TECHNICAL REPORT NO. 316

Linear-Processor DNC Algorithms for Sublinear Separators in Planar Graphs

by

Gregory E. Shannon

October 1990

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
Bloomington, Indiana 47405-4101

Linear-Processor DNC Algorithms for Sublinear Separators in Planar Graphs*

Technical Report #316

Gregory E. Shannon
Department of Computer Science
Indiana University
Bloomington, Indiana 47405

October 10, 1990

Abstract

This report describes a sequence of linear-processor deterministic NC algorithms for finding sublinear-size cyclic separators in planar graphs. The sequence culminates in the first NC algorithm for finding an $O(\sqrt{n})$-size separator using only n processors. Previous algorithms for finding such a small separator have produced larger separators or used more time or processors, and some were randomized. Our results are based on a better understanding of the cumulative effects of recursively applying separators to subdivide a planar graph into equally-sized parts with small boundaries.

---

*This research has been supported in part by Hewlett-Packard's Faculty Development Program and NSF Grant CCR-89-09535.

1 Introduction

The problems of breadth-first-search (BFS) and single-source shortest paths are of fundamental importance in computational graph theory [1, 14, 18, 22]. For the important case of planar graphs, algorithms for finding $O(\sqrt{n})$-size separators have proven to be important tools in solving these two problems [5, 14, 17], and are also of independent importance [6, 7, 14, 15, 17]. (A vertex set S of an n-vertex graph G *separates* G if no connected component in $G - S$ has more than $\frac{2n}{3}$ vertices.)

Though these three problems have (near-)linear-time sequential algorithms [1, 13, 15, 18], none has an NC (polylog time) parallel algorithm which uses a linear number of processors. The NC algorithm for shortest paths in general digraphs still requires n^3 processors [11]. Therefore, there is a great deal of progress yet to be made in finding fast parallel algorithms with near-linear processor-time products for these well-studied problems.

We present here deterministic linear-processor NC algorithms for finding $O(n^\epsilon)$-size cyclic separators in n-vertex triangulated planar graphs where $\frac{1}{2} \leq \epsilon < 1$. To find an $O(n^{\epsilon_i})$-size separator, $\epsilon_i = \frac{1}{2} + \frac{1}{2}\alpha^i$, our algorithm uses $O(c^i \log^3 n)$ time and only n processors, for some constants $0 < \alpha < 1$ and $c \geq 1$. In practice, α is about 0.8 or 0.9. When i is $\Theta(\log\log n)$, then ϵ_i is $O(\sqrt{n})$ and the time is *polylog*. Our algorithms are for the concurrent-read concurrent-write parallel RAM (CRCW PRAM) model of computation. See [11] for more background on the PRAM model.

In [20] we establish the basic techniques for iteratively producing algorithms for finding successively smaller separators in planar graphs. Generally, these techniques imply linear-processor DNC algorithms, though the time to find an $O(\sqrt{n})$-size separator was non-polylog. The time is exponential in the number of iterations used to generate that algorithm. This time explosion is due to the $O(\log n)$ applications of the previous algorithm. Since $O(\log\log n)$ iterations are necessary, the algorithm for finding $O(\sqrt{n})$-size separators uses $O((\log n)^{d\log\log n})$

time for some constant $d > 0$. Our approach in this report is to reduce the number of calls to the previous algorithm from $O(\log n)$ to $O(1)$. With this change, the time complexity for the end algorithm falls to polylog, $O(c^{d \log\log n})$ for some constant $c > 1$.

The key insight is that a recursive decomposition of an (r, ϵ)-*division* can be done with Miller's separator algorithm [15] in the same way that a recursive decomposition of a triangulated graph is done in [20] and [5]. A new division from this approach does not have such small boundaries as does one implied by the algorithms in [20], but it embodies enough of an improvement in order for the general iterative amelioration scheme set up in [20] to work correctly.

Previous algorithms use more processors, are randomized, have worse processor-time products, or produce larger separators. See [20] for a full report on the previous work. Our algorithm for finding $O(\sqrt{n})$-size cyclic separators in planar graphs is the first one to use polylog time and n processors. In addition, it is deterministic.

We expect significant applications of our results in the design of efficient parallel algorithms based based on the work of Alon, Seymour, and Thomas in [2]. Their work shows that any class of graphs with a largest excluded minor of size $O(h)$ has a separator of size $O(h^{\frac{3}{2}}\sqrt{n})$.

Given an algorithm for finding (r, ϵ)-*divisions*, in the next section we show how to subdivide a planar graph into regions of size $O(r = n^{1-\frac{1}{M}})$ with fewer boundary vertices using only two levels of calls to the (r, ϵ)-*division* algorithm and a linear number of additional processors. In Section 3, we show how to use the these same techniques to guarantee that the boundaries of each region in the new division are smaller than those produced by the (r, ϵ)-*division* algorithm. With the algorithm from Section 2, this immediately implies an algorithm which produces an (r, ϵ'')-division where $\frac{1}{2} \leq \epsilon'' < \epsilon$. In Section 4, we show that repeated applications of the algorithm from Section 3 can be used to find an $(r, \frac{1}{2})$-division and an $O(\sqrt{n})$-size cyclic separator in a planar graph in polylog time using n processors. We close with some comments in Section 5. Since this report is a continuation of [20], refer to that report for most definitions and useful basic properties of planar graphs.

2 Subdividing (r, ϵ)-Divisions

In an earlier paper on finding (r, ϵ)-*divisions* of planar graphs [20], we showed how Frederickson's techniques for recursively subdividing a maximal planar graph with small separators produce interesting results even when the separator size is greater than $O(\sqrt{n})$ $(\omega(\sqrt{n}))$. In the linear-processor algorithms for finding sublinear separators, we required that new (r, ϵ)-*divisions* be found for a collection of triangular faces. We observed that this produced a great deal of recomputation of the lower quality separators and divisions, but it was not clear how to re-use division information. We can now show that the new division implied by one of the two regions created from (r, ϵ)-*division*-based separator is not necessarily a (r, ϵ)-*division* of the corresponding region.

Therefore, the question is, "Is all the effort used to find an (r, ϵ)-*division* of any use after it has been used to find one separator?" We answer *yes* by considering what happens in our and Frederickson's techniques when the restriction that each face is a triangle is removed. We find that nothing really changes except that the number of boundary vertices is increased by a factor equal to the maximum face size and the lower bound on the smallest region size possible is obviously higher.

Of particular interest and importance is that the number of boundary vertices generated in the new division is still an improvement by a constant factor in the exponent if the new region size is sublinear in the size of the faces. More specifically, consider finding a separator in an $(r = n^{1-\frac{1}{M}}, \epsilon)$-division D of an n-vertex triangulated planar graph. n^M is the number of processors needed to multiply two $n \times n$ matrices over the ring of integers in $O(\log n)$ time.

The value of M is bounded below by 2 and is currently bounded above by 2.376 [4]. However, the algorithms in this report still generally work as claimed if $M = 3$ is used (straightforward matrix multiplication).

The size of the separator using Miller's algorithm [15] is $O(n^{\epsilon'})$ where $\epsilon = \frac{1}{2M} + \epsilon(1 - \frac{1}{M})$ [20, 15, 9, 8]. If we recursively subdivide D with Miller's algorithm into new regions of size $s = n^{1-\frac{1}{2M}}$, we find we generate $O(\frac{n}{s^{1-\epsilon''}})$ boundary vertices where $\epsilon'' = \epsilon - \frac{1}{2} \cdot \frac{1-2\epsilon}{2M-1}$. The best we could have hoped for (as in [20]) is to generate only $O(\frac{n}{s^{1-\epsilon'}})$ boundary vertices. However, since $\epsilon' < \epsilon'' < \epsilon$ if $\epsilon > \frac{1}{2}$, we see that we're on the right track! That is, ϵ'' might be good enough for our purposes. Therefore, while the reuse of D doesn't give us the ϵ' we might want, it does give us an exponent ϵ'' sufficiently less than ϵ.

In the rest of this section and the next, we address the problem that though the total number of boundary vertices is well constrained, many faces might be too large (more than $s^{\epsilon''}$ boundary vertices on a face). Also, s is not small enough to generate an (r, ϵ'')-division and thereby enable us to iterate the process as in [20].

In our previous work [20] we focused on a general interpretation of Frederickson's r-division techniques. This time, however, we focus on a particular structure to make very clear how to apply this section's results in the following sections.

Let D be an $(r = n^{1-\frac{1}{M}}, \epsilon)$-division of a set F of n triangular faces from a planar graph. Each region in D contains between r and r^{ϵ} vertices and has a boundary of at most r^{ϵ} vertices. Let $s = n^{1-\frac{x-1}{xM}}$ for some $x > 1$. Use Miller's algorithm on D to recursively decompose D into regions of size s as in [20]. A new region's size is determined by the number of original faces from F that it contains. Use Miller's algorithm [15] to find the separators. (The processor complexity is always linear in n since there are at most $n^{\frac{1}{M}}$ faces in D.) The above algorithm is called ϵ-*Divide*. The lemma below characterizes the subdivision of F that ϵ-*Divide* produces.

Lemma 1 *Given an $(r = n^{1-\frac{1}{M}}, \epsilon)$-division D of a set F of n triangular faces from a planar graph, the algorithm* ϵ-Divide *divides F into $\Theta(n/s)$ regions of size $\Theta(s = n^{1-\frac{x-1}{xM}})$ with $O(\frac{n}{s^{1-\epsilon''}})$ boundary edges overall where* $\epsilon'' = \dfrac{\frac{1}{2} + x\epsilon(M-1)}{1 + x(M-1)}$.

Proof. As in [20, 5], there are $O(\log n)$ levels of recursion and $\Theta(n/s)$ regions containing $O(s)$ of F's faces.

The difficulty in bounding the number of boundary vertices is that in an (r, ϵ)-*division* each region certainly has only $O(r^{\epsilon})$ boundary vertices, but a constant factor of regions might contain only r^{ϵ} vertices instead of $O(r)$. Therefore, as the recursion in ϵ-*Divide* proceeds, some branches many contain *only* regions from the original division which each contain $O(r^{\epsilon})$ vertices. Since the ratio of the regions' boundaries to their sizes is equal, the regions effectively do not have small boundaries! The question, then, is whether this affects the number of boundary vertices in any significant way.

We contend that it does not. Consider the following simplified approach for an (r, ϵ)-*division* D with its $f = n^{\frac{1}{M}}$ faces. Each face has a boundary of $b = O(n^{\epsilon(1-\frac{1}{M})})$ vertices. With weight 1 on each region in D, use Miller's algorithm to subdivide D into $\Theta(f^{1-\frac{1}{x}})$ new regions, each of weight $t = f^{\frac{1}{x}}$.

What has this new subdivision accomplished? Each new region may contain up to $s = t * n^{1-\frac{1}{M}} = f^{\frac{1}{x}} * n^{1-\frac{1}{M}} = (n^{\frac{1}{M}})^{\frac{1}{x}} * n^{1-\frac{1}{M}} = n^{1-\frac{x-1}{xM}}$ original faces but possibly only $t * n^{\epsilon(1-\frac{1}{M})}$ original faces. Nevertheless, there are $O(n^{\frac{x-1}{xM}})$ new regions in D and F since there were originally only $O(n^{\frac{1}{M}})$ regions in D. By Miller's algorithm, the size of the boundary produced at each level of recursion is equal to the size of the maximum face boundary ($b = n^{\epsilon(1-\frac{1}{M})}$) times the root of the number of faces ($f = n^{\frac{1}{M}}$), $b\sqrt{f}$. Overall, the recurrence below characterizes the total number of boundary vertices necessary for the new subdivision.

$$
\begin{aligned}
B(f,t,b) &= \{0 && \text{if } f \le t \\
&= \{b\sqrt{f} + B(\alpha f, t, b) + B((1-\alpha)f, t, b) && \text{for } \tfrac{1}{3} \le \alpha \le \tfrac{1}{2}, \text{if } f > t
\end{aligned}
$$

Rearranging the recurrence, we have the total number of boundary vertices equal to $b \cdot B(f,t)$.

$$\begin{aligned} B(f,t) &= \{0 && \text{if } f \leq t \\ &= \{\sqrt{f} + B(\alpha f, t) + B((1-\alpha)f, t) && \text{for } \tfrac{1}{3} \leq \alpha \leq \tfrac{1}{2}, \text{ if } f > t \end{aligned}$$

A bound on this recurrence can be found in [5, 20] and is $B(f,t) \leq \frac{f}{\sqrt{t}}$. Therefore, the total number of boundary vertices is

$$\frac{bf}{\sqrt{t}} = \frac{n^{\epsilon(1-\frac{1}{M})} n^{\frac{1}{M}}}{\sqrt{n^{\frac{1}{zM}}}}. \tag{1}$$

Unfortunately, this not particularly useful since we really want to characterize the number of boundary vertices with an expression like $\frac{n}{s^{1-\epsilon''}}$ for some $\epsilon'' < \epsilon$. That is, ϵ'' characterizes the average size the of the boundary of each new region in the subdivision. Therefore, we solve the equation below for ϵ''.

$$\frac{n^{\epsilon(1-\frac{1}{M})} n^{\frac{1}{M}}}{\sqrt{n^{\frac{1}{zM}}}} = \frac{n}{\left(n^{1-\frac{z-1}{zM}}\right)^{1-\epsilon''}} \tag{2}$$

With simple algebra we see that ϵ'' is as stated in the lemma.

Note that the above method for finding a subdivision with respect to t can generate only *more* boundary vertices than *ϵ-Divide*. This is because *ϵ-Divide* proceeds exactly as above except that its recursion might be as deep. It cannot recurse deeper since the resulting regions would have fewer faces than t from D. ∎

Given the above lemma, we can now subdivide n faces into $\Theta(n^{\frac{1}{M}})$ regions, each of size $O(r = n^{1-\frac{1}{M}})$ with $O(\frac{n}{r^{1-\epsilon''}})$ boundary vertices given an algorithm for finding an $(r = n^{1-\frac{1}{M}}, \epsilon)$-division in a region F with n faces.

Consider the whole graph F as an initial new region. Iterate the following until there is no new region with more than r faces. For each new region with $R > r$ faces, find an $(R^{1-frac1M}, \epsilon)$-division of the new region. Use the algorithm *ϵ-Divide* to subdivide R into new regions of size $R^{1-\frac{z-1}{zM}}$.

If $x = 2$, then the above procedure has at most three iterations. After the first application of ϵ-*Divide*, each new region has at most $O(n^{1-\frac{1}{2M}})$ vertices. After the next application to the resulting regions, each region has at most $O(n^{(1-\frac{1}{2M})^2})$ vertices. Since $(1 - \frac{1}{2M})^2 > 1 - frac1M$, there must be one more application of ϵ-*Divide* to the new regions. After the third application, then $(1 - \frac{1}{2M})^3 < 1 - frac1M$, and the resulting new regions are then small enough. However, they have become too small and therefore also have too many boundary vertices for our purposes.

Rather than determine where to stop in the third iteration for $x = 2$, we will determine the largest x necessary in order for two and three iterations to be sufficient. In either case, all we need to do is solve the equation below for $i = 2$ and $i = 3$.

$$\left(1 - \frac{x-1}{xM}\right)^i = 1 - \frac{1}{M} \tag{3}$$

By numerical approximation, we see that the solutions are $x \geq x_2 = \frac{7}{3}$ for $i = 2$ and $x \geq x_3 = \frac{5}{3}$ for $i = 3$ if $M = 2.376$. However, we can just as well use $M = 3$. In fact, though we use $M = 2.376$ throughout this paper, all of the algorithms remain correct if $M = 3$. Some parameters might have to be adjusted and the algorithms will generally run slower, but the overall results in this paper will remain unchanged. Now we can prove the theorem below: the first phase in generating an (r, ϵ'')-division.

Theorem 1 *Let* DIV(r, ϵ) *be an algorithm for finding* (r, ϵ)-divisions *in planar graphs using* $T_D(n)$ *time and* $P_D(n)$ *processors. Given a set* F *of* n *triangular faces from a planar graph, two and three applications of the algorithm* ϵ-Divide, *with* $x = x_2 = \frac{7}{3}$ *and* $x = x_3 = \frac{5}{3}$ *respectively, divides* F *into* $\Theta(n/s)$ *regions with* $O(\frac{n}{s^{1-\epsilon''}})$ *boundary edges where* $\epsilon'' = \dfrac{\frac{1}{2} + x\epsilon(M-1)}{1 + x(M-1)}$. *These iterations of* ϵ-Divide *use* $O(\log^3 n + T_D(n))$ *time and* $\max(n, P_D(n))$ *processors.*

Proof. By Lemma 1 and the discussions above, only two or three applications of ϵ-

Divide are necessary in order to correctly produce an $(n^{1-frac1M}, \epsilon'')$-division of F. See also the algorithm *Well-Formed_Separator* and its proof of correctness in [20].

The time complexity depends on the number of times Miller's algorithm is applied, its complexity, and the time for bookkeeping between calls to Miller's algorithm. As in [20] and [5], there are $O(\log n)$ levels of recursion in each call to ϵ-*Divide*. At each level, Miller's algorithm uses $O(\log^2 n)$ time [15, 10]. The bookkeeping for each recursive level is generally $O(\log n)$ time, and at most $O(\log^2 n)$ time. This includes routines for treefix computations, arbitrary spanning trees, and parallel prefix [16, 21, 12, 3]. With $O(\log n)$ iterations, $O(\log^3 n)$ time used is overall. Since each application of Miller's algorithm is on a (sub)graph of x vertices with at most $O(x^{frac1M})$ faces, each application uses only $O(x)$ processors [15, 8]. Therefore, the simultaneous applications of Miller's algorithm never require more than $O(n)$ processors. Since all of the other subalgorithms use at most $O(n)$ processors, only n processors are needed overall. ■

The above theorem is significant because in the corresponding theorem by Shannon and Wan [20], the time complexity is $O(\log^2 n * T_D(n))$. Therefore, in their iterative approach to improving the separator's size, the time complexity grows exponentially with $\log n$ as the base. In the above theorem, the two time terms are added instead of multiplied together. Therefore, the time complexity grows exponentially with only a constant as the base. With $O(\log \log n)$ necessary for convergence to a $O(\sqrt{n})$ size separator, this new theorem implies a *polylog* time complexity. In the next section, we show to adjust the boundary sizes (as in [20] and [5]) in order to produce an (r, ϵ'')-division from an (r, ϵ)-*division*.

3 Subdivisions with Sublinear Region Boundaries

In Theorem 1 from the previous section, we showed how to subdivide a collection of n faces from a planar graph into $\Theta(n/r)$ with a total of $O(\frac{n}{r^{1-\epsilon''}})$ boundary vertices (for $r = n^{1-\frac{1}{M}}$).

The challenge in this section is to guarantee that each region in the subdivision has a boundary of size $O(r^{\epsilon''})$ by subdividing regions with large boundaries as necessary. Once we have the boundaries bounded as above, then we have created an (r, ϵ'')-division.

As in the previous section, this second phase of finding an (r, ϵ'')-division will rely on a constant number (one) of (r, ϵ)-*divisions* of each region with more than $cr^{\epsilon''}$ boundary vertices for a constant $c \geq 1$. Let R be a region of size $r = n^{1-\frac{1}{M}}$ with $B > cr^{\epsilon''}$ boundary vertices. An $(s = r^{1-\frac{1}{M}}, \epsilon)$ division D_R of R implies subregions of size $O(n^{(1-\frac{1}{M})^2})$. D_R is computed independent of which faces are adjacent to R's boundary vertices. Now proceed using Miller's algorithm to find separators which subdivide D_R into new regions with at most $O(r^{\epsilon''})$ boundary vertices. The weight of each region in D_R is initially equal to the number of R's adjacent boundary vertices. As in [20] and [5], when each separator is found, the weights of the regions adjacent to it are appropriately increased. The algorithm for this second phase is called *Boundary-ε-Divide.*

Theorem 2 *Let* DIV(r, ϵ) *be an algorithm for finding* (r, ϵ)-divisions *in planar graphs with n faces using $T_D(n)$ time and $P_D(n)$ processors. Given a set F of n triangular faces from a planar graph that have been subdivided into $O(n/r)$ regions of size $r = n^{1-\frac{1}{M}}$ with $O(\frac{n}{r^{1-\epsilon''}})$ boundary vertices, the algorithm* Boundary-ε-Divide *finds an (r, ϵ'')-division of F where $\epsilon'' = \dfrac{\frac{1}{2} + x\epsilon(M-1)}{1 + x(M-1)}$. The algorithm uses $O(\log^3 n + T_D(n))$ time and* $\max(n, T_D(n))$ *processors.*

Proof. The main questions are how large the separators can be, what is maximum number of times a region R is subdivided, and what is the total number of new boundary vertices produced.

For region R, $|R|$ is $O(r = n^{1-\frac{1}{M}})$. The $(s = r^{1-\frac{1}{M}}, \epsilon)$-division R_D of R has regions with boundaries of size $O((s^{1-\frac{1}{M}})^{\epsilon})$. There are $O(s^{\frac{1}{M}})$ regions or superfaces in R_D. Therefore, applying Miller's algorithm to R_D produces a separator of size $O(r^{\epsilon'})$ where $\epsilon' = \frac{1}{2M} + \epsilon(1 - \frac{1}{M})$. As *Boundary-ε-Divide* subdivides R, all of the separators generated contain $O(r^{ep'})$ vertices since the number of faces and boundary vertices that Miller's algorithm considers never increases.

Following the discussion in [20] and [5], assume R initially has i boundary vertices. The recurrence below for $R(i)$ characterizes how many new regions can be generated in guaranteeing that R is subdivided into regions with at most $O(r^{\epsilon''})$ vertices.

$$\begin{aligned} R(i) &= 0, & & i \leq 9cr^{\epsilon''} \\ R(i) &= 1 + R(\alpha f + cr^{\epsilon'}) + R((1-\alpha)f + cr^{\epsilon'}), & & \text{otherwise, } \frac{1}{3} \leq a \leq \frac{1}{2} \end{aligned}$$

A straightfoward proof by induction shows that $R(i)$ is bounded above by $\frac{i-2cr^{\epsilon''}}{cr^{\epsilon''}} - 1$ since $\epsilon' < \epsilon''$.

Now we can characterize the number of new boundary vertices generated by *Boundary-ϵ-Divide*. Let t_i be the number of regions in the initial subdivision of F with i boundary vertices. By Theorem 1, these initial subdivisions have $\Sigma_i(it_i) \leq O(\frac{n}{r^{1-\epsilon''}})$ boundary vertices. Since i boundary vertices can induce $O(ir^{\epsilon''})$ new regions with $O(r^{\epsilon''})$ size boundaries each, the total number of new regions is $\Sigma_i\left(O(\frac{i}{r^{\epsilon''}})\, t_i\right) \leq O(n/r)$ and the total number of new boundary vertices is $O(n/r) * r^{\epsilon''} = O(\frac{n}{r^{1-\epsilon''}})$. Therefore, *Boundary-$\epsilon$-Divide* generates an (r, ϵ'')-division of F from the initial subdivision.

The time complexity is the same as in Theorem 1. $O(\log n)$ levels of recursion for applying Miller's $O(\log^2 n)$ time algorithm implies that *Boundary-ϵ-Divide* uses at least $O(\log^3 n)$ time. Since there are at most $r^{\frac{1}{M}}$ superfaces induced by the (r, ϵ)-*division* of R, Miller's algorithm uses only $O(r)$ processors for each initial region. Each level of recursion uses at most $O(\log n)$ time and n processors for bookkeeping as discussed in the proof of Theorem 1. ∎

The theorem above shows that the impact of reducing the sizes of the region boundaries produced by two or three applications of *ϵ-Divide* is only an additive term of $O(log^3 n + T_D(n))$ in the time complexity. This gives us the corollary below for finding an (r, ϵ'')-division of n triangular faces.

Corollary 1 *Let* DIV(r,ϵ) *be an algorithm for finding* (r,ϵ)-divisions *in planar graphs with* n *faces using* $T_D(n)$ *time and* $P_D(n)$ *processors for* $r = n^{1-\frac{1}{M}}$. *Given a set* F *of* n *triangular faces from a planar graph, the algorithms* ϵ-Divide *and* Boundary-ϵ-Divide *generate an* (r,ϵ'')-*division of* F *where* $\epsilon'' = \dfrac{\frac{1}{2} + x\epsilon(M-1)}{1+x(M-1)}$. *These two algorithms use* $O(\log^3 n + T_D(n))$ *time and* $\max(n, T_D(n))$ *processors.*

In the next section we demonstrate how these new techniques imply an NC algorithm for finding $O(\sqrt{n})$ cyclic separators in planar graphs.

4 NC Algorithms for Small Separators

In this section we show how to derive $O(\sqrt{n})$-size separators using only polylog time and n processors. We first review how a sequence of algorithms for finding smaller and smaller separator is implied by Corollary 1 in the previous section. As in [20], this converges to an algorithm for finding an $O(\sqrt{n})$-size separator. We then show how this final algorithm uses only polylog time — in [20] the final algorithm uses $O((\log n)^{O(\log\log n)})$ time.

We first observe that if we have an $(r = n^{1-\frac{1}{M}}, \frac{1}{2})$-division of a set F of n faces, then with one application of Miller's algorithm we have an $O(\sqrt{n})$ size cyclic separator of F ($b = \sqrt{r}$, $f = n/r$, separator has size $b\sqrt{f} = \sqrt{n}$). It uses $O(\log^2 n)$ time and only n processors. Therefore, we need now only to derive an algorithm for building $(r, \frac{1}{2})$-divisions.

Corollary 1 gives us the mechanism to develop divisions of a region F of faces that successively smaller boundaries. Unfortunately, the constant factors on the number of boundary vertices are not the same for the (r,ϵ)-division and the (r,ϵ'')-division in that corollary. (See [20] for more discussion on the problem with the constants.) For now, we will assume the constants are equal; below we will fix this problem.

Now we are ready to generate a sequence of algorithms with produce successively better (r,ϵ)-*divisions*. The basis case is $\epsilon = e(0) = 1$. By the discussions in [20], [23], and indirectly from [19], we can partition a collection of n faces into $n^{\frac{1}{M}}$ regions, each of size $O(r)$. This gives us an algorithm for finding $(r^{1-\frac{1}{M}}, 1)$-divisions using n processors and $O(log^2 n)$ time. Corollary 1 shows that we can therefore find an (r, ϵ)-*division* where $\epsilon = e(1) = \frac{\frac{1}{2}+x\cdot e(0)\cdot(M-1)}{1+x(M-1)}$ for $x = \frac{7}{3}$ or $x = \frac{5}{3}$ depending on whether ϵ-*Divide* was applied two or three times, respectively. Let $\beta = x(M-1)$. In general, the ith algorithm implied by Corollary 1 generates an $(r, e(i))$-division where

$$\begin{aligned} e(0) &= 1 \\ e(i+1) &= \frac{\frac{1}{2} + e(i)\cdot\beta}{\beta+1}. \end{aligned}$$

Solving the recurrence, we have

$$e(i) = \frac{1}{2} + \frac{1}{2}\left(\frac{\beta}{\beta+1}\right)^i.$$

Now we can compensate for fact that the new division algorithm in Corollary 1 does not have the same constants in the bound on the number of vertices in each boundary. If we slightly increase the exponent in the new division's boundary bound, then the increase in the constant is canceled as long as the problem is large enough and the exponent ϵ is far enough away from $\frac{1}{2}$. (That is, if the exponent is increased by an additive term of γ, then $n^\gamma \geq C$ must hold where C is the bothersome constant factor that we are trying to cancel out.)

To effect this increase in the exponent in an orderly manner, we assume that the ith algorithm has an exponent on each region's bound size of $f(i)$ for the constant factor $\alpha > 1$.

$$f(i) = \frac{1}{2} + \frac{1}{2}\left(\frac{\alpha\cdot\beta}{\beta+1}\right)^i.$$

However, for how large of an i will this work? Consider algorithm $i+1$ based on an algorithm for finding $(r, f(i))$-divisions. Since the true, unadjusted exponent for algorithm is

$$\frac{\frac{1}{2} + f(i)\cdot\beta}{\beta+1},$$

the difference with $f(i+1)$ leaves a term of

$$f(i+1) - \frac{\frac{1}{2} + x \cdot f(i)(M-1)}{1 + x(M-1)} = \frac{1}{2}\left(\frac{\alpha \cdot \beta}{\beta+1}\right)^i \left(\frac{\beta(\alpha-1)}{\beta+1}\right)$$

to compensate for a constant factor C.

By setting n raised to the above compensation term to be greater than C and solving for $i = I$, we know for I algorithms C can be effectively canceled with our trick.

$$\begin{aligned} n^{f(I+1) - \frac{\frac{1}{2} + x \cdot f(I)(M-1)}{1+x(M-1)}} &> C \\ I &\leq \frac{\log\log n - 1 - \log\left(\frac{\beta+1}{\beta(\alpha-1)}\right) - \log\log C}{\log\left(\frac{\beta+1}{\beta(\alpha-1)}\right)} \end{aligned}$$

The bound on I represents the limit of our approach's usefulness. The key question then is: "How close $\sqrt{n}$ is to $n^{f(I)}$?" Since the ratio of $n^{f(I)}$ and $\sqrt{n}$ as n increases is

$$\frac{n^{f(I)}}{\sqrt{n}} = C^{\log\left(\frac{\beta+1}{\beta(\alpha-1)}\right)},$$

$n^{f(I)}$ is $O(\sqrt{n})$. If $\alpha = 1.1$, the implied additional constant factor is $O(C^4)$ when $x \geq \frac{3}{2}$. Therefore, the Ith algorithm produces an $(r, \frac{1}{2})$-division.

Now we need to characterize the time complexity. Let $T_i(n)$ be the time complexity of the ith algorithm. Based on Corollary 1 and the discussion above, we see that for constants c_1, c_2, and c_3,

$$\begin{aligned} T_0(n) &\leq c_1 \log^3 n \\ T_{i+1} &= c_2 \log^3 n + c_3 T_i(n). \end{aligned}$$

Bounding this recurrence with constant $c = \max(c_1, c_2, c_3)$, we have

$$\begin{aligned} T_i &\leq \left(\sum_{j=1}^{i+1} c^j\right) \log^3 n \\ T_i &\leq c^{i+2} \log^3 n \end{aligned}$$

For the Ith algorithm, $I+2$ is $O(\log\log n)$. Therefore, for constant $d \geq 1$,

$$\begin{aligned} T_I &\leq c^{d \log\log n} \log^3 n \\ T_I &\leq (\log n)^{d \log c} \log^3 n \end{aligned}$$

The above equation shows that the Ith algorithm uses only polylog time. With the comments at the beginning of this section, this immediately implies a linear-processor DNC algorithm for finding $O(\sqrt{n})$-size cyclic separators in planar graphs. This result and the algorithms leading up to the Ith one are characterized in the theorem and corollary below.

Theorem 3 *Given a set F of n triangular faces from a planar graph, the algorithm implied by the ith application of Corollary 1, produces an $(r = n^{1-\frac{1}{M}}, f(i))$-division of F and uses $O(c^i \log^3 n)$ time, for a constant $c \geq 1$, and n processors if i is $O(\log\log n)$.*

Corollary 2 *Given a set F of n triangular faces from a planar graph, the algorithm implied by the Ith application of Corollary 1 (where I is defined above), produces an $(r = n^{1-\frac{1}{M}}, \sqrt{n})$-division of F and uses $O(\log^{d+3} n)$ time, for a constant $d \geq 1$, and n processors if $I = c \log\log n$ for some constant $c > 0$.*

5 Conclusion

We have presented linear-processor DNC algorithms for finding sublinear-size cyclic separators in planar graphs. Improved divisions are found more efficiently by carefully using old divisions of the graph. This avoids the exponential computations used by Shannon and Wan in [20] and culminates in the first linear-processor DNC algorithm for $O(\sqrt{n})$-size cyclic separators.

There are two drawbacks to the final algorithm (the Ith algorithm at the end of Section 4). One is that the constant on the separator's size is not trivial. However, it should be less than 100 and can likely forced much lower by improved analysis and slight adjustments to the algorithms. The other drawback is that the time complexity is only known to be polylog. The exponent on the time could easily be 10 or even 15. Again, we suspect that a much more reasonable time complexity is possible with careful analysis and slight adjustments to the algorithms.

Acknowledgements

The author would like to thank Fang Wan for fruitful discussions on this work and Angie Allen for proofreading the draft.

References

[1] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] N. Alon, P. Seymour, and R. Thomas. A separator theorem for graphs with an excluded minor and its applications. In *Proceedings of the 22st Annual ACM Symposium on Theory of Computing*, pages 293–299, 1990.

[3] R. Cole and U. Vishkin. Optimal parallel algorithms for expression tree evaluation and list ranking. In *Proceedings of the 3rd Agean Workshop on Computing: VLSI Algorithms and Architectures*, pages ???–???, 1988.

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 1–6, 1987. NC integer matrix multiplication using $n^{2.376}$ processors.

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

[6] G. Frederickson and R. Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18:843–857, 1989.

[7] G. Frederickson and R. Janardan. Space-efficient message routing in c-decomposable networks. *SIAM Journal on Computing*, 19, 1990. To appear.

[8] H. Gazit. *Processor Efficient Parallel Graph Algorithms.* PhD thesis, University of Southern California, Los Angeles, CA, August 1988.

[9] H. Gazit and G. Miller. A parallel algorithm for finding a separator in planar graphs. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 238–251, 1987.

[10] H. Gazit and G. Miller. An improved parallel algorithm that computes the BFS numbering of a planar graph. *Information Processing Letters*, 28(2):61–65, June 1989.

[11] R. Karp and V. Ramachandran. A survey of parallel algorithms for shared-memory machines. Technical Report UCB/CSD 88/408, Computer Science Division (EECS), University of California, Berkeley, CA, March 1988. To appear in the *Handbook of Theoretical Computer Science* by North-Holland.

[12] P. Klein and J. Reif. An efficient parallel algorithm for planarity. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 465–477, 1986.

[13] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Algebraic and Discrete Methods*, 36(2):177–189, April 1979.

[14] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, August 1980.

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

[16] G. Miller and J. Reif. Parallel tree contractions and its applications. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 478–489, 1985.

[17] V. Pan and J. Reif. Fast and efficient solution of path algebra problems. *Journal of Computer and System Sciences*, 38:494–510, 1989.

[18] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[19] G. Shannon. A linear processor algorithm for depth-first search in planar graphs. *Information Processing Letters*, 29(3):119–124, October 1988.

[20] G. Shannon and F. Wan. Approximating small separators in planar graphs: Trade-offs in time, processors, and separator size. Technical Report 309, Department of Computer Science, Indiana University, Bloomington, Indiana 47405, June 1990.

[21] Y. Shiloach and U. Vishkin. An $O(\log n)$ parallel connectivity algorithm. *Journal of Algorithms*, 3:57–67, 1982.

[22] R. Tarjan. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, 1983.

[23] F. Wan. A linear-processor algorithm for finding small cycle separators on undirected planar graphs. Technical Report 308, Department of Computer Science, Indiana University, Bloomington, Indiana 47405, April 1990.