to BFS of [6], so that we need to compute an embedding of the given graph in order to solve any of the applications discussed in this paper. In practice, the requirement that the given graph be given together with an embedding is not a serious constraint, as in many large scale applications dealing with planar graphs such as GIS, we know that the graph is planar only because we are given an embedding. From a theoretical point of view, however, I/O-efficient algorithms for planarity testing and computing a planar embedding would be desirable, in order to have a firm grip on planar graphs as an abstract graph class in external memory.

Acknowledgments. I would like to thank Lyudmil Aleksandrov and Anil Maheshwari for helpful discussions on planar separators and external memory issues.

References

[1] J. Abello, A. L. Buchsbaum, and J. Westbrook. A functional approach to external graph algorithms. In *Proceedings of the 6th European Symposium on Algorithms*, pages 332–343, 1998.

[2] P. Agarwal, L. Arge, T. Murali, K. Varadarajan, and J. Vitter. I/O-efficient algorithms for contour-line extraction and planar graph blocking. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 117–126, 1998.

[3] L. Aleksandrov and H. Djidjev. Linear algorithms for partitioning embedded graphs of bounded genus. *SIAM Journal of Discrete Mathematics*, 9:129–150, 1996.

[4] L. Arge. The buffer tree: A new technique for optimal I/O-algorithms. In *Proceedings of the Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 334–345, 1995.

[5] L. Arge, G. S. Brodal, and L. Toma. On external memory MST, SSSP, and multi-way planar separators. In *Proceedings of SWAT'2000*, 2000.

[6] L. Arge, U. Meyer, L. Toma, and N. Zeh. On external-memory planar depth first search. In *Proceedings of WADS'2001*, 2001. to appear.

[7] K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and Systems Science*, 13:335–379, 1976.

[8] A. Broder, R. Kumar, F. Manghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web: Experiments and models. *Computer Networks and ISDN Systems*.

[9] A. L. Buchsbaum and J. R. Westbrook. Maintaining hierarchical graph views. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 566–575, 2000.

[10] Y.-J. Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and J. S. Vitter. External-memory graph algorithms. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1995.

[11] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *Journal of Computer and Systems Science*, 30(1):54–76, 1985.

[12] K. Diks, H. N. Djidjev, O. Sykora, and I. Vrto. Edge separators of planar and outerplanar graphs with applications. *Journal of Algorithms*, 14:258–279, 1993.

[13] H. N. Djidjev. Partitioning graphs with costs and weights on vertices: Algorithms and applications. volume 1284 of *Lecture Notes in Computer Science*, pages 130–143. Springer Verlag, 1997.

[14] H. N. Djidjev and J. R. Gilbert. Separators in graphs with negative and multiple vertex weights. Technical Report TR94-226, Department of Computer Science, Rice University, April 1994.

[15] S. Even and R. E. Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2:339–344, 1976.

[16] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, December 1987.

[17] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

[18] D. Hutchinson, A. Maheshwari, and N. Zeh. An external memory data structure for shortest path queries. In *Proceedings of the 5th ACM-SIAM Computing and Combinatorics Conference*, volume 1627 of *Lecture Notes in Computer Science*, pages 51–60. Springer Verlag, July 1999. To appear in Discrete Applied Mathematics.

[19] V. Kumar and E. J. Schwabe. Improved algorithms and data structures for solving graph problems in external memory. In *Proceedings of the 8th IEEE Sumposium on Parallel and Distributed Computing*, October 1996.

[20] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs: International Symposium (Rome 1966)*, pages 215–232, New York, 1967. Gordon and Breach.

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

[22] A. Maheshwari and N. Zeh. External memory algorithms for outerplanar graphs. In *Proceedings of the 10th International Symposium on Algorithms and Computation*, volume 1741 of *Lecture Notes in Computer Science*, pages 307–316. Springer Verlag, December 1999.

[23] A. Maheshwari and N. Zeh. I/O-efficient algorithms for graphs of bounded treewidth. In *Proceedings of SODA'2001*, pages 89–90, 2001.

[24] K. Mehlhorn and P. Mutzel. On the embedding phase of the hopcroft and tarjan planarity testing algorithm. *Algorithmica*, 16:233–242, 1996.

[25] U. Meyer. External memory bfs on undirected graphs with bounded degree. In *Proceedings of SODA'2001*, pages 87–88, 2001.

[26] K. Munagala and A. Ranade. I/O-complexity of graph algorithms. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1999.

[27] J. Vitter and E. Shriver. Algorithms for parallel memory I: Two-level memories. *Algorithmica*, 12(2–3):110–147, 1994.