Planar Graph Blocking for External Searching*

Surender Baswana† and Sandeep Sen‡

Department of Computer Science and Engineering,
Indian Institute of Technology Delhi,
Hauz Khas, New Delhi-110016, India.
E-mail:{sbaswana, ssen}@cse.iitd.ernet.in

April 18, 2002

Abstract

We present a new scheme for storing a planar graph in external memory so that any online path can be traversed in an I-O efficient way. Our storage scheme significantly improves the previous results for planar graphs with bounded face size. We also prove an upper bound on I-O efficiency of any storage scheme for well-shaped triangulated meshes. For these meshes, our storage scheme achieves optimal performance.

1 Introduction

There are many search problems like robot motion planning and searching in constraint networks that require online traversal in an undirected graph. A number of problems in computational geometry are also related to online traversal in graphs, namely, ray shooting in a simple polygon [4] and reporting intersection of a line segment with a triangulated mesh. Very often, the applications that require solving these problems are of very large scale, and therefore it is important to design I-O efficient algorithms for the online graph traversal problem. The I-O efficiency is related to the maximum number of I-O block transfers required to perform an arbitrary online traversal of any given graph. This is critically dependent on the scheme used for storing a graph in the external memory, which is often referred to as *Graph Blocking*. The efficiency of a blocking scheme is measured in terms of *speed-up*: σ, which is the average number of edge-traversals between two consecutive I-O operations. The efficiency of a blocking scheme is proportional to the value of σ, namely, a larger σ leads to better I-O performance. In this paper, we address the problem of blocking of planar undirected graphs.

We assume that the graph is of bounded degree, and a vertex is allowed to be present in more than one block. These assumptions are consistent with most of the applications of the graph-blocking problem. Suppose that a disk block can hold B vertices, and the internal memory can hold M vertices. The parameters B and M are related to the page size and the internal-memory

*A preliminary version of this paper appeared in the proceedings of FSTTCS 2000, LNCS vol. 1974, pp. 252-263.
†The work was supported in parts by a fellowship from Infosys Technologies Ltd., Bangalore.
‡The work was supported in parts by an AICTE career award.

size respectively (by a constant factor). At any stage of the online traversal, the next vertex to be visited can be any of the neighbors of the most recently visited vertex. With every vertex, we store its associated adjacency list; if a neighbor w of a vertex v is not present in the same block as that of v, we store the block address of w in the adjacency list of v.

Since the graph size is too large to be accommodated fully in the internal memory, at any stage, there will be some vertices which are not present in internal memory. Whenever a traversal reaches a vertex which is not present in internal memory, the block containing the adjacency list of that vertex has to be fetched from the disk. Let v be the last vertex visited, and let there be a vertex w at distance d from v, not present in internal memory. It is obvious that there is a path leading to w which will force an I-O operation with in the next d steps. Conversely, if all the vertices lying at distance d from v are present in the internal memory, then no block transfer can be forced in next d steps. The following definition will be useful for the remaining paper. For $k \leq B$, *k-neighborhood* of a vertex v is defined as the set of k vertices nearest to v. It is easy to see that growing a breadth-first-search (BFS) tree until we have included k vertices yields *k-neighborhood*[1] of v. Based on the discussion above, here is a naive blocking scheme for a graph $G(V, E)$:

For each $v \in V$, store the k-neighborhood of v in a block on the disk.

During an online traversal, whenever a vertex v is not found in internal memory, the block containing the k-neighborhood of v can be fetched from external memory. So the blocking scheme mentioned above ensures a speed-up of $r^-(k)$, which denotes the minimum depth of a BFS tree of size k in the given graph. However, this speed-up is achieved at the expense of a k-fold blow up in the storage requirement, which is not desirable for most practical situations. A blocking scheme is said to be *space optimal* if the number of blocks required to store a graph is at most a constant multiple of the minimum number of blocks required to store the graph. Like other previous approaches to graph blocking [1, 3], we do not consider the cost of preprocessing.

1.1 Previous Work

Goodrich et al. [3] presented a space optimal blocking schemes for grid graphs and complete d-ary trees. They derived a nontrivial upper bound of $r^+(B)$ on the speed-up achievable in any graph, where $r^+(B)$ denotes the maximum depth of B-size BFS tree in the given graph. They also gave a space optimal blocking scheme for the family of graphs with bounded $\frac{r^+(B)}{r^-(B)}$. Although they did not present any blocking scheme for general graphs, they conjectured the existence of a scheme which achieves a speed-up of $r^-(B)$ with optimal storage. Agarwal et al. [1] gave a space optimal blocking scheme for planar graphs that achieves a speed-up of $r^-(\sqrt{B})$. For the family of planar graphs, $r^-(B)$ can lie anywhere between $\log_d B$ (e.g., a d-ary tree), where d is the maximum degree of a vertex in the graph, and B (e.g., a chain). For the smaller extreme (i.e., $r^-(B) \approx \log_d B$), the speed-up achieved by the blocking scheme of Agarwal et al. [1] is close to $r^-(B)$ or more precisely, $\sigma = \frac{r^-(B)}{2}$. But the speed-up deteriorates steadily from the ideal value of $r^-(B)$, with increasing $r^-(B)$. As a case in point, note that for a planar graph with $r^-(k) = k^\alpha$ (where α is some positive fraction), the speed-up achieved is only $B^{\frac{\alpha}{2}}$th fraction of $r^-(B)$. Therefore, in the case of planar graphs with $r^-(B) = \sqrt{B}$, the speed-up achieved is $B^{\frac{1}{4}}$. The gap between $r^-(B)$ and the speed-up achieved widens even further for the planar graphs with larger values of $r^-(B)$.

[1] a BFS tree of size $k \leq B$, rooted at v, may not be unique. But all of them will have same depth. It will soon become clear that for the purpose of this paper there is no need to distinguish among them; any of them can be considered *k-neighborhood* of v.

1.2 Our Results

We present an efficient blocking scheme for general planar graphs, with improved speed-up over that of Agarwal et al. [1]. Our blocking scheme guarantees a speed-up of $r^-(s)$, where s is the largest number $k \leq B$ such that $k \leq r^-(k)\sqrt{\frac{B}{c}}$, and c is the maximum face size in the graph. It can be seen that, for the family of planar graphs having small face size and $r^-(B) \geq \sqrt{B}$, the speed-up achieved by our blocking scheme is $\Omega(r^-(B))$. Unlike the speed-up achieved by blocking scheme of Agarwal et al. [1], which deviates further from $r^-(B)$ as $r^-(B)$ increases, the speed-up achieved by our blocking scheme approaches $r^-(B)$ as $r^-(B)$ increases. In fact, the speed-up matches $r^-(B)$ for $r^-(B) \geq \sqrt{B}$. There are a number of situations that involve planar graphs with small face size, like trees and geometric graphs (grids and meshes). For such graphs, our blocking scheme outperforms earlier blocking scheme.

We also prove a bound on the best speed-up achievable in a planar mesh in terms of degree of local uniformity of mesh and block size. Most of the meshes in practical applications possess a good degree of local uniformity. Intuitively speaking, these meshes are well-shaped. We prove that the worst-case speed-up achievable by the best scheme in a planar mesh is $O(\sqrt{B})$, where the constant of proportionality depends upon the degree of well-shapedness of the mesh. We use our blocking scheme of planar graphs to achieve matching speed-up in a planar mesh.

2 Efficient Blocking of Planar Graphs

In this section, we shall devise efficient blocking scheme for planar graphs that achieves an improved speed-up over earlier blocking scheme given by Agarwal et al. [1]. We extend the following idea of partitioning a planar graph for our blocking scheme. Consider a planar graph of size N partitioned into $O(N/B)$ regions with each region containing at most B vertices and surrounded by boundary vertices such that every path going from a vertex in one region to a vertex in another region will pass through one or more of these boundary vertices. By storing the *k-neighborhood* around every boundary vertex in a block and storing vertices of a region together in a block, the following block-transfer strategy (referred as $\mathcal{A}$ henceforth) achieves a speed-up of $\sigma = \Omega(r^-(k))$ for any online traversal.

Strategy $\mathcal{A}$: *Whenever the traversal extends to a vertex v, not present in internal memory, and v is a boundary vertex, then read the block storing its k-neighborhood from the disk; otherwise read the block corresponding to the region in which v lies.*

It is clear that for every two blocks that we read from disk, at least $r^-(k)$ vertices of a path are traversed. Thus the speed-up achieved is $\Omega(r^-(k))$. For achieving maximum speed-up but keeping space optimality, we have to choose the largest $k \leq B$ such that the space used for storing all the *k-neighborhoods* around the boundary vertices is $O(\frac{N}{B})$ blocks.

Agarwal et al. [1] gave an efficient blocking scheme along the lines of the idea outlined above. They used a technique developed by Fredrickson [2] for partitioning planar graphs. Based on the separator theorem of Lipton and Tarjan [5], Fredrickson gave an algorithm for partitioning a planar graph into $O(\frac{N}{B})$ regions, with each region having at most B vertices and a total of $O(\frac{N}{\sqrt{B}})$ boundary vertices. By storing each of $O(\frac{N}{B})$ regions in blocks, and storing the $\sqrt{B}$*-neighborhood* around every boundary vertex, strategy $\mathcal{A}$ achieves a speed-up of $\Omega(r^-(\sqrt{B}))$ and optimal storage space.

For the class of planar graphs with bounded degree, $r^-(B)$ can be as small as $\log_d B$ on one extreme (where d is the maximum degree of a vertex in a graph) and as large as B on the other extreme. For planar graphs with $r^-(B) \approx \log_d B$, the speed-up achieved, using the blocking scheme of Agarwal et al. [1] is $\frac{1}{2}\log_d B$, which is indeed close to $r^-(B)$. Though the speed-up achieved is close to $r^-(B)$ for small values of $r^-(B)$, it degrades drastically as $r^-(B)$ increases. To appreciate this point, note that for planar graphs having $r^-(k) = k^\alpha$ (where $\alpha \leq 1$ is some constant), the speed-up is just $B^{\alpha/2}$ th fraction of $r^-(B)$.

We devise a refinement of the above mentioned blocking scheme for achieving a better speed-up. Note that the blocking scheme achieves a speed-up of $r^-(\sqrt{B})$ because we stored $\sqrt{B}$*-neighborhood* around every boundary vertex in the partition. To improve the speed-up, we aim to store a neighborhood of size greater than $\sqrt{B}$ around every boundary vertex. Since the number of boundary vertices is $\Omega(\frac{N}{\sqrt{B}})$, any attempt to increase the size of neighborhood around a boundary vertex beyond $\sqrt{B}$ will lead to nonlinear space (an undesirable situation).

2.1 An overview of our approach

We make the following useful observation: We need not store separate $\sqrt{B}$*-neighborhoods* for boundary vertices which are in close proximity. For example, let v be a boundary vertex, and $v_1, v_2, \ldots, v_j$ be other boundary vertices which lie within distance of $\frac{r^-(\sqrt{B})}{2}$ from v. Starting from any of these boundary vertices, we must traverse at least $\frac{r^-(\sqrt{B})}{2}$ steps to cross a $\sqrt{B}$*-neighborhood* of v (it follows from the triangle inequality). Therefore, instead of storing $\sqrt{B}$*-neighborhood* around every vertex $v_i \in v_1, \ldots, v_j$, we can just store $\sqrt{B}$*-neighborhood* around v only (as a common neighborhood for $v_1, \ldots, v_j$). In doing so, the speed-up is reduced at most by half; but we will be storing less than $\frac{N}{\sqrt{B}}$ neighborhoods. This reduction in the total number of neighborhoods allows us to increase the size of the neighborhoods (while still maintaining the linear space constraint).

In order to exploit the idea above, the partitioning scheme must ensure that the separator vertices be contiguous. The separator computed using the Lipton-Tarjan separator theorem [5] does not guarantee a separator with sufficiently *clustered* vertices. On the other hand, the planar-separator theorem given by Miller [6] shows the existence of a vertex or a cycle as a balanced separator for a planar graph. For the case when the separator is a cycle, we can form clusters of the separator-vertices by breaking the cycle into several chains of the same length. The set of vertices belonging to a chain define a cluster. We finally store just one neighborhood per cluster.

2.2 Our Blocking scheme

Having given the overview, we shall now formally describe the new blocking scheme and analyze the speed-ups that can be achieved for various classes of planar graphs. First we state the planar-separator theorem given by Miller [6]

Theorem 2.1 (Miller) *If G is embedded planar graph consisting of N vertices, then there exists a balanced separator which is a vertex or a simple cycle of size at most $2\sqrt{2.\lfloor\frac{c}{2}\rfloor N}$, where c is maximum face size. Such a separator is constructible in linear sequential time.*

Let $G(V, E)$ be the given planar graph with maximum face size equal to c, and N be the number of vertices of the graph. If $N < B$ we store G in a block on disk; otherwise we proceed as follows: we compute a separator using Miller's theorem given above. If the separator is a vertex v, we store the *B-neighborhood* around v; otherwise (the separator is a cycle $\mathcal{C}$ of size $\leq 2\sqrt{cN}$) let s be a number in the range $(\sqrt{B}, B)$ that will be specified later. Pick every $\frac{r^-(s)}{2}$th vertex of $\mathcal{C}$ to form a set S. For every vertex $v \in S$, store the *s-neighborhood* of v in a block. Associate every vertex w of separator $\mathcal{C}$ with the block that contains the *s-neighborhood* of $v \in S$ nearest to w. Let us denote the block associated with a boundary vertex w by B_w. Whenever a path extends to a boundary vertex w, and w is not present in internal memory, we shall bring the block containing B_w into internal memory. Now let the separator $\mathcal{C}$ partition V into two subsets P_1 and P_2, each of size at most $\frac{2}{3}N$. We recursively carry out blocking of subgraphs induced by P_1 and P_2.

It can be verified that by using the block-transfer strategy $\mathcal{A}$, the speed-up achieved by the blocking scheme described above is $\Omega(r^-(s))$. So a larger value of s will result in a larger speed-up. For a given s, the total space used for blocking satisfies the following recurrence:

$$\mathcal{S}(N) = \mathcal{S}(N_1) + \mathcal{S}(N_2) + \frac{4\sqrt{cN}}{r^-(s)}s \qquad \text{where} \qquad N_1, N_2 < \frac{2N}{3}$$

the solution of which is

$$\mathcal{S}(N) = c_1 N + c_2 \frac{\sqrt{c}N}{\sqrt{B}r^-(s)}s$$

where c_1, c_2 are constants independent of s.
To maximize s and keeping linear space constraint$(\mathcal{S}(N) = O(N))$, we choose $s = \min\left(r^-(s)\sqrt{\frac{B}{c}}, B\right)$; i.e. s is the largest number $k \leq B$ such that $k \leq r^-(k)\sqrt{\frac{B}{c}}$.

Theorem 2.2 *A planar graph of size N and maximum face size c can be stored in $O(\frac{N}{B})$ blocks so that any online path of length t can be traversed using $O\left(\frac{t}{r^-(s)}\right)$ I-O operations, where $s = \min\left(r^-(s)\sqrt{\frac{B}{c}}, B\right)$.*

The new blocking scheme gives improvement in speed-up for planar graphs with bounded face size. The improvement achieved is most significant in case of graphs with $r^-(k) = k^\alpha$ (the graphs for which the previous blocking schemes are not effective).

Remark 1. For planar graphs with $r^-(B) = \Omega(\sqrt{B})$ and maximum face size $= c$, value of s is equal to $\frac{B}{c}$. So the new blocking scheme achieves a speed-up equal to $r^-\left(\frac{B}{c}\right)$, which is significantly better for planar graphs with *small* c than the previous speed-up of $r^-(\sqrt{B})$ in Agarwal et al. [1].
Remark 2. For a tree, the new blocking scheme achieves speed-up of $r^-(B)$.
Remark 3. For planar graphs with $r^-(k) = k^\alpha$, for some constant $\alpha \leq \frac{1}{2}$, value of s is $\left(\frac{B}{c}\right)^{\frac{1}{2(1-\alpha)}}$. Thus the speed up achieved by the new blocking scheme is $\left(\frac{B}{c}\right)^{\frac{\alpha}{2(1-\alpha)}}$ which is a significant improvement for planar graphs with *small* c (maximum face size) over the previous speed-up of $B^{\frac{\alpha}{2}}$ achieved by the blocking scheme of Agarwal et al. [1].

3 Blocking of Planar Meshes

In various problems of scientific computing, the underlying graphs are often defined geometrically; for example, grid graphs and graphs in VLSI technology. In addition to combinatorial structure, these graphs also have geometric structure associated with them. One such family of graphs is the *mesh*. A mesh in d-dimensional space is a subdivision of a d-dimensional domain into simplices which meet only at shared faces, e.g., a mesh in 2-dimension is a triangulation of a planar region, where triangles intersect only at shared edges and vertices.

Unlike a grid graph, where the edges are of same length and placement of vertices has high degree of symmetry, a mesh need not be uniform and symmetric. We define two parameters α, γ to be associated with a planar mesh which (intuitively speaking) measure its *well-shapedness*. We address the problem of blocking of planar meshes. We present two results: First, we show that the worst-case speed-up achievable by the best scheme in a planar mesh is $O(\sqrt{B})$, where the constant of proportionality depends upon parameters α, γ. Next, we use the blocking scheme for planar graphs described in the previous section to achieve a speed-up of $\Omega(\sqrt{B})$, where the constant of proportionality that depends upon α, γ becomes smaller as well-shapedness of mesh reduces. Thus for meshes having good degree of well-shapedness, the speed-up achieved by the blocking scheme matches the best possible.

3.1 Well-Shaped Planar Meshes

A planar mesh is a triangulation of a region in 2-dimensions, where the triangles meet only at shared edges and vertices. For simplicity, we assume that a planar mesh extends infinitely in all directions (e.g., a mesh embedded on a torus or a sphere). A planar mesh need not possess perfect uniformity and symmetry like a grid graph. There may be variations in the edge-lengths and density of vertices as we move from one region to another region in the mesh. But as observed in most of applications, there is certain degree of local uniformity present in mesh, i.e., in a neighborhood around a vertex there is not *too much* variation in edge-lengths and vertex-density though the variation may be unbounded for the whole mesh. In visualizing mesh as a triangulation, this local uniformity can be viewed in the following way: the triangles constituting the mesh are fat and the variation of the size (area) of these triangles is bounded in a finite neighborhood. This local uniformity captures formally the notion of well-shapedness of a planar mesh. We now define parameters to measure the local uniformity of a planar mesh. We parameterize fatness of triangles by the smallest angle α of a triangle in planar mesh. We parameterize the variation in sizes of triangles within a B-neighborhood in the following way : Let u be a vertex and B_u be the set of vertices of a BFS tree of size B rooted at u. Let $\triangle_B^u$ be the set of triangles with at least one vertex belonging to B_u. γ is defined as the ratio of area of the largest-area triangle to area of the smallest-area triangle belonging to the set $\triangle_B^u$. The parameters α and γ measure local uniformity of a planar mesh.

The area of any triangle in the set $\triangle_B^u$ lies in the range $[A, \gamma A]$ for some A. Using elementary geometry, it can be shown that the length of any edge e in the subgraph induced by B_u has the following bounds:

$$e_{min} = 2\sqrt{A \tan \frac{\alpha}{2}} \leq 2\sqrt{\gamma A \cot \alpha} = e_{max} \tag{1}$$

Lemma 3.1 *The value of $r^-(B)$ of a planar mesh with parameters α, γ is $\Omega(\frac{\sqrt{\tan \alpha}}{\sqrt{\gamma}}\sqrt{B})$.*

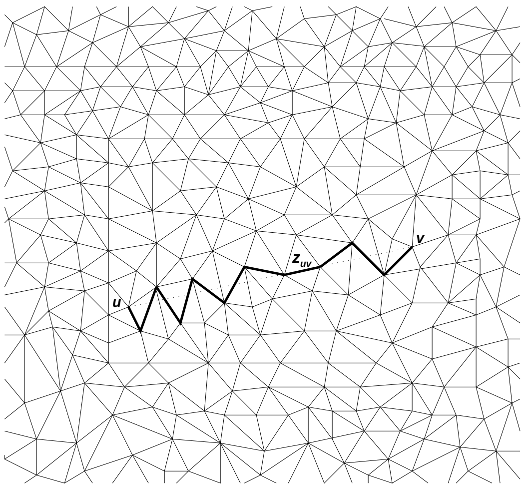


Figure 1: Zig-zag path z_{uv} between vertices u and v of neighborhood B_u in a planar mesh

Proof: Let u be an arbitrary vertex of the planar mesh, and let l be the depth of BFS tree of size B, rooted at u. Let w be a vertex of B_u at maximum Euclidean distance d_{max} from u. Consider a circle C centered at u with radius d_{max}. The number of vertices lying in the circle is at least B since B_u lies inside it. Thus the number of triangles lying inside the circle is at least $\frac{B}{3}$. Note that the maximum number of triangles lying inside a circle of radius d_{max} is $\leq \frac{\pi d_{max}^2}{A}$. Hence the following inequality must hold: $\frac{\pi d_{max}^2}{A} \geq \frac{B}{3}$, i.e., $d_{max} \geq \sqrt{\frac{A}{\pi}}\sqrt{\frac{B}{3}}$. Also note that d_{max} is bounded by $l \cdot e_{max}$. Thus $l \geq \frac{\sqrt{A}}{\sqrt{\pi} e_{max}}\sqrt{\frac{B}{3}}$. Using the bound on e_{max} from equation 1, and the definition of $r^-(B)$, it follows that $r^-(B) = \Omega(\frac{\sqrt{\tan \alpha}}{\sqrt{\gamma}}\sqrt{B})$. □

For a planar mesh, the following theorem gives a lower bound on the Euclidean distance between two vertices in terms of the length of the shortest path separating them.

Theorem 3.1 *Let v be a vertex belonging to B_u in a planar mesh. If p_{uv} is the shortest path-length from u to v, the Euclidean distance d_{uv} between u and v is $\Omega(p_{uv}\sqrt{A})$, where the constant of proportionality depends upon the well-shapedness parameters (α, γ) of the mesh.*

Proof: Let w be a boundary vertex of the neighborhood B_u lying at the closest Euclidean distance from u, and $\mathcal{N}$ be the set of vertices lying within distance d_{uw} from u. It can be seen that $\mathcal{N} \subset B_u$. To prove the theorem, it suffices to show that for a vertex $v \in \mathcal{N}$, separated by a shortest path of length l from u, the Euclidean distance d_{uv} is $\Omega(l\sqrt{A})$. We proceed as follows: Let v be a vertex belonging to $\mathcal{N}$, and S be the line segment joining u and v. We build a path z_{uv} as we move along S from u. Let p denote the vertex most recently added to the path we are building (initially $p = u$). We add edges to our path maintaining the following invariant :

I : *p lies on the edge most recently intersected by S; and every edge forming the path has some point in common with S.*

While moving along S building the path, let e be an edge intersected by S. If e contains p, we keep p unchanged. Otherwise (e does not contain p), there is one end point, say q, of edge e adjacent to p such that pq is the valid edge to be added to our path maintaining the invariant I. So we extend our path by adding the edge pq to it, and p gets updated to q. We continue this process until we reach v. In the special case when S passes through a vertex, say x, we update p to x. We shall now bound length of this zig-zag shaped path z_{uv}. The segment S intersects triangles of $\triangle_B^u$ only, therefore the ratio of areas of intersected triangles is bounded by γ. Path z_{uv} divides the segment S into subsegments whose total number is equal to the number of vertices lying on the path z_{uv} excluding u and v. Consider any three consecutive edges of z_{uv}. There will be one (or two adjacent) subsegment(s) of S intercepted between these three edges. Because of the constraints imposed by bounded α and γ, the length of the intercepted subsegment (or sum of the lengths of two intercepted subsegments) is at least $e_{min}\sin\alpha$. Hence the number of subsegments into which the segment S is divided by the path z_{uv} (and so length of the path z_{uv}) is at most $\frac{d_{uv}}{e_{min}\sin\alpha}$. Using the bound on e_{min} from equation 1, it follows that the length of the path z_{uv} is at most $\frac{d_{uv}}{c_\alpha\sqrt{A}}$, where $c_\alpha = 2\sqrt{\tan\frac{\alpha}{2}}\sin\alpha$.

Since the length p_{uv} of the shortest path between u, v is less than or equal to the length of the path z_{uv}, we have :

$$p_{uv} \le \left(\frac{d_{uv}}{c_\alpha\sqrt{A}}\right) \tag{2}$$

In other words, if v is a vertex belonging to B_u and separated by a path of length p_{uv} from v, then the Euclidean distance d_{uv} between u, v has the following lower bound :

$$d_{uv} \ge c_\alpha p_{uv}\sqrt{A} \tag{3}$$

□

We showed in Lemma 3.1 that the shortest path-length from u to a boundary vertex of B_u is $r^-(B) = \Omega(\frac{\sqrt{\tan\alpha}}{\sqrt{\gamma}}\sqrt{B})$. By substituting the value of $r^-(B)$ for p_{uv} in equation 3, we get the following corollary:

Corollary 3.1.1 *The minimum Euclidean distance between u and boundary vertex of B_u in a planar mesh is $\rho \ge c_\alpha\sqrt{\frac{\tan\alpha}{\gamma}}\sqrt{B}\sqrt{A}$.*

We now state the following Lemma which gives an upper bound on $r^+(B)$:

Lemma 3.2 *The value of $r^+(B)$ for a planar mesh with parameters α, γ is $O\left(\frac{\sqrt{\gamma}}{c_\alpha}\sqrt{B}\right)$.*

Proof: Let u be an arbitrary vertex of the planar mesh, and let v be a boundary vertex of B_u at minimum Euclidean distance d_{min} from u. Now consider a circle C centered at u, with radius d_{min}. Clearly C lies completely inside B_u, and so the number of vertices lying inside C is at most B. From elementary geometry, it follows that the circle C contains at least $\frac{3\pi d_{min}^2}{\gamma A}$ vertices. Therefore $\frac{3\pi d_{min}^2}{\gamma A} \le B$. So $d_{min} \le \sqrt{\frac{\gamma BA}{3\pi}}$. Now using the lower bound on d_{uv}, from equation 3 of the theorem given above, it follows that $p_{uv} \le \frac{\sqrt{\gamma}}{c_\alpha\sqrt{3\pi}}\sqrt{B}$. In other words, a boundary vertex of B_u is separated from u, by a path of length $O\left(\frac{\sqrt{\gamma}}{c_\alpha}\sqrt{B}\right)$, i.e., $r^+(B) = O\left(\frac{\sqrt{\gamma}}{c_\alpha}\sqrt{B}\right)$ □

3.2 Upper Bound on the Speed-up for Planar Meshes

Goodrich et al. [3] proved an upper bound of $r^+(B)$ on the worst-case speed-up achievable by the best scheme in a graph. We showed in the previous subsection (Lemma 3.2) that $r^+(B)$ for a planar mesh is $O(\frac{\sqrt{\gamma}}{c_\alpha}\sqrt{B})$. We can now state the following theorem:

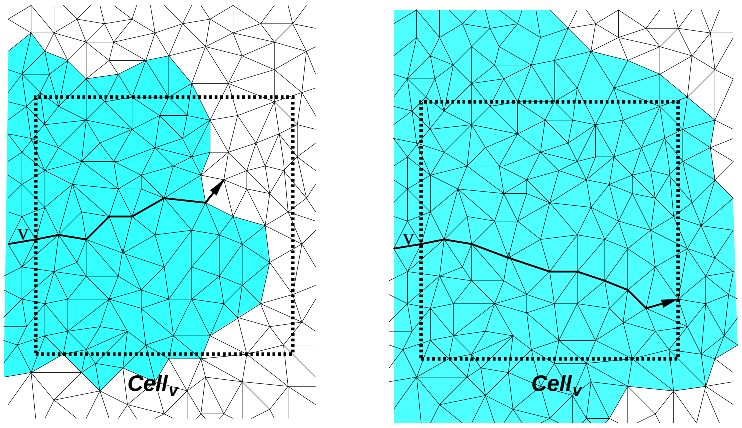
Theorem 3.2 *For a planar mesh, the worst-case speed-up that can be achieved by the best blocking scheme is $\sigma = O(\mathcal{C}_{\alpha\gamma}\sqrt{B})$, where $\mathcal{C}_{\alpha\gamma}$ is a constant depending upon the parameters (α, γ) that capture the well-shapedness of the mesh.*

For the sake of completeness, we give an alternate proof for the upper bound on the speed-up in a planar mesh. Consider a planar mesh in the $x - y$ plane. We present a traversal strategy which will ensure one block transfer on average for every $O(\sqrt{B})$ steps traversed, irrespective of the underlying blocking scheme. Define a vertex to be *covered* if it happens to be in internal memory at least once. Initially, before starting traversal, no vertex is present in internal memory, and so all the vertices are uncovered. Let u be the most recently visited vertex (path-front). It is obvious that if there is an uncovered vertex separated by a path of length $\leq \sqrt{B}$ from u, we can extend our path to that uncovered vertex (and thus force a block transfer in $\sqrt{B}$ steps). But what if all the vertices separated by paths of length $\leq \sqrt{B}$ from u are covered? Note that at least one block-transfer is required to cover a set of B uncovered vertices. So in case there is no uncovered vertex separated by path of length $\sqrt{B}$ from the path-front, we move the next $\sqrt{B}$ steps in such a way that we can associate distinct $\Omega(B)$ covered vertices to these steps. This would still imply that there is a block transfer after every $O(\sqrt{B})$ steps on average. This is the basic idea underlying the traversal strategy.

For a vertex u of mesh, $Cell_u$ denotes a square with base parallel to x-axis, and u lying on its left vertical side. Length of each of its four sides is chosen to be $\rho/2$, where ρ is the minimum Euclidean distance between u and a boundary vertex of B_u. It follows from Corollary 3.1.1 that the number of vertices lying in $Cell_u$ is more than $c_o B$ and every vertex of $Cell_u$ is reachable from u by path of length less than $c\sqrt{B}$ for some constants c, c_o depending upon well-shapedness of the mesh. Here is the traversal strategy :

We start from any vertex and always move rightward within the mesh. The path can be visualized as a sequence of sub-paths of type p' and p^x. At a point, let v be the most recently visited vertex. If there is any uncovered vertex inside $Cell_v$, we extend our path to that uncovered vertex (we call it sub-path of type p') and thus force a block-read from disk; otherwise we extend our path to a covered vertex lying closest to the right edge of $Cell_v$ (we call it sub-path of type p^x).

Let b be the number of block transfers encountered in traversing t steps in the mesh according to the strategy above. Every sub-path of type p' causes a block transfer. So the number of sub-paths of type p' is at most b. Also note that for every sub-path of type p^x, the number of covered vertices lying to the left of path-front in the mesh increases by $c_o B$. Thus we can associate a set of unique $c_o B$ covered vertices to a sub-path of type p^x (uniqueness follows from the unidirectionality of traversal). Since a block transfer can cover at most B vertices, it follows that the number of sub-paths of type p^x is at most $\frac{b}{c_o}$. So the total number of sub-paths (of type p' and p^x) is bounded by $\frac{2b}{c_o}$. Also note that the length of each sub-path is no more than $c\sqrt{B}$ (from definition of $Cell$ given above). Hence $t \leq \frac{2c\sqrt{B}b}{c_o}$ or in other words, the number of block-transfers b required to traverse t steps is $\Omega(\frac{t}{\sqrt{B}})$. Hence we can conclude that the traversal strategy described above will ensure



(i) (ii)

(i) Subpath of type ***p'*** if there is any uncovered vertex in ***Cell$_v$***

(ii) Subpath of type ***p_x*** if all the vertices of ***Cell$_v$*** are covered

Figure 2: Two types of sub-paths from a vertex v in the mesh (the vertices lying in the shaded region are covered vertices)

a block transfer after every $O(\sqrt{B})$ steps on an average, irrespective of the underlying blocking scheme of the mesh.

3.3 Efficient Blocking of Planar Meshes

We can block planar meshes efficiently using our blocking scheme for planar graphs described in Section 2. From Lemma 3.1, it follows that $r^-(k) = \Omega(\sqrt{\frac{\tan\alpha}{\gamma}}\sqrt{k})$ for $k \leq B$. Also note that face size in a planar mesh is 3. So It follows easily from Theorem 2.2 that our blocking scheme guarantees speed-up of $\Omega(\frac{\tan\alpha}{\gamma}\sqrt{B})$ in a planar mesh with parameters α, γ. In previous subsection we established an upper bound of $O(\sqrt{B})$ on the speed-up in a planar mesh. Thus our blocking scheme achieves optimal speed-up in a planar mesh having good degree of well-shapedness.

Theorem 3.3 *There is a space optimal blocking scheme which ensures a speed-up of* $\Omega(\frac{\tan\alpha}{\gamma}\sqrt{B})$ *in a planar mesh, where the parameters* (α, γ) *measure the well-shapedness of the mesh.*

4 Conclusions

We addressed the problem of planar graph blocking in this paper. We described a blocking scheme which guarantees improved speed-up in planar graphs of bounded face size over previous blocking schemes. We also established a bound on the worst-case speed-up that can be achieved by the best blocking scheme in a planar mesh. For planar meshes with good degree of well-shapedness (local uniformity), our blocking scheme achieves optimal speed-up.

There is still no space optimal blocking scheme which can ensure I-O efficient traversal in general graphs (not necessarily planar). Such a scheme will help solve a large number of problems which require I-O efficient traversal in general graphs.

References

[1] Pankaj K. Agarwal, Lars Arge, T.M. Murli, Kasturi R. Varadarajan, J.S. Vitter, I/O-efficient algorithms for contour-line extraction and planar graph blocking, *ACM-SIAM Symposium on Discrete Algorithms*, **9**(1998), 117-126.

[2] G.N. Fredrickson, Fast algorithms for shortest paths in planar graphs, with applications, *SIAM Journal of Computing*, **16**(1987), 1004-1022.

[3] M.T. Goodrich, M.H. Nodine and J.S. Vitter, Blocking for external graph searching, *Algorithmica*, **16**(2), August 1996, 181-214.

[4] John Hershberger and Subhash Suri, A pedestrian approach to ray shooting : shoot a ray, take a walk, *Journal of Algorithms*, **18**(1995), 403-432.

[5] R.J. Lipton and R.E. Tarjan, A separator theorem for planar graphs. *SIAM journal of Applied Math.*, **36**(1979), 177-189.

[6] G. Miller, Balanced cyclic separator for 2-connected planar graphs, *Journal of Computer and System Sciences*, **32**(1986), 265-279.