

Expected Parallel Time and Sequential Space Complexity of Graph and Digraph Problems

John Reif¹ and Paul Spirakis²

Abstract. This paper determines upper bounds on the expected time complexity for a variety of parallel algorithms for undirected and directed random graph problems. For connectivity, biconnectivity, transitive closure, minimum spanning trees, and all pairs minimum cost paths, we prove the expected time to be $O(\log \log n)$ for the CRCW PRAM (this parallel RAM machine allows resolution of write conflicts) and $O(\log n \cdot \log \log n)$ for the CREW PRAM (which allows simultaneous reads but not simultaneous writes). We also show that the problem of graph isomorphism has expected parallel time $O(\log \log n)$ for the CRCW PRAM and $O(\log n)$ for the CREW PRAM. Most of these results follow because of upper bounds on the mean depth of a graph, derived in this paper, for more general graphs than was known before.

For undirected connectivity especially, we present a new probabilistic algorithm which runs on a randomized input and has an expected running time of $O(\log \log n)$ on the CRCW PRAM, with $O(n)$ expected number of processors only.

Our results also improve known upper bounds on the expected space required for sequential graph algorithms. For example, we show that the problems of finding connected components, transitive closure, minimum spanning trees, and minimum cost paths have expected sequential space $O(\log n \cdot \log \log n)$ on a deterministic Turing Machine. We use a simulation of the CRCW PRAM to get these expected sequential space bounds.

Key Words. Graph algorithms, Parallel, Average case, Random graphs, Complexity.

1. Introduction

1.1. The Parallel Machine Models. We consider here some fundamental models of parallel computation, all of which assume the presence of an unlimited number of processors. In the first model, the CREW (Concurrent Read Exclusive Write) PRAM (see [Wyllie, 1979] and [Fortune and Wyllie, 1978]), different processors can read the same memory location at the same time. They may store information at different memory locations simultaneously, but no two processors can attempt to change the contents of the same memory cell on the same step. Reif [1982a,b] proposed a probabilistic PRAM where processors are capable of doing in-

¹ Department of Computer Science, Duke University, Durham, NC 27706, USA. This research was supported by National Science Foundation Grant DCR-85-03251 and Office of Naval Research Contract N00014-80-C-0647.

² Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece, and Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA. This research was partially supported by the National Science Foundation Grants MCS-83-00630, DCR-8503497, by the Greek Ministry of Research and Technology, and by the ESPRIT Basic Research Actions Project ALCOM.

dependent probabilistic choices on a fixed input (but, again, simultaneous writing at the same location is not allowed).

We also consider the CRCW (Concurrent Read Concurrent Write) PRAM where simultaneous access to the same memory location is allowed for both read and write operations. In the case of a simultaneous write attempt, exactly one processor succeeds but we make no assumption of which one succeeds. We denote this machine model by WRAM for ease of notation. (See [Shiloah and Vishkin, 1981] and [Goldschlager, 1978] for similar models).

A time bound for a problem in the WRAM can be translated into a space bound for the same problem for a deterministic Turing Machine by the following lemma:

LEMMA 1.1. *Let L be a language accepted by a $T(n)$ time-bounded WRAM, with a polynomial number of processors. Then L is accepted by an*

$$O(T(n) \cdot \max\{T(n), \log n\})$$

space-bounded deterministic Turing machine.

For a proof, see [Goldschlager, 1978], [Cook, 1980], and [Ruzzo, 1982]. The result has been proved for the SIMDAG machine which is more powerful than the WRAM. Also, we have

PROPOSITION 1.1. *Any parallel algorithm for the WRAM of time $T(n)$ implies a parallel algorithm on the CREW PRAM model of time $O(T(n) \cdot \log m)$, where m is the total number of processors per step.*

PROOF. We can simulate the nondeterministic choice of write conflict resolutions by a tree of depth $\log m$ of pairwise selections to find a unique winner. \square

1.2. Known Parallel Graph and Digraph Algorithms and Open Problems. We list here previous results. The best-known deterministic algorithm for undirected graph connectivity is of $\Theta(\log^2 n)$ time in the CREW PRAM model, with $n^2/\log^2 n$ processors [Chin *et al.*, 1982]. Hirschberg *et al.* [1979] presented a version of the above algorithm which used $\Theta(n^2/\log n)$ processors.

DEFINITION. Let $G = (V, E)$ be a (directed or undirected) graph and let $d(v, \omega)$, for $v, \omega \in V$, be the length of the shortest path from v to ω if any, $-\infty$ otherwise. Then the depth, $d(G)$, of the graph G is defined to be the $\max_{v, \omega \in V} \{d(v, \omega), 1\}$.

Note that for undirected graphs G with nontrivial connected components, $d(G)$ is the maximum of the diameters of the connected components of G . Savage and Ja'Ja' [1981] remarked that the Hirschberg *et al.* [1979] algorithm is actually of time $\Theta(\log n \cdot \min\{\log n, d\})$, where d is the depth of the graph.

The best-known deterministic algorithm for undirected connectivity in the WRAM is due to Shiloah and Vishkin [1981] and is of $\Theta(\log n)$ time, with

$\Theta(n + m)$ processors, where m is the number of the edges of the graph. Shiloah and Vishkin [1981] has the ingenious idea of using distributed collapsing of the height of the data structures which are candidates for connected components. Their algorithm is essentially different from that of Hirschberg, *et al.* [1979] in the sense that the algorithm of Hirschberg *et al.* [1979] cannot be adjusted to run in time $\Theta(\log n)$ in the stronger model WRAM.

Cole and Vishkin [1986] presented an optimal (with respect to time-processors tradeoff) algorithm for undirected connectivity, for the WRAM model, which, however, has a running time of $O(\log n \cdot \log \log n \cdot \log \log \log n)$. Also, Gazit [1986] presented a *randomized* parallel algorithm for finding the connected components of an undirected graph, with $O(\log n)$ time and $O((m + n)/\log n)$ processors.

Also, Ja'Ja' [1978] has a $\Theta(\log n \log d)$ time and $n^3/\log n$ processors algorithm for the CREW PRAM, where d is the depth of the graph. For the biconnected components, Ja'Ja' and Savage [1981] give a $\Theta(\log^2 n)$ -time algorithm in deterministic CREW PRAM with $n^3/\log n$ processors and also a $\Theta(\log n \cdot \log d \cdot \log k)$ -time algorithm with $mn + n^2 \log n$ processors, where k is the number of components and m is the number of edges of the graph. Also, for the problem of finding the biconnected components of a graph, Tsin and Chin [1983] gave a $\Theta(\log^2 n)$ -time algorithm, with $n^2/\log^2 n$ processors, in the CREW PRAM model. For the minimum spanning tree problem, Ja'Ja' and Savage [1981] give a $\Theta(\log^2 n)$ -time algorithm with n^2 processors. For the same problem, Tsin and Chin [1983] also give a $\Theta(\log^2 n)$ -time algorithm in the CREW PRAM model with $n^2/\log^2 n$ processors. For all the above undirected graph problems, Reif [1982a] gives a $\Theta(\log n)$ -time probabilistic algorithm for the CREW PRAM, requiring $n^{O(1)}$ processors. Note that the results of Reif [1982a] are not applicable to digraph problems.

For transitive closure and strong components in digraphs, the best result [Ja'Ja', 1978] for the CREW PRAM is of $\Theta(\log n \cdot \log d)$ time with $n^3/\log n$ processors. For the all pairs shortest paths in digraphs, Dekel, *et al.* [1981] give a $\Theta(\log^2 n)$ algorithm with n^3 processors. No parallel algorithm for isomorphism was known up to now.

The above discussion of previous work indicates several basic unanswered questions:

- (a) Can parallel time be dropped below $\Omega(\log^2 n)$ for directed connectivity problems on the CREW PRAM?
- (b) Can parallel time be dropped below $\log n$ for connectivity, biconnectivity, minimum spanning tree, etc., on the WRAM model?
- (c) Can parallel time for undirected connectivity be dropped below $\Omega(\log^2 n)$ and, simultaneously, can the number of processors be dropped below $\Omega(n^2/\log^2 n)$ on the CREW PRAM?

In fact, Shiloah and Vishkin [1983] conjecture that the barrier of $\log n$ cannot be surpassed by any polynomial number of processors.

This paper gives affirmative answers to the above questions if the words "parallel time" and "number of processors" are preceded by the word "expected," over inputs which are randomly chosen.

Furthermore, it is the first time that an algorithm for connectivity is presented with $O(\log \log n)$ expected parallel time on the WRAM.

1.3. Known Sequential Space Bounds for Graph Problems. A space-efficient sequential probabilistic search technique was first introduced by Aleliunas *et al.* [1979] to test connectivity. Lewis and Papadimitriou [1982] observe that this technique immediately implied a space $O(S(n))$ probabilistic algorithm to simulate a symmetric nondeterministic machine with space $S(n)$. Reif [1982a] extended the above techniques to yield $O(\log n)$ probabilistic sequential space algorithms for many undirected graph problems. However, the above do not extend to directed graph problems, for which the problem of achieving less than $\Omega(\log^2 n)$ sequential space is still open. (The $\Omega(\log^2 n)$ space bound is implied by the theorem of Savitch [1970]).

1.4. Average Performance of Graph Algorithms and Random Graphs. Considerable work has been done on sequential graph algorithms which are fast on the average given random input graphs. This includes the work by Angluin and Valiant [1979], Karp [1976], Karp and Sipser [1981], Schnorr [1978], Karp, and Tarjan [1980], Reif and Spirakis [1981], and Spirakis [1981].

Almost no previous work examined the average performance of parallel graph and digraph algorithms, with the exception of Shamir and Upfal [1982] on coloring random graphs, Karp and Sipser [1981] on parallel matchings in random graphs, and Coppersmith *et al.* [1987] on coloring, finding Hamilton circuits, and finding lexicographically first maximal independent sets on random graphs. Furthermore, no previous work analyzes the expected sequential space of graph algorithms.

We analyze here the average performance of parallel algorithms for connectivity, biconnected components, strong connectivity and transitive closure, minimum cost spanning tree, minimum cost all pair shortest paths, and graph isomorphism. Our analysis also gives upper bounds on the expected sequential space complexity of the above problems.

The input to our algorithms is assumed to be a random graph of the model $G_{n,p}$ as defined by Erdos and Renyi [1960], where n is the number of vertices and p is the edge probability, or a random digraph $D_{n,p}$ as defined by Angluin and Valiant [1979].

In the Appendix we investigate the probability distribution of the depth of a random graph $G_{n,p}$ or a random digraph $D_{n,p}$ (the depth was defined in Section 1.2).

The mean value d (and bounds on the probability distribution) of the depth is crucial to a number of our results. We prove the following theorems:

THEOREM 1.1. *There is a constant $c_0 > 2$ such that, for any probability p in the range*

$$[0, 1] - \left[\frac{c_0}{n}, \frac{2c_0}{n} - \left(\frac{c_0}{n} \right)^2 \right],$$

the graph $G_{n,p}$ has average depth $d = O(\log n)$. Furthermore, $\text{prob}\{d = O(\log n)\}$ tends to 1 as n tends to ∞ .

THEOREM 1.2. *There is a constant $c_0 > 2$ such that, for any probability p in the range*

$$[0, 1] - \left[1 - \sqrt{1 - \frac{c_0}{n}, \frac{c_0}{n}} \right],$$

the digraph $D_{n,p}$ has average depth $d = O(\log n)$. Furthermore, $\text{prob}\{d = O(\log n)\}$ tends to 1 as n tends to ∞ .

We also prove that, for

$$p \geq \frac{c}{n^{1/3}}$$

(c a particular constant > 1), the depth d of $G_{n,p}$ (or $D_{n,p}$) is less than or equal to 3 with probability $\rightarrow 1$ as $n \rightarrow \infty$.

The result $d = O(\log n)$ for nonsparse random graphs (with $p = (c/n) \cdot \omega(n)$ and $\omega(n) \rightarrow \infty$ as $n \rightarrow \infty$) can be deduced by the results of Erdos and Renyi [1960] (see the Appendix). Also, Bollobas [1984] has proved that when

$$p \geq \frac{c \log n}{n}$$

(for a suitable c such that the random graph becomes connected with high probability), then

$$\bar{d} \leq \left\lceil \frac{\log n + 6}{\log \log n} \right\rceil + 3.$$

For even denser graphs, with

$$p \geq \frac{d(n)}{n}$$

such that $d(n) - \log n \rightarrow \infty$, Bollobas [1984] proved that

$$\bar{d} \leq \left\lceil \frac{\log n + 6}{\log d(n)} \right\rceil + 3.$$

In contrast, the results for sparse graphs with edge probability $p = \Theta(1/n)$ and for very dense graphs (with $p \geq c/n^{1/3}$) are totally new contributions of our paper.

1.5. Our Expected Time Results for Parallel Graph Problems. Using the above facts and some nontrivial analysis, we show that the connected components of a random graph $G_{n,p}$ and the strong components and transitive closure of a random

digraph $D_{n,p}$ and also the minimum spanning tree of a weighted random graph and the all-pairs shortest paths of a weighted random graph or digraph can be found in average parallel time $O(\log \log n)$ in the WRAM and $O(\log n \cdot \log \log n)$ in the CREW PRAM. The average time for biconnected components is

$$O((\log \log n)^2)$$

for the WRAM and $O(\log n \cdot (\log \log n)^2)$ for the CREW PRAM. The average time for graph isomorphism is $O(\log \log n)$ in the WRAM and $O(\log n)$ in the CREW PRAM. The results for digraph connectivity and digraph and graph all-pairs shortest paths are implied by the result that the average depth of $G_{n,p}$ or $D_{n,p}$ is $d = O(\log n)$ (see Section 1.4).

However, our result for undirected connectivity in the WRAM (and the result for minimum spanning tree in the WRAM) are not implied by the fact that $d = O(\log n)$. A large part of our paper is devoted to a new algorithm (and probabilistic analysis) for undirected connectivity in the WRAM. A side benefit of this new algorithm is a result about the expected number of processors needed for undirected connectivity in the CREW PRAM model, which is

$$n^2/(\log n \cdot \log \log n)$$

for time $O(\log n \cdot \log \log n)$, and $O(n)$ in the WRAM model. A random sampling technique is used to derive this result.

The above results answer the (open) questions (a)–(c) of Section 1.2 in an affirmative way, given that we concentrate on average-case analysis.

For directed connectivity and transitive closure, and biconnected components we need $n^3/\log n$ processors for the CREW PRAM model and n^3 processors for the WRAM model. For graph isomorphism we need n^2 processors for the WRAM and $n^2/\log n$ for the CREW PRAM. For the all pairs shortest paths we need $n^3/\log n$ processors in the CREW PRAM model and n^3 processors in the WRAM model. For the minimum weight spanning tree we use an expected number of n^3/p processors, which is $\Theta(n^2)$ for $pn = \Theta(1)$.

We remark that there is generally a considerable drop in time and number of processors complexity when we concentrate on the average case of the above problems rather than worst-case input. Note that none of the previously known algorithms for undirected connectivity imply an expected time below $\Omega(\log^2 n)$ in the CREW PRAM and below $\log n$ in WRAM and a simultaneous drop of expected number of processors below $n^2/\log n$ even if the average depth result $d = O(\log n)$ is used. Note also that our results for the WRAM model can be applied to the SIMDAG model of Goldschlager [1978] and other parallel computation models allowing resolution of write conflicts in a constant number of steps (see [Kucera, 1981]).

1.6. Our Results for Expected Space Complexity. Our results improve known upper bounds on the expected space required for sequential graph algorithms. By using our expected parallel time results for WRAMs and Lemma 1.1 we provide

an upper bound of $O(\log n \cdot \log \log n)$ for the expected deterministic sequential space needed for undirected connectivity, directed (strong) connectivity and transitive closure, minimum cost spanning tree and directed and undirected all pairs shortest paths, and graph isomorphism. Reif, [1982b] provides $\Theta(\log n)$ probabilistic sequential space bounds for the undirected versions of all the above problems, but his method does not seem to extend to directed graph problems. Our results answer the question posed in Section 1.3 in an affirmative way when we concentrate on expected space and our methods could be viewed as indirect ways of getting results about expected sequential space.

2. Results that Follow from the Bounds on the Mean Depth of Random Graphs

2.1. Digraph Connectivity. The following well-known algorithm for computing the transitive closure of an $n \times n$ boolean matrix A can be used for digraph connectivity and transitive closure:

```

input  $A$ 
begin
     $B \leftarrow A + I$ 
 $L: B' \leftarrow B$ 
     $B \leftarrow B \cdot B$ 
    if  $B' \neq B$  then goto  $L$  else output  $B$ 
end

```

This algorithm was shown by Dekel *et al.* [1981] to be of $\Theta(\log^2 n)$ worst-case time by using n^3 processors in the CREW PRAM model. Ja'Ja' [1978] did an efficient implementation of this algorithm showing that it could be done in $\Theta(\log n \cdot \log d)$ time on the CREW PRAM model and by using $n^3/\log n$ processors, where d is the depth of the digraph D . The n^3 factor in the number of processors in [Ja'Ja', 1978] is due to multiplication of $n \times n$ matrices. Parallelization of the faster matrix multiplication algorithms known today, will lead to a time $\Theta(\log n \cdot \log d)$ on the CREW PRAM model, using only $t(n)/\log n$ processors, where $t(n)$ is the sequential time for matrix multiplication.

It is easy to observe, that the primitive set of operations of Ja'Ja' [1978] (i.e., the union of $O(n)$ sets of $O(n)$ integers each, from 1 to n) can be done in $O(1)$ time in the WRAM, implying an $O(\log d)$ algorithm for strong components and transitive closure (see also [Kucera, 1982]). Using our results on the average depth of $D_{n,p}$ (see the Appendix) we get

COROLLARY 2.1. *The average parallel time for transitive closure of directed graphs is $O(\log \log n)$ for the WRAM model and $O(\log n \cdot \log \log n)$ for the CREW PRAM model. The probability that the parallel time is more than this is less than $n^{-\gamma}$, $\gamma > 2$. The WRAM model uses n^3 processors and the CREW PRAM model uses $n^3/\log n$ processors for the stated time bounds.*

Ja'Ja' [1978] also gave an algorithm for finding the strong components of a digraph in parallel time $\Theta(\log n \cdot \log d)$ in the CREW PRAM model, by using $n^3/\log n$ processors. Our results on the depth of $D_{n,p}$ also imply

COROLLARY 2.2. *The average parallel time for finding the strong components of a digraph in the CREW PRAM model is $O(\log n \cdot \log \log n)$, using $n^3/\log n$ processors. In the WRAM model the average time is $O(\log \log n)$ using n^3 processors. The probability that the parallel time is more than this, is less than $n^{-\gamma}$ for some $\gamma > 2$.*

By using Lemma 1.1 we also get

COROLLARY 2.3. *The average deterministic sequential space needed for transitive closure and strong components of digraphs and connected components of undirected graphs is $O(\log n \cdot \log \log n)$. The probability that the space needed is*

$$O(\log n \cdot \log \log n)$$

is at least $1 - n^{-\gamma}$ for some $\gamma > 2$.

Note that the same techniques can be applied to undirected graphs to compute the connected components by the CREW PRAM in $O(\log n \cdot \log \log n)$ average time. However, the number of processors used is wasteful. In Section 3 we give a derivation of this result which is independent of our results about the depth of $G_{n,p}$ and optimizes simultaneously over expected time and expected number of processors.

2.2. Biconnected Components. Savage and Ja'Ja' [1981] provided an algorithm for the CREW PRAM which finds the biconnected components of a graph in parallel time $O(\log n \cdot \log d \cdot \log k)$, where d is the depth of the graph and k is the number of biconnected components. The number of processors needed is $mn + n^2 \log n$. Again, this algorithm can be implemented on the WRAM model to run in $O(\log d \cdot \log k)$ parallel time with $mn + n^2 \log n$ processors. The Appendix shows the expected depth is $d = O(\log n)$. Reif and Spirakis [1985] show the expected number of biconnected components is $k = O(\log n)$. These results imply

COROLLARY 2.2.1. *The expected parallel time complexity for finding biconnected components in the random graph $G_{n,p}$ with $p \geq c/n$, $c > 5$, is $O((\log \log n)^2)$ for the WRAM model and $O(\log n \cdot (\log \log n)^2)$ for the CREW PRAM. The number of processors needed is $mn + n^2 \log n$ for the WRAM (also for the CREW PRAM). The probability that the parallel time is more than this is $o(1/n)$.*

By using Lemma 1.1 we get

COROLLARY 2.2.2. *The expected deterministic space for finding biconnected components in the random graph $G_{n,p}$ with $p \geq c/n$, $c > 5$, is $O(\log n \cdot (\log \log n)^2)$.*

2.3. All Pairs Shortest Paths. Given an n vertex graph G or digraph D , the all pairs shortest path matrix A is an $n \times n$ matrix such that $A(i, j)$ is the length of a shortest path from i to j in G (or D). Let $A^k(i, j)$ denote the length of a shortest path from i to j going through at most k intermediate vertices. Clearly, $A(i, j) = A^d(i, j)$ where $d = \text{depth of } G \text{ (or } D)$. Let $A^0(i, j)$ be 1 if $\{i, j\}$ is in G (if (i, j) is in D) and ∞ otherwise. It is easy to see that

$$A^k(i, j) = \min_m \{A^{k/2}(i, m) + A^{k/2}(m, j)\}.$$

Hence we can compute A^d by computing

$$A^2, A^4, \dots, A^{2^i}, \quad i = \lceil \log d \rceil.$$

A^k can be computed from $A^{k/2}$ using the matrix multiplication algorithm with $+$ substituted for $*$ and \min for $+$. (See also [Dekel et al. 1981].)

Assuming binary representation of lengths and by using bit addition with parallel carry, we can perform all additions in the k th phase of the algorithm in constant time on the WRAM, by assigning one processor to each pair of (i, m) , (m, j) . The min operation can obviously be done in constant time in the WRAM. To see this we have to use $O(n^2)$ processors as follows: Assign $n - 1$ processors to each m . Each one of them compares the computed $v_m = A^{k/2}(i, m) + A^{k/2}(m, j)$ with one of the $n - 1$ other values. If v_m is bigger, the corresponding processor writes a 1 in a memory location C_m otherwise it does not write. Next, one processor for each m checks C_m . If C_m is not written, then this processor outputs its v_m into $A^k(i, j)$. Only one C_m will not be written in this step. Hence,

THEOREM 2.3.1. *The all pairs shortest paths of a graph or digraph can be computed in $O(\log d)$ worst-case time on a WRAM, where d is the depth of the graph or digraph.*

Note that the above procedure is also valid if we are computing the all pairs minimum weight shortest paths provided that edge weights are nonnegative and satisfy the triangle inequality. The only modification needed is $A^0(i, j) = \text{the weight of the edge } (i, j), \text{ if it exists, } \infty \text{ otherwise.}$ This leads to an expected time $O(\log \log n)$ for the WRAM model for random graphs $G_{n,p}$ or digraphs $D_{n,p}$ since the expected d is $O(\log n)$ (see the Appendix).

COROLLARY 2.3.1. *The expected time for all pairs shortest paths in $G_{n,p}$ or $D_{n,p}$ models with p as in the Theorems 1.1 and 1.2 and arbitrary weights, nonnegative and satisfying the triangle inequality, is $O(\log \log n)$ for the WRAM model by using n^3 processors. It is $O(\log n \cdot \log \log n)$ in the CREW PRAM by using $n^3/\log n$ processors. The probability that the time is more than stated above is $\leq n^{-\gamma}$, $\gamma > 2$.*

By Lemma 1.1 we get

COROLLARY 2.4. *The expected deterministic sequential space for all pairs shortest paths in the random graphs $G_{n,p}$ or random digraphs $D_{n,p}$ (which are arbitrarily*

weighted with nonnegative weights satisfying the triangle inequality) is

$$O(\log n \cdot \log \log n).$$

2.4. Minimum Weight Spanning Trees

2.4.1. An Algorithm for Minimum Spanning Trees of Unidirected Graphs in the WRAM. Let G be a weighted undirected connected graph, with positive weights. The connectivity algorithm of Hirschberg *et al.* [1979] can be used to obtain a minimum weight spanning tree, by picking the edge with minimum weight while merging supervertices with one another (see [Savage and Ja'Ja', 1981]). Let G be an instance of the random graph $G_{n,p}$, $p \geq c/n$, as input, with arbitrary nonnegative edge weights. We can use our new algorithm, Algorithm MERGE, of Section 3 for connectivity by using the Ja'Ja' [1978] algorithm for the deterministic part. Picking the minimal weight edge will take constant time in the WRAM if we use $n^3 \cdot p$ processors. Hence,

COROLLARY 2.4.1. Suppose we are given an instance of the random graph $G_{n,p}$, $p \geq c/n$ (c is the constant defined in Section 3.1), as input, with arbitrary nonnegative edge weights. Then we can compute a minimum weight spanning tree in the WRAM in expected parallel time $O(\log \log n)$, using $n^3 p$ processors on the average.

COROLLARY 2.4.2. The expected deterministic sequential space for a minimum weight spanning tree is $O(\log n \cdot \log \log n)$, when the input graph is as in Corollary 2.4.1.

2.5. Isomorphism of Random Graphs. The following algorithm, given by Babai and Kucera [1979], is essentially a Breadth-First-Search procedure applied to random graphs of the model $G_{n,p}$ with $p = \frac{1}{2}$.

1. Classify vertex by valences so each vertex is labeled by its valence. Let C_1, \dots, C_h be the classes of the above classification, arranged by the order induced by this classification.
2. In the first refinement we let $N_i(v)$ be the number of neighbors of v in C_i and let $N_i^4(v)$ be the smallest nonnegative integer congruent to $N_i(v) \bmod 4$. Then two vertices v, u are ordered now if (a) they had a different label before or (b) they had same labels but

$$(N_1^4(v), \dots, N_h^4(v)) < (N_1^4(u), \dots, N_h^4(u))$$

lexicographically.

3. Let C_1, C_2, \dots, C_4 , be the equivalence classes of step 2. Apply step 2 to each of these classes.

THEOREM 2.5.1 [Babai and Kucera, 1979]. Let U be the set of vertices whose class is not a singleton after step 3. Then

$$\text{Prob}\{|U| \geq 2\} \leq e^{-2cn}$$

for some absolute constant $c > 0$.

The valence classification can be done in $O(\log \log n)$ time on the WRAM model by using $O(n^2)$ processors. In the CREW PRAM it can be done in time $O(\log n)$ by using $n^2/\log n$ processors. The partial orders created by the refinements are best stored as a function corresponding to them. These function computations cost no more than $O(\log \log n)$ time in the WRAM and $O(\log n)$ in the CREW PRAM, by using n^2 and $n^2/\log^2 n$ processors, respectively. So,

COROLLARY 2.5.2. *A canonical labeling algorithm on $G_{n,p}$, $p = \frac{1}{2}$, takes expected parallel time $O(\log \log n)$ in the WRAM model with n^2 processors and $O(\log n)$ time in the CREW PRAM with $n^2/\log n$ processors.*

By Lemma 1.1, we get

COROLLARY 2.5.3. *The expected sequential deterministic space needed for a canonical labeling algorithm on $G_{n,p}$, $p = \frac{1}{2}$, is $O(\log n \cdot \log \log n)$.*

3. Expected Parallel Time and Processor Bounds for Connectivity in Undirected Graphs

3.1. Introductory Remarks and Assumptions. We present here a new algorithm for finding connected components of a graph on a WRAM. Although it is related to the algorithm of Shiloah and Vishkin [1982], it uses probabilistic choice so the processors of the WRAM model are allowed to choose random bits independently. Also, we assume random resolution of write conflicts. If k processors attempt to write simultaneously at the same memory location, then exactly one succeeds and the probability that each particular one succeeds is $1/k$. Both of these assumptions can be removed by using the randomness of the input. Our new algorithm for graph connectivity has $O(\log \log n)$ expected probabilistic parallel time in the WRAM, which leads to an $O(\log n \cdot \log \log n)$ expected parallel time algorithm in a probabilistic CREW PRAM model.

Since our algorithm works on a random input (a graph of the $G_{n,p}$ model) we can use the randomness of the input in order to implement the probabilistic choices of the processors of the WRAM. In fact, our algorithm has three stages, out of which only one requires probabilistic choice. This particular stage “looks” at about one-third of the edges of the random graph (see details at the description of the algorithm). The remaining edges can be used in order to simulate the probabilistic choices of the processors of the WRAM. Hence, our results about the expected parallel time hold also for the deterministic WRAM model, which, however, allows random resolution of write conflicts.

Our algorithm uses $n + 2m$ processors for a graph of n vertices and m edges, hence it uses $n + 2pn^2$ processors on the average. An improvement over this is $n^2/(\log n \cdot \log \log n)$ processors and that is shown in Section 3.5 for the CREW PRAM. An additional improvement (shown in Section 3.6) brings the processor count down to $O(n)$ by using random sampling.

3.2. A Preview of the Algorithm and Its Analysis. Our algorithm uses the following data structure: Each vertex v has a pointer field $D(v)$ through which it

points on another vertex or to itself. We can regard $v \rightarrow D(v)$ as a (directed) edge in an auxiliary graph, called the pointers-graph. In the probabilistic part of our algorithm, this pointers-graph will always be a collection of sets called supervertices, each of which is a rooted star of height one, with a self-loop at the roots. As the algorithm proceeds, the number of sets decreases while each individual set increases or disappears. This is caused by a “hooking” operation in which a supervertex can be hooked on another supervertex.

The algorithm has three stages. In the first stage the algorithm proceeds as in [Shiloah and Vishkin, 1982], for a parallel time of $O(\log \log n)$ steps. Due to this, the algorithm constructs a sufficient number of supervertices of polylogarithmic size, at the end of the stage.

In the second stage, which goes through $O(\log \log n)$ phases, the algorithm becomes probabilistic. In each phase each of the supervertices decides probabilistically (depending on their size) to be either an acceptor or a proposer. Only proposers can hook themselves to other supervertices which must be acceptors. Let us vaguely speak here about the expected size of supervertices in phase i , and denote it by $s(i)$. Each supervertex decides to be an acceptor with a probability proportional to $1/s(i)$. This results in an expected number of approximately $s(i)$ proposers per acceptor, resulting in a quadratic growth of supervertices (i.e., $s(i+1) = s(i)^2$) per phase. This argument presumes that the size of the proposers and acceptors is at least $s(i)$. In our algorithm the supervertices have a fast probabilistic way to determine whether their size is at least $s(i)$. (This technique is similar to the logical OR and can be implemented in a constant number of parallel steps of the WRAM.) Only “large” supervertices become acceptors and proposers.

Due to the quadratic growth of components during the second stage, a parallel time of $O(\log \log n)$ steps results in the construction of $O(n)$ -size connected components.

The last stage is a clean-up stage. It is deterministic and follows the algorithm of Shiloah and Vishkin [1982].

Note that we do partial collapsing in each phase so that the supervertices remain rooted stars of height one. At the end, each remaining set is a connected component of the graph and is still a rooted star.

In the following, $s(t)$ intuitively corresponds to the size of supervertices at phase t of stage 2 of our algorithm.

DEFINITION. Let $s(t)$ be the following sequence:

$$s(1) = 2 \log^6 n$$

and, for $i \geq 1$,

$$s(i+1) = \frac{s(i)^2}{2^i(1+\varepsilon)\log^6 n}$$

(ε is a small constant, to be determined later). We now describe the algorithm in detail.

3.3. Algorithm MERGE. *Input:* Graph $G = (V, E)$, an instance of the random graph $G_{n,p}$. We use $D_t(v) = u$ to denote that an edge from vertex $v \in V$ points to vertex u after the t th phase.

Initially, at phase 1, for each $v \in V$, $\{v\}$ is a supervertex, these are the only supervertices, each supervertex is declared large and $D_t(v) = v$ for each vertex v .

In the following text we use "vertex" and the assigned processor to that vertex interchangeably. We allocate a vertex processor to each vertex. Each undirected edge $\{u, v\}$ is viewed as two ordered pairs $\langle u, v \rangle, \langle v, u \rangle \in V \times V$ and we allocate one edge processor to each such pair.

The algorithm has three stages. Before the start of the first stage, each edge processor draws a random number between 0 and 1, uniformly. We require that $\langle u, v \rangle$ and $\langle v, u \rangle$ draw the same number, but otherwise the choices are independent. Those processors whose number is less than $\frac{1}{3}$ participate in the first stage only. Those whose number is between $\frac{1}{3}$ and $\frac{2}{3}$ participate in the second stage only. The rest participate in the third stage. This procedure results in partitioning the instance of the random graph $G_{n,p}$ into three instances of a random graph $G_{n,p/3}$, in a way guaranteeing that edges used in one stage are not reused in another stage.

Stage 1. We apply the Shiloah and Vishkin [1982] connectivity algorithm for $2 + 6 \log \log n$ of its phases.

Stage 2. It consists of $\leq d_0 \log \log n$ phases. (d_0 is a positive constant, its exact value is determined later.) For the pairs of processors associated with each edge and with stage 2, flip a coin at the beginning of each phase of stage 2, with the probability of heads equal to $1/2$, to decide whether the edge will be used in that phase. Those who succeed are used only for that phase. Those unused, try again in the next phase. (This sampling of edges guarantees that an edge is not used in two different phases of stage 2 and hence the randomness of the input is not destroyed.)

Set $i = 1$.

Execute the following loop until $i = d_0 \log \log n$.

Phase i of stage 2

- (a) The purpose of this step is to estimate quickly the size of each supervertex. Let β be a positive constant greater than 3, to be determined later. For each supervertex, construct a table of size $s(i)/(\beta \log n)$. Initially, all entries in the table are 0. Each vertex of the supervertex sets a randomly chosen cell of the table to 1 on this phase. We declare a supervertex to be large if all its cells are found to be 1.
- (b) Let k be a constant greater than 3, whose exact value is determined later. Let the root of each large supervertex decide to be an acceptor with probability $\log^k n / (\alpha \cdot s(i))$ (α is a small positive constant > 1 , to be determined later). All other supervertices are declared proposers.
- (c) Each proposer supervertex chooses a random edge departing from it to an acceptor (if such an edge exists) and merges into that acceptor. This is done in constant time on the WRAM as follows: All the processors associated with

vertices of the proposer check whether their neighbors are nodes of an acceptor. If one finds such a neighbor u , it tries to hook its supervertex on $D_{i-1}(u)$. If more than one try this, then each one is allowed to succeed with equal probability by use of random resolution of simultaneous writing.

(d) (Collapsing): For all vertices v we assign $D_i(v) = D_{i-1}(D_{i-1}(v))$.

Notice that phase i takes a constant number of steps on a WRAM. To do (a) in constant time, we must place the assigned table in contiguous memory. In addition, we assume a distinct register, which is set to 1 at the beginning of (a). The root of each currently large proposer writes 1 into a randomly chosen memory cell of the root of the acceptor it proposes to. Then each of the vertices of the acceptor selects randomly one of the cells of their root, and if any chosen cell has value 0, then it writes a 0 in the register of the root of the acceptor. Otherwise, they do nothing. The root of the acceptor then reads its register and declares the acceptor large if the register has value 1 and otherwise declares the acceptor small. Finally, each currently large proposer then resets its chosen memory cells to be 0, and the root of the acceptor sets its register to 1.

To do (b) in constant time on the WRAM, we let each root of a large supervertex first make an independent probabilistic choice and then have the processors associated with each node of the set read the result.

Stage 3. We run the (deterministic) connectivity algorithm of Shiloah and Vishkin [1982] until the number of supervertices does not change.

Finally, output the connected components. These are given by the supervertices remaining.

3.4. Probabilistic Analysis of Algorithm MERGE. The following will be needed in the analysis:

FACT 1. For any n, p, β' with $n > 0$, $1 \geq p \geq 0$, and $0 < \beta' < 1$,

$$(a) \quad \sum_{k=0}^{\lfloor (1-\beta')np \rfloor} \binom{n}{k} p^k (1-p)^{n-k} \leq e^{-(\beta')^2 np/2}$$

and

$$(b) \quad \sum_{k=\lceil (1+\beta')np \rceil}^n \binom{n}{k} p^k (1-p)^{n-k} \leq e^{-(\beta')^2 np/3}.$$

The above fact follows from the bounds of Chernoff [1952].

In this section we prove the following theorem:

THEOREM 3.1. *Algorithm MERGE computes the connected components of $G_{n,p}$, with $p \geq c/n$ and $c \geq 192$, in expected parallel time $O(\log \log n)$ in the WRAM. The expected number of processors used is $O(n + pn^2)$. The probability that the parallel time exceeds $O(\log \log n)$ is $O(1/n)$. (Note that with a constant factor of time increase we can reduce the number of processors to $n + pn^2$.)*

PROOF OF THEOREM 3.1. We analyze each stage of the algorithm. The combined results yield the proof of the theorem.

(a) *Analysis of Stage 1.* Let K_0 be the number of supervertices at the end of stage 1. Let K_1 be the number of supervertices of size at least $2 \log^6 n$ and let K_2 be the number of supervertices of size less than $2 \log^6 n$. Then K_0 is at most $n/4 \log^4 n$ (since the number of supervertices drops by one-half on each phase). Also, at least $n/2$ of the nodes will be in supervertices of size at least $2 \log^6 n$. (If the opposite was true, then K_2 would be greater than $(n/2)/(2 \log^6 n)$, i.e., greater than $n/(4 \log^6 n)$, but then $K_2 > K_0$, a contradiction.)

(b) *Analysis of Stage 2*

(b1) *Preliminary lemmas.* It is easy to see that

PROPOSITION 3.1. *At the end of (a) of phase i of stage 2, each large supervertex has size at least $\lceil s(i)/(\beta \log n) \rceil$, and this holds for every i .*

PROOF. The number of vertices in the large supervertex has to be at least equal to the number of entries of the table. \square

We now show

LEMMA 3.1. *If a supervertex has size $\geq s(i)$, then it is declared large with probability at least $1 - n^{-(\beta-1)}$.*

PROOF. Let $p(i)$ be the number of cells of the table of the supervertex, which stay at value 0 after (a) of stage 2. We know that $s(i)$ is increasing with i and $s(1)$ is $2 \log^6 n$. The probability that the particular supervertex fails to be declared large is

$$\begin{aligned} & \text{Prob}\{\text{there is a particular memory cell staying at value 0}\} \\ &= \left(1 - \frac{1}{s(i)/(\beta \log n)}\right)^{s(i)} \leq e^{-\beta \log n} = n^{-\beta}. \end{aligned}$$

So, $\text{Prob}\{p(i) = 0\} \leq n \cdot n^{-\beta} = n^{-(\beta-1)}$. \square

DEFINITION 3.1. Let N_i be the number of large supervertices at the end of (a) of phase i .

LEMMA 3.2. *For any $\beta' \in (0, 1)$, if $N_i \geq s(i)\alpha(\beta - 1)$, then the number of large acceptors at the end of (b) is at least*

$$L = (1 - \beta')N_i \frac{\log^k n}{\alpha s(i)},$$

with probability $\geq 1 - n^{-k(\beta')^2(\beta-1)/2}$.

PROOF. By Fact 1(a), applying it to the Bernoulli experiment of N_i trials, and success probability

$$r = \frac{\log^k n}{\alpha s(i)},$$

we get

$$\begin{aligned} \text{Prob}\{\text{number of successes} \geq L\} &\geq 1 - \exp(-(\beta')^{2N_i r/2}) \quad (\text{by Fact 1(a)}) \\ &\geq 1 - \exp(-(\beta')^{(2N_i)((\log^k n)/2\alpha s(i))}) \\ &\geq 1 - n^{-k(\beta')^2(\beta-1)/2}, \quad \text{since } \log^k n \geq K \log n. \end{aligned}$$

□

By Lemma 3.2 and Proposition 3.1 we then get

COROLLARY 3.1(a). *The number of nodes in acceptors at the end of phase i , step (b) of stage 2, is at least $(1 - \beta')N_i \log^{k-1} n(1/\alpha\beta)$, with the same probability as that stated in Lemma 3.2.*

PROOF. Just multiply L (of Lemma 3.2) with $[s(i)/(\beta \log n)]$, specified in Proposition 3.1. □

By working as in Lemma 3.2 and applying Fact 1(b), we also get

COROLLARY 3.1(b). *For any $\beta'' \in (0, 1)$, if $N_i \geq s(i)\alpha(\beta - 1)$, then the number of large acceptors at the end of (b) is at most $(1 + \beta'')N_i(\log^k n/\alpha s(i))$, with probability at least $1 - n^{-K(\beta'')^2(\beta-1)/3}$.*

PROOF. By Fact 1(b), applying it to the Bernoulli experiment of N_i trials, and success probability $r = \log^k n/\alpha \cdot s(i)$. □

(b2) *Determining the number of nodes in large supervertices.* We now inductively assume that, for $i \geq 1$, the event “at least $n/2^i$ nodes are in large supervertices” holds with probability at least $1 - n^{-2}$. The basis ($i = 1$) was proved previously, in Section 3.4(a). The following results prepare for the induction step:

We first get an estimate of N_i based on the inductive hypothesis. Since the product of the number of large supervertices times their minimum size has to be at least equal to the minimum number of nodes in them, we get

$$(*) \quad N_i \cdot \frac{s(i)}{\beta \log n} \geq \frac{n}{2^i} \quad \text{implying} \quad N_i \geq \frac{n \cdot \log n \cdot \beta}{2^i s(i)}.$$

DEFINITION 3.2. Let q be the probability that a random edge, not yet used at stage i , goes into an acceptor, given the inequality of Corollary 3.1(a).

COROLLARY 3.2.

$$q \geq (1 - \beta') \cdot \frac{\log^k n}{\alpha s(i) 2^i}.$$

PROOF.

$$\begin{aligned} q &= \frac{\# \text{ nodes in acceptors at end of phase } i}{n} \\ &\geq \frac{(1 - \beta') N_i \log^{k-1} n (1/\alpha \beta)}{n} \quad (\text{by Corollary 3.1(a)}) \\ &\geq \left(\frac{1}{n}\right) (1 - \beta') \log^{k-1} n \left(\frac{1}{\alpha \beta}\right) \frac{(n \log n) \beta}{2^i s(i)} \quad (\text{by inequality } (*)) \end{aligned}$$

implying

$$q \geq (1 - \beta') \cdot \left(\frac{1}{\alpha}\right) \cdot \frac{\log^k n}{2^i s(i)}. \quad \square$$

Our next lemma is “at the heart” of the proof of the induction step and is central to the proof of Theorem 3.1. (Compare with the main argument in the preview of the analysis, Section 3.2.)

LEMMA 3.3. *For any $\alpha > 4$, each large proposer, P , has at least one departing edge into a node of an acceptor, with probability at least $1 - n^{-\alpha}$, given the condition of Corollary 3.1(a) (about the number of nodes in acceptors at the end phase i).*

PROOF. Let $|P|$ be the number of nodes in P . Since P is large, $|P| \geq s(i)/\beta \log n$. Let $c' = c/3$. Since edges are not reused by the algorithm, each new edge tried is a random edge, with probability $p' \geq c/3n \cdot 2^i = c'/n \cdot 2^i$ of being there. \square

Now there are $(n-1)|P|$ possible edges with (at least) one end in P . The probability that each of them both appears and joins P with an acceptor is $p'q$, by using Corollary 3.2 and the fact that the estimation of q is conditioned on the event that the edge appears in the instance of the random graph. Hence, the probability, g , that P has no departing edge into a node of an acceptor is

$$\begin{aligned} (1) \quad g &\leq (1 - p'q)^{(n-1)|P|} \\ &\leq \left(1 - \left(\frac{c'}{n}\right) \left(\frac{1 - \beta'}{\alpha}\right) \left(\frac{\log^k n}{4^i s(i)}\right)\right)^{(n-1)(s(i)/(\beta \log n))} \end{aligned}$$

Let $\lambda_1 = (1 - \beta')/\alpha$. By using the known inequality $(1 - (F/x))^x \leq e^{-F}$, and putting (1) in this form, with $x = (n-1)(s(i)/\beta \log n)$ and

$$F = c' \left(\frac{n-1}{n}\right) \left(\frac{1}{\beta \log n}\right) \left(\frac{1 - \beta'}{\alpha}\right) \left(\frac{\log^k n}{4^i}\right),$$

we get

$$(2) \quad g \leq \exp\left(-c'\left(\frac{n-1}{n}\right)\left(\frac{\lambda_1}{\beta \log n}\right)\left(\frac{\log^k n}{4^i}\right)\right).$$

However, $4^i \leq 2^{2d_0 \log \log n} \leq (\log n)^{2d_0}$. So, by choosing $d_0 < 1$,

$$(3) \quad g \leq \exp\left(-c'\left(\frac{n-1}{n}\right)\left(\frac{\lambda_1}{\beta}\right)\left(\log n^{K-2d_0-1}\right)\right).$$

Also, for all $n \geq 2$, $(n-1)/n > \frac{1}{2}$ and, furthermore,

$$\log n^{K-2d_0-1} \geq (K-2) \log n$$

for $k \geq 3$. So, (3) becomes

$$g \leq n^{-c'(k-2)(\lambda_1/2\beta)} = n^{-\alpha} \quad \text{for } \alpha = (K-2)c'\left(\frac{\lambda_1}{2\beta}\right),$$

i.e., $\alpha = (k-2)c'((1-\beta')/(2\alpha\beta))$. We can now choose K , β' (free up to now), and α , β to make $\alpha > 4$. Up to now, the restrictions on our constants are:

$$\alpha > 1,$$

$$\beta > 3,$$

$$0 < \beta' < 1,$$

$$c' \geq 64,$$

which are consistent with $\alpha > 4$, if we choose $k = 3$.

We now complete the proof of the induction step.

LEMMA 3.4 (Induction Step). *At least $n/2^{i+1}$ of the nodes are (at the end of phase i , stage 2) in supervertices of size at least $s(i+1)$, with probability at least $1 - n^{-2}$.*

In order to prove Lemma 3.4, we need the following remarks, A and B.

REMARK A. Let us assume that we have A acceptors and Am proposers and that each proposer has (at least) one outgoing edge, which is directed to one acceptor, selected at random from the set of A acceptors, with probability $1/A$. Furthermore, let m be of the form $m = \lambda(n)b \log n$ where $n > \lambda(n) > 1$, $b > 4$ and n is the number of vertices. Then the following event holds with (conditional) probability $\geq 1 - n^{-(b-2)}$.

“Each acceptor is proposed by at least $\lambda(n)$ proposers.”

PROOF OF REMARK A. Let us separate the proposers in $\lambda(n)$ groups of $b \cdot A \cdot \log n$ proposers each. Consider first a particular group g . The probability that a particular acceptor does not get an edge is

$$\leq \left(1 - \frac{1}{A}\right)^{bA \log n} \leq n^{-b}.$$

Hence, the probability that there exists an acceptor which is not being proposed, is

$$\leq n \cdot n^{-b} = n^{-(b-1)}.$$

Therefore,

$$\text{Prob} \left\{ \begin{array}{l} \text{all acceptors get at least one} \\ \text{edge each, because of group } g \end{array} \right\} \geq 1 - n^{-(b-1)}.$$

The $\lambda(n)$ groups of proposers act in an independent way. Let E_g be the event "all acceptors get at least one edge each, because of group g " and consider an enumeration $1, 2, \dots, \lambda(n)$ of the groups. Let E be the event $E_1 \cup E_2 \cup \dots \cup E_{\lambda(n)}$. Then

$$\begin{aligned} \text{Prob}(E) &\geq (1 - n^{-(b-1)})^{\lambda(n)} \geq 1 - \lambda(n)n^{-(b-1)} \\ &\geq 1 - n^{-(b-2)}. \end{aligned}$$

The event E means that each acceptor is proposed by (at least) $\lambda(n)$ proposers. \square

By conditioning on the events described in Lemmas 3.1, 3.2, 3.3, and the corollaries, we have (for stage i)

$$A \geq N_i(1 - \beta') \frac{\log^k n}{\alpha \cdot s(i)} \quad \text{and} \quad A \leq (1 + \beta'') N_i \frac{\log^k n}{\alpha \cdot s(i)}$$

and that the other conditions of Remark A are satisfied (with overwhelming probability), since

$$\begin{aligned} \# \text{ proposers} &= N_i - A \\ &\geq A \left(\frac{N_i}{A} - 1 \right) \\ &\geq A \cdot \left(\frac{\alpha \cdot s(i)}{(1 + \beta'') \log^k n} - 1 \right), \end{aligned}$$

i.e.,

$$m = \frac{\alpha \cdot s(i)}{(1 + \beta'') \log^k n} - 1 \geq 4 \log n \quad \text{for all } i.$$

So, we conclude that (conditioned on the events of Lemmas 3.1, 3.2, 3.3, and the corollaries), at the end of stage i , each acceptor will be proposed by at least

$$\frac{m}{b \log n} = \frac{\alpha \cdot s(i)}{b(1 + \beta'') \log^{k+1} n} - \frac{1}{b \log n}$$

proposers (with conditional prob $\geq 1 - n^{-(b-2)}$). Therefore, the new size of the formed supervertices will be at least

$$\left(\frac{s(i)}{\beta \log n} \right) \left(\frac{\alpha \cdot s(i)}{b(1 + \beta'') \log^{k+1} n} \right) + \frac{s(i)}{\beta \log n},$$

i.e., at least

$$\frac{\alpha \cdot s^2(i)}{b\beta(1 + \beta'') \log^{k+2} n} \geq \frac{\alpha \cdot s^2(i)}{b\beta(1 + \beta'') \log^5 n},$$

which is $\geq s(i+1)$ if we choose $\varepsilon > 1$. By multiplying probabilities of the events described in Lemmas 3.1, 3.2, 3.3, and Remark A, we conclude that

REMARK B. There is a constant $h > 2$ such that the following event is true with probability at least $1 - hn^{-2}$: "Each acceptor of phase i grows into a supervertex of size $\geq s(i+1)$ at the end of phase i ."

Since almost all the proposers propose to the acceptors at phase i , we conclude (by Lemmas 3.3 and 3.1 applied for phase $i+1$) that: at least one-half of the nodes which were in large supervertices at the beginning of phase i , end up in large supervertices at the beginning of phase $i+1$, with probability at least $1 - h' \cdot n^{-2}$, where $h' = 3h > 6$.

This concludes the proof of the induction step. \square

(b3) *What happens at the end of Stage 2?* The induction of (b2) was carried out because we assumed (in Lemma 3.2) that $N_i \geq s(i)\alpha(\beta - 1)$. This inequality (let us call it inequality (I)) certainly holds for $i = 1$. If the inequality holds throughout the induction argument, then we conclude that, at the end of stage 2, there exists a supervertex of size at least $n/\log^5 n$ with probability at least $1 - h' \cdot n^{-2}$. This is due to the induction hypothesis and to the fact that stage 2 lasts $d_0 \log \log n$ phases.

Now let us assume that the inequality (I) is first violated during phase i of stage 2, and that there exists a supervertex of size greater than $n/\log^5 n$. Since the induction is carried out until phase i , we can apply the inequality (*), i.e.,

$$N_i \geq \frac{n \log n \cdot \beta}{2^i s(i)}.$$

If y_1 is the number of supervertices (like B) which will connect to A in one phase, then

$$\text{Prob}\{y_1 = 0\} = \left(1 - \frac{1}{\sqrt{n}}\right)^{\sqrt{n}} < e^{-1}.$$

Then the probability that there is no pair of such supervertices A, B which merge is $(1 - e^{-1})^{\sqrt{n}} \leq e^{-c''\sqrt{n}}$, with $c'' = -\log(1 - e^{-1})$. We conclude that, with overwhelmingly high probability, some large supervertex will double its size in one phase.

Working then as in Case 1, we conclude that stage 3 will need $O(\log \log n)$ more phases to complete the finding of the connected components. This completes the proof of Lemma 3.6. \square

The combined results of (a), (b), (c), and (d) complete the proof of Theorem 3.1. \square

3.5. Improving the Number of Processors

(a) *The CREW PRAM Case.* We now assume the CREW PRAM model with probabilistic choice instead of the WRAM. In the CREW PRAM we can run Algorithm MERGE in $O(\log n \cdot \log \log n)$ expected time, by simulating each probabilistic selection of a processor in a writeconflict by a tree of pairwise selections.

It is shown by Chin *et al.* [1982] that a minimum of n elements can be computed on a CREW PRAM in time T , where

$$T = \left\lceil \frac{n}{k} \right\rceil - 1 + \log k \quad \text{for } k \leq \frac{n}{2}, \quad k = \text{number of processors.}$$

Using the above technique, we have that the total expected time is

$$\sum_{t=0}^{\log \log n} \left(\frac{s(t)}{k} - 1 + \log k \right) \leq O\left(\frac{n}{k} + (\log \log n) \log k\right).$$

Selecting $k = n/(\log n \cdot \log \log n)$ we have the optimal value $O(\log n \cdot \log \log n)$ of the expected total time with an expected number of $nk = n^2/(\log n \cdot \log \log n)$ processors.

COROLLARY 3.3. *The expected parallel time for connected components of the random graph $G_{n,p}$ with $p \geq c/n$ is $O(\log n \cdot \log \log n)$ in the CREW PRAM model and the expected number of processors is $n^2/(\log n \cdot \log \log n)$.*

(b) *The WRAM Case.* By Theorem 3.1, Algorithm MERGE needs $O(pn^2 + n)$ processors on the average. This is $O(n)$ in the case $p = \Theta(1/n)$.

For the cases of nonsparse graphs (i.e., $p = (c/n) \cdot \omega(n)$ with $\omega(n) \rightarrow \infty$ as $n \rightarrow \infty$) we modify the algorithm as follows:

1. We use $cn = (pn^2/\omega(n))$ processors to select a total of cn of the edges of the input $G_{n,p}$. This can be done in $O(1)$ time and a new, sparser, graph is constructed, $G_{n,p'}$, with $p' \approx c/n$.
2. We run Algorithm MERGE on this (sparse) graph by using $O(p'n^2 + n) = O(n)$ processors and construct its connected components.
3. At this stage, $O(\log n)$ components have been constructed (with probability tending to 1 as $n \rightarrow \infty$). They consist of a giant component and $O(\log n)$ "small" components with at most εn vertices ($\varepsilon < 1$) in all of them (as we showed in the analysis of stage 3).

Then, by studying only the edges emanating out of the small components ($O(n)$ of them on the average) and not used in step 2, we can complete the construction of the connected components of the original graph. This requires running a deterministic connectivity algorithm on a multigraph of $O(\log n)$ vertices and $O(n)$ edges on the average. By using $O(1)$ time to collapse multiple edges among each two components into one edge, we can then complete the process in $O(\log \log n)$ time in the WRAM.

By considering steps 1–3 we have

COROLLARY 3.4. *The expected parallel time for connected components of the random graph $G_{n,p}$ with $p \geq c/n$ is $O(\log \log n)$ in the WRAM model, and the expected number of processors is $O(n)$.*

Acknowledgments We apologize to U. Vishkin and Y. Shiloah for stating in a previous version of this paper that their parallel connectivity algorithm could be easily derived from a previously known algorithm.

Appendix. The Depth of a Random Graph $G_{n,p}$ and of a Random Digraph $D_{n,p}$ We prove here Theorems 1.1 and 1.2.

LEMMA A.0. *Theorem 1.1 implies Theorem 1.2.*

PROOF. Let D be any instance of $D_{n,p}$. The depth of D cannot exceed twice the depth of the underlying undirected version G of D (which is produced from D by removing all edge directions) with high probability. In fact, it can be easily shown that the depth of D is less than or equal to ((the depth of G) + $\alpha \cdot \log n$), with probability at least $1 - n^{-\alpha}$, where α is any constant > 1 . However, G is an instance of $G_{n,p'}$ with $p' = 2p - p^2$. Theorem 1.1 holds for $G_{n,p'}$ for p' in the range

$$R = [0, 1] - \left[\frac{c_0}{n}, \frac{2c_0}{n} - \left(\frac{c_0}{n} \right)^2 \right],$$

where c_0 is a fixed constant (to be determined in the proof of Theorem 1.1). Hence, Theorem 1.2 holds for $D_{n,p}$ for p determined by $p' = 2p - p^2 \in R$.

From this inequality and the complement of (I), we get

$$\frac{n\beta \log n}{2^i s(i)} \leq N_i < s(i)\alpha(\beta - 1)$$

which implies that

$$\frac{n\beta \log n}{2^i \alpha(\beta - 1)} < s^2(i),$$

i.e., that $s(i) > \gamma \sqrt{n \log n}$ where

$$\gamma = \frac{\sqrt{\beta}}{\sqrt{\alpha(\beta - 1)}}.$$

We can certainly choose β to get $\gamma > \frac{1}{2}$. So we conclude that

LEMMA 3.5. *After the end of stage 2, with probability at least $1 - n^{-2}$, we either have a supervertex which is bigger than $n/\log^5 n$ vertices in size, or we have less than \sqrt{n} supervertices, each of size at least $\gamma \sqrt{n \log n}$, with $\gamma > \frac{1}{2}$.*

PROOF. By Lemma 3.1, 3.2, 3.3, 3.4, and by the remarks of part (c) of the analysis. \square

Now we prove our full theorem, in part (c) of the analysis.

(c) Analysis of Stage 3

LEMMA 3.6. *At most $O(\log \log n)$ phases of stage 3 leave $O(\log^5 n)$ supervertices around, with probability at least $1 - n^{-3}$. Then the algorithm of stage 3 finds the connected components of the graph in $O(\log \log n)$ time.*

PROOF. By Lemma 3.5, either we already have a very large supervertex of size at least $n/\log^5 n$ at the end of stage 2, or we have $\leq \sqrt{n}$ supervertices of sizes at least $\gamma \sqrt{n \log n}$ each, and the probability of the disjunction is $\geq 1 - n^{-2}$. Stage 3 examines random edges with density c'/n . We now examine two cases:

Case 1. In the case of the existence of a large supervertex S of size $> n/\log^5 n$, in $O(\log \log n)$ phases, we claim that all but $\log^5 n$ of the remaining vertices will join S with high probability. To prove this, we work as follows: The probability that one particular vertex does not join with S directly is at most

$$\left(1 - \frac{c'}{n}\right)^{n/\log^5 n} \leq 1 - \frac{c'}{\log^5 n},$$

i.e., the probability q that a particular vertex joins with S directly is at least $c'/\log^5 n$.

If x_1 is the number of vertices which join directly, we have (from the Chernoff bounds)

$$\text{Prob}\{x_1 \geq (1 - \delta)qN\} \geq 1 - \exp\left(\frac{-qN\delta^2}{2}\right),$$

where $N = n - n/\log^5 n$ is the total number of vertices not in S , in the worst case, and δ is a constant in $(0, 1)$. We conclude that

$$\text{Prob}\left\{x_1 \geq (1 - \delta') \frac{c'n}{\log^5 n}\right\} \geq 1 - \exp\left(-\frac{\delta^2}{2} \left(\frac{c'n}{\log^5 n}\right)\right),$$

where $\delta < \delta' < 1$. By choosing δ suitably, we can make $c'(1 - \delta') > 1$. Hence, we conclude that

$$\text{Prob}\left\{x_1 \geq \frac{n}{\log^5 n}\right\} \geq 1 - \exp\left(\frac{\delta^2}{2} \left(\frac{c'n}{\log^5 n}\right)\right).$$

That is, S doubles in the first phase of stage 3, with probability $\geq 1 - \exp(-\delta^2 c'n)/(2 \log^5 n)$. While $|S| \leq n/2$, S will continue to double with at least the same probability in each phase, since the first phase is the most difficult. Hence, in $O(\log \log n)$ phases, S will become of size $> n/2$, with

$$\text{probability} \geq 1 - O(\log \log n) \cdot \exp\left(-\frac{\delta^2 c'n}{2 \log^5 n}\right) \geq 1 - e^{-\sqrt{n}} \quad (\text{Remark 1})$$

As soon as S is of size $> n/2$, the probability that a particular vertex will fail to join in one phase will be at most $(1 - (c'/n))^{n/2} < e^{-c'/2}$. This implies that, if $N' < n/2$ is the total number of vertices not in S now, at least $(1 - \delta)N' \cdot (1 - e^{-c'/2})$ of them will join S in the next step, with probability $\geq 1 - \exp(-(\delta^2/2)(1 - e^{-c'/2})N')$. Assuming that initially N' is $> \log^5 n$, stage 3 needs at most $O(\log \log n)$ additional phases to make S of size at least $n - \log^5 n$, with probability at least $1 - O(\log \log n) \cdot n^{-4}$ (Remark 2).

Finally, the remaining vertices out of S are at most $\log^5 n$ (with probability at least $1 - n^{-3}$, by combining the previous two remarks). It will now take an additional $O(\log \log n)$ parallel time for stage 3 to complete the finding of the connected components.

Case 2. Consider two particular supervertices A, B of sizes at least $\gamma\sqrt{n} \log n$ each. The probability that B connects to A (assuming that B has at least one edge out of it) is at least $1/\sqrt{n}$, since there are at most \sqrt{n} large supervertices around.

A.1. Proof of Theorem 1.1 for $p = \Theta(1/n)$. We prove here Theorem 1.1 for undirected graphs. This provides an upper bound on both the average depth and the depth distribution of $G_{n,p}$ for all $p = \Theta(1/n)$ except those p in

$$\left[\frac{c_0}{n}, \frac{2c_0}{n} - \left(\frac{c_0}{n} \right)^2 \right],$$

where c_0 is the cutoff constant for paths of length $d \log n$, $d > 2$. This provides a partial answer to the problem of finding the depth of $G_{n,p}$ for $p = \Theta(1/n)$, open up to now. The remaining cases of $p = o(1/n)$ and $p = \Omega(1/n)$ can easily be derived by results of Erdos and Renyi [1959] (see Sections A.2 and A.3).

In addition Bollobas [1984] showed that, for

$$p \geq \frac{c \log n}{n}$$

(and c such that $G_{n,p}$ becomes almost surely connected), the mean depth d of $G_{n,p}$ satisfies

$$\bar{d} \leq \left\lceil \frac{\log n + 6}{\log \log n} \right\rceil + 3.$$

In fact, Bollobas also showed that, for any

$$p = \frac{d(n)}{2n}$$

with $d(n) - \log n \rightarrow \infty$ as $n \rightarrow \infty$, the mean depth of $G_{n,p}$ satisfies

$$\bar{d} \leq \left\lceil \frac{\log n + 6}{\log d(n)} \right\rceil + 3.$$

Although Bollobas conjectured that his results hold also for $p = \Theta(1/n)$, his proof techniques do not carry out in that case, since they are heavily based on the fact that $pn \geq \log n$.

Consider the following two-stage experiment: Draw an instance of G_{n,p_1} with

$$p_1 = \frac{c'}{n} \quad (c' > 0).$$

Then, for each edge which failed to appear, draw again with probability

$$p_2 = \frac{c'}{n}.$$

The result of the combined experiments is an instance of the graph $G_{n,p}$ with $p = p_1 + (1 - p_1)p_2 = 2c'/n - (c'/n)^2 = \Theta(c'/n)$. The first stage of the above experiment constructs an instance I of G_{n,p_1} .

Let E be the event "the instance I has a path of length $\delta \cdot \log n$ (where δ is a fixed constant)" and let E' be the complement of E . Let us consider the probability of E . We need the following lemma:

LEMMA A1.1 [Erdos and Renyi, 1960]. *Given $G_{n,p}$ with $p = c/n$, let $\tau_{\delta \cdot \log n}$ be the number of paths of length $\delta \cdot \log n$ in $G_{n,p}$. Then*

$$\lim_{n \rightarrow \infty} \text{Prob}\{\tau_{\delta \cdot \log n} = 0\} = e^{-\lambda}$$

with

$$\lambda = (2c)^{\delta \cdot \log n - 1} \frac{(\delta \cdot \log n)^{\delta \cdot \log n - 2}}{(\delta \cdot \log n)!}.$$

The above lemma can be used to prove that "there exists $c_0 > 0$: for $c \geq c_0$ the graph $G_{n,p}$ with $p = c/n$ has at least one path of length $\delta \cdot \log n$ with probability $\rightarrow 1$ as $n \rightarrow \infty$ " and for $c < c_0$ the $G_{n,p}$ with $p = c/n$ has no such path with probability $\rightarrow 1$ as $n \rightarrow \infty$ (to prove that, find the c_0 such that $\lambda \rightarrow \infty$ if $c \geq c_0$ and $\lambda \rightarrow 0$ if $c < c_0$ for $n \rightarrow \infty$). Assume in our construction that $c' \geq c_0$. Then $\text{Prob}(E) \geq 1 - n^{-\alpha'(c')}$, where $\alpha'(c') > 2$ is a constant depending on c' . Let us condition the following on the event E .

We start from the vertices of the path T of length $\delta \cdot \log n$ (at least such a path exists, by E) and use them as the initial border set B_0 for a breath-first-search (BFS) process in the graph induced on the vertices of $G_{n,p}$ by just the second stage of the experiment.

Let $|B_0| = \delta \cdot \log n$ and let us find the distribution of the number of vertices to which the vertices of T are immediate neighbors. The set of these vertices will be the new border set of the second stage of BFS, and let us call it B_1 . We are interested in estimating the $\text{Prob}\{|B_1| \geq 2 \cdot \delta \cdot \log n\}$. Let $|S_0|$ be equal to $|B_0| = \delta \cdot \log n$ (the number of visited vertices at the beginning of the first stage of BFS). The number of unvisited vertices is $n - |S_0| = n - \delta \cdot \log n$ and each of them has probability

$$h_0 = 1 - (1 - p_2)^{\delta \log n}$$

to be connected to B_0 . Hence

$$\text{Prob}\{|B_1| = x\} = \binom{n - |S_0|}{x} h_0^x (1 - h_0)^{n - \delta \log n - x},$$

but $h_0 = 1 - (1 - c'/n)^{\delta \cdot \log n}$ and since $\log n/n \rightarrow 0$ as $n \rightarrow \infty$ we have that $h_0 \rightarrow (c'\delta \log n)/n$ as $n \rightarrow \infty$. The mean value of $|B_1|$ is

$$(n - |S_0|)h_0 = c'\delta \log n - \frac{c'\delta^2 \log^2 n}{n} \rightarrow c'\delta \log n \quad \text{as } n \rightarrow \infty.$$

Let $c' > 2$ and let us find the probability $\text{Prob}\{|B_1| < 2\delta \cdot \log n = 2|B_0|\}$. From p. 140 of [Feller, 1968] this is less than

$$Q = \binom{n - |S_0|}{2\delta \log n} h_0^{2\delta \log n} (1 - h_0)^{n - \delta \log n - 2\delta \log n} \cdot r,$$

where

$$r = \frac{(n - \delta \log n - 2\delta \log n + 1)h_0}{(n - \delta \log n + 1)h_0 - 2\delta \log n},$$

which is asymptotically equal to

$$\frac{c'\delta \log n}{c'\delta \log n - 2\delta \log n},$$

tending to $c'/(c' - 2)$ as $n \rightarrow \infty$. Then Q is asymptotically equal to

$$\left(\frac{n - \delta \log n}{2\delta \log n}\right) \left(\frac{c'\delta \log n}{n}\right)^{2\delta \log n} \left(1 - \frac{c'\delta \log n}{n}\right)^{n - 3\delta \log n} \cdot \frac{c'}{c' - 2},$$

i.e., Q is less than or equal to

$$\left(\frac{n - \delta \log n}{2\delta \log n}\right)^{2\delta \log n} \left(\frac{c'\delta \log n}{n}\right)^{2\delta \log n} n^{-c'\delta} \cdot \frac{c'}{c' - 2},$$

i.e.,

$$Q \leq n^{2\delta \log(c'/2)} \cdot n^{-c'\delta} \cdot \frac{c'}{c' - 2},$$

i.e.,

$$Q \leq n^{-\delta(c' - 2 \log(c'/2)) + \log(c'/(c' - 2))/\log n},$$

which implies $Q \leq n^{-\beta}$ for any $\beta > \delta(c' - 2 \log(c'/2))$. Hence,

$$\text{Prob}\{|B_1| \geq 2|B_0|\} \geq 1 - n^{-\beta}.$$

We now inductively form sets S_k and B_k such that $S_k = B_0 \cup B_1 \cup \dots \cup B_k$ and

$$h_k = 1 - (1 - p_2)^{n - |S_k|},$$

and $|B_{k+1}|$ being selected according to the density

$$\text{Prob}\{|B_{k+1}| = x\} = \binom{n - |S_k|}{x} h_k^x (1 - h_k)^{n - |S_k| - x}.$$

(This models the BFS process.) We claim that while $|S_k| \leq n/2$,

$$\text{Prob}\{|B_i| \geq 2|B_{i-1}| \text{ given } F\} \geq 1 - n^{-\beta},$$

where

$$F = \bigwedge_{j=1}^{i-1} (|B_j| \geq 2|B_{j-1}|).$$

This is so, since the first extension (from the smallest border set B_0 , with $|B_0| = \delta \cdot \log n$) is the most difficult. Then the event

$$E_1 = \bigwedge_{i=1}^{k+1} (|B_i| \geq 2|B_{i-1}|)$$

has probability at least

$$\prod_{j=1}^{k+1} (1 - n^{-\beta}) \geq 1 - n^{-\beta+1},$$

and let k be the last index for which $|S_k| \leq n/2$. Set $|S_k| = (n/2)(1 - \varepsilon)$, with $|B_k| \geq |S_k|/2 = (n/4)(1 - \varepsilon)$. From $|S_k| = |B_0| + \dots + |B_k| \geq |B_0| + 2|B_0| + \dots + 2^k|B_0|$ we get $|S_k| \geq (2^{k+1} - 1)|B_0|$, implying

$$k = \left\lfloor \log \left(\frac{|S_k|}{|B_0|} + 1 \right) - 1 \right\rfloor,$$

i.e.,

$$k = \left\lfloor \log \left(\frac{n(1 - \varepsilon)}{2\delta \log n} + 1 \right) - 1 \right\rfloor,$$

i.e.,

$$k = \left\lfloor \log n + \log \frac{1 - \varepsilon}{2\delta} - \log \log n - 1 \right\rfloor.$$

Hence,

$$|S_{k+1}| = |S_k| + |B_{k+1}| \geq |S_k| + 2|B_k| \geq \frac{n}{2}(1 - \varepsilon) + \frac{n}{2}(1 - \varepsilon) \geq n(1 - \varepsilon).$$

So, we proved that

LEMMA A1.2. *Conditioned on the event E , the BFS process will visit at least $n(1 - \varepsilon)$ nodes of G_{n,p_2} in at most $\log n$ stages, with probability at least $1 - n^{-\beta+1}$.*

DEFINITION. Let T_1 be the number of stages needed by the BFS in order to visit at least $n(1 - \varepsilon)$ nodes of G_{n,p_2} . Let V_1 be the set of visited vertices. We just proved that, with probability at least $1 - n^{-\beta+1}$, there exists a path of size $O(\log n)$ between any two nodes of V_1 in G_{n,p_2} (conditioned on the event E).

Let us consider the subgraph G_2 of G_{n,p_2} induced by the set of vertices not visited by the BFS in T_1 stages (i.e., the vertices in $V - V_1$). Consider a spanning tree T_{r_1}, \dots, T_{r_m} in each of the connected components CC_1, \dots, CC_m of G_2 . Let us partition each tree T_{r_j} ($j = 1, \dots, m$) in a set π of connected pieces of size $q \cdot \log n$ (where q is an appropriate constant, to be determined). There may be some pieces of smaller size which, however, connect to at least a piece of size $q \cdot \log n$ (or they belong to a very small connected component, of size less than $q \cdot \log n$). The number of pieces of this partition is less than n . Consider a particular connected piece P of size $q \cdot \log n$. The probability that there is no edge connecting a node of P to a node of V_1 is $(1 - p_2)^{n(1-\varepsilon) \cdot q \log n}$, i.e., equal to $(1 - c'/n)^{n(1-\varepsilon)q \log n}$, i.e., $\leq n^{-c'q(1-\varepsilon)}$.

Since the number of such pieces is less than n , the probability that there is such a piece, with no edge connecting any of its nodes to any node of V_1 , is $\leq n \cdot n^{-c'q(1-\varepsilon)}$, i.e., $\leq n^{-c'q(1-\varepsilon)+1}$.

Let E_1 be the event "there exists a path of size $O(\log n)$ between any two nodes of V_1 in G_{n,p_2} ." Let E_2 be the event "for every piece of the partition π , there is at least one edge between some node of that piece and some node of V_1 ."

We have proved that

$$\text{Prob}\{E\} \geq 1 - n^{-\alpha'}, \quad \alpha' > 2,$$

$$\text{Prob}\{E_1 \text{ given } E\} \geq 1 - n^{-\beta+1},$$

and

$$\text{Prob}\{E_2 \text{ given } E, E_1\} \geq 1 - n^{-c'q(1-\varepsilon)+1}.$$

Hence,

$$\text{Prob}\{E \text{ and } E_1 \text{ and } E_2\} \geq 1 - O(n^{-\alpha'} + n^{-\beta+1} + n^{-c'q(1-\varepsilon)+1}),$$

but E and E_1 and E_2 together imply that $G_{n,p}$ has depth which is $\leq 2q \log n + 1 + 2 \log n + \delta \log n$, i.e., $O(\log n)$.

We conclude that

LEMMA A1.3. *The depth of $G_{n,p}$ with $p \geq 2c'/n - (c'/n)^2$ is $\leq (2q + \delta + 2) \log n + 1$, with probability at least $1 - O(n^{-\alpha'} + n^{-\beta+1} + n^{-c'q(1-\varepsilon)} + 1)$.*

Note that $\delta \geq 1$ is arbitrary, c' can be picked up by Lemma A1.1, $\beta = \delta(c' - 2 \log(c'/2))$, and q, ε can be picked up in an arbitrary manner. By selecting them suitably, we can guarantee a probability of at least $1 - n^{-\gamma}$, with $\gamma > 2$, for the event that the depth of $G_{n,p}$ is $O(\log n)$.

Note also that, for $p \leq c'/n$, Lemma A1.4 implies that

$$\text{Prob}\{\text{depth of } G_{n,p} \leq \delta \cdot \log n\}$$

tends to 1 as $n \rightarrow \infty$. Hence, we managed to prove our result for any p not in $[c'/n, 2c'/n - (c'/n)^2]$, where c' is the cutoff constant for paths of length $\delta \cdot \log n$, $\delta > 2$. This proves Theorem 1.1 with $c_0 = c'$. The problem is open for

$$p \in [c'/n, 2c'/n - (c'/n)^2]$$

and we conjecture that Theorem 1.1 holds for that range too. \square

A.2. The Case of $p = o(n^{-1})$

LEMMA A.2 [Erdos and Renyi, 1960]. *Let the average valence of a graph of n vertices and m edges be the number $2m/n$. Let a graph be called balanced if its average valence is greater than or equal to the average valence of any of its subgraphs. Then the probability that $G_{n,p}$ contains at least one member of the class $B_{k,l}$ of balanced graphs of k vertices and l edges is*

$$\text{Prob}\{|B_{k,l}| \geq 1\} = O\left(\frac{p}{n^{-k/l}}\right).$$

COROLLARY A.1. $\text{Prob}\{\text{depth of } G_{n,p} < \log n\} \rightarrow 1$ as $n \rightarrow \infty$ for $p = o(n^{-1})$.

PROOF. For $p = o(n^{-1})$ we can always write p as $p = (c/(n \cdot \omega(n)))$ where $c > 0$ is a constant and $\omega(n) \rightarrow \infty$ as $n \rightarrow \infty$. Let $B = B_{\log n, \log n - 1}$ be the class of paths of $\log n$ vertices. From Lemma A.2 we get

$$\begin{aligned} & \text{Prob}\{G_{n,p} \text{ contains at least one member of } B\} \\ &= O\left(\frac{c}{n \cdot \omega(n) \cdot n^{-\log n / (\log n - 1)}}\right) = O\left(\frac{c}{\omega(n)}\right) \rightarrow 0 \quad \text{as } n \rightarrow \infty. \end{aligned}$$

The above implies that the average depth of $G_{n,p}$ is less than $\log n$ for $p = o(n^{-1})$ and all $n > n_0$ for some n_0 . \square

A.3. The Case of $p = \Omega(n^{-1})$

LEMMA A.3 [Erdos and Renyi, 1960]. Let $p = (c/n) \cdot \omega(n)$ where $c > 0$ is a constant and $\omega(n) \rightarrow \infty$ as $n \rightarrow \infty$. Then if $|B_{k,l}|$ is the number of balanced graphs of the class $B_{k,l}$ (k vertices, l edges) contained in $G_{n,p}$ we have

$$\text{Prob}\{|B_{k,l}| < \omega(n)^m\} = O\left(\frac{1}{\omega(n)}\right)$$

implying

$$\text{Prob}\{|B_{k,l}| = 0\} = O\left(\left(\frac{1}{\omega(n)}\right)^m\right).$$

COROLLARY A.2. The depth of $G_{n,p}$ is $\leq \log n$ for $p = \Omega(n^{-1})$, with probability going to 1 as $n \rightarrow \infty$. This implies that, for $p = \Omega(n^{-1})$, the average depth of $G_{n,p}$ is not more than $\log n$ for all $n > n_0$, for some $n_0 > 0$.

PROOF. Let $B_{k,l}$ be the class of binary balanced trees of n vertices, $n - 1$ edges. Then

$$\text{Prob}\{|B_{k,l}| \geq 1\} \geq 1 - O\left(\left(\frac{1}{\omega(n)}\right)^{n-1}\right).$$

Hence

$$\text{Prob}\{|B_{k,l}| \geq 1\} \rightarrow 1 \quad \text{as } n \rightarrow \infty. \quad \square$$

A.4. The Case of High Density $p \geq cn^{-1/3}$, $c > 1$

LEMMA A.4. The depth of $G_{n,p}$, for $p \geq cn^{-1/3}$, is ≤ 3 , with probability tending to 1 as n goes to ∞ . Here $c > 1$ is a constant.

PROOF. Consider any pair u, v of vertices of $G_{n,p}$. Let S_u (resp. S_v) be the sets of neighbors of u (resp. of v). Then, for any $\beta \in (0, 1)$,

$$\text{Prob}\{|S_u| \geq p(n-1)(1-\beta)\} \geq 1 - e^{-p\beta^2(n-1)/2}$$

by the Chernoff bounds [Chernoff, 1952]. For $\beta = \frac{1}{2}$ we get

$$\text{Prob}\left\{|S_u| \geq \frac{cn^{2/3}}{2}\right\} \geq 1 - e^{-(c/8)n^{2/3}}.$$

Similarly,

$$\text{Prob}\left\{|S_v| \geq \frac{cn^{2/3}}{2}\right\} \geq 1 - e^{-(c/8)n^{2/3}}.$$

Hence

$$\text{Prob}\left\{\text{both } |S_u|, |S_v| \geq \frac{cn^{2/3}}{2}\right\} \geq 1 - 2e^{-(c/8)n^{2/3}}.$$

Let E be the event “both $|S_u|, |S_v| \geq cn^{2/3}/2$.” Then

$$\begin{aligned} & \text{Prob}\{\text{no edge between } S_u, S_v \text{ given } E\} \\ & \leq (1-p)^{|S_u| \cdot |S_v|} \leq \left(1 - \frac{c}{\sqrt{n}}\right)^{(c^2/4)n^{4/3}} \leq e^{-(c^3/4) \cdot n}. \end{aligned}$$

Hence, combining terms,

$$\begin{aligned} & \text{Prob}\{\text{there is a path of length } \leq 3 \text{ between } u, v\} \\ & \geq (1 - 2e^{-(c/8) \cdot n^{2/3}})(1 - e^{-(c^3/4) \cdot n}) \geq 1 - 3e^{-(c/8) \cdot n^{2/3}}. \end{aligned}$$

Hence,

$$\begin{aligned} & \text{Prob}\{\text{for each pair } u, v \text{ there is a path of length } \leq 3 \text{ between them}\} \\ & \geq 1 - 3n^2 e^{-(c/8) \cdot n^{2/3}} \rightarrow 1 \quad \text{as } n \rightarrow \infty. \end{aligned} \quad \square$$

References

- Aleliunas, R., Karp, R. M., Lipton, R. J., Lovász, L., and Rackoff, C. Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems, *Proc. 20th IEEE Symp. on Foundations of Computer Science*, Puerto Rico, 1979, pp. 218–223.
- Angluin, D., and L. G. Valiant, Fast Probabilistic Algorithms for Hamiltonian Paths and Matchings, *J. Comput. System Sci.*, **18**, 1979, 155–193.
- Babai, L., and L. Kucera, Canonical Labelling of Graphs in Linear Average Time, *Proc. 20th IEEE Symp. on Foundations of Computer Science*, Puerto Rico, 1979, pp. 39–46.
- Babai, L., Erdos, and S. M. Selkow, Random Graph Isomorphism, *SIAM J. Comput.*, **9**(3), 1980, 628–634.
- Bollobas, B., The Evolution of Sparse Graphs, in *Graph Theory & Combinatorics*, Academic Press, London, 1984, pp. 35–57.
- Bollobas, B., Extremal Graph Theory with Emphasis on Probabilistic Methods, in *The Evolution of Random Graphs*, Regional Conference Series in Mathematics, No. 62, Conference Board of the Mathematical Sciences, pp. 37–43.
- Bollobas, B., *Random Graphs*, Academic Press, New York, 1985.
- Borodin, A., and J. E. Hopcroft, Routing, Merging and Sorting on Parallel Models of Computation, *Proc. 14th ACM Symp. on Theory of Computing*, San Francisco, CA, 1982, pp. 338–344; also in *J. Comput. System Sci.*, **30**, 1985, 130–145.
- Chernoff, H., A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations, *Ann. of Math. Statist.*, **23**, 1952.
- Chin, F., J. Lam, and T. Chen, Efficient Parallel Algorithms for Some Graph Problems, *Comm. ACM*, **25**(9), 1982, 659–665.
- Cole, R., and U. Vishkin, Deterministic Coin Tossing and Accelerating Cascades: Micro and Macro Techniques of Designing Parallel Algorithms, *Proc. 18th ACM Symp. on Theory of Computing*, 1986, pp. 206–219; also in *Inform. and Control*, **70**, 1986, 32–53.

- Cook, S., Towards a Complexity Theory of Synchronous Parallel Computations, Paper presented at Specker Symp. on Logic and Algorithms, Zurich, February 5–11, 1980; also *Enseign. Math.*, **27**, 1981, 99–124.
- Coppersmith D., P. Raghavan, and M. Tompa, Parallel Graph Algorithms that Are Efficient on Average, *Proc. 28th IEEE Symp. on Foundations of Computer Science*, 1987, pp. 260–269. Also in *Inform. Comput. J.*, **91**, 1989, 318–333.
- Dekel, E., D. Nassimi, and S. Sahni, Parallel Matrix and Graph Algorithms, *SIAM J. Comput.*, **10**(4), 1981, 657–675.
- Erdos, P., and A. Renyi, On the Evolution of Random Graphs, *Publ. Math. Inst. Hungar. Acad. Sci.*, **5A**, 1960, 17–61.
- Feller, W., *An Introduction to Probability Theory and Its Applications*, vol. 1, 3rd edn., Wiley, New York, 1968.
- Fortune, S., and J. Wyllie, Parallelism in Random Access Machines, *Proc. 10th ACM Symp. on Theory of Computing*, 1978, pp. 114–118.
- Gazit, H., An Optimal Randomized Parallel Algorithm for Finding Connected Components in a Graph, *Proc. 27th IEEE Symp. on Foundations of Computer Science*, 1986, pp. 492–501.
- Goldschlager, L., A Unified Approach to Models of Synchronous Parallel Machines, *Proc. 10th ACM Symp. on Theory of Computing*, May, 1978.
- Hirschberg, D., A. Chandra, and D. Sarwate, Computing Connected Components on Parallel Computers, *Comm. ACM*, **22**(8), 1979, 461–464.
- Ja'Ja', J., Graph Connectivity Problems on Parallel Computers, TR GS-78-05, Department of Computer Science, Penn State University, PA, 1978.
- Karp, R. M., The Probabilistic Analysis of Combinatorial Search Algorithms, in *Algorithms and Complexity: New Directions and Recent Results*, J. F. Traub, ed., Academic Press, New York, 1976, pp. 1–19.
- Karp, R. M., and M. Sipser, Maximum Matchings in Sparse Random Graphs, *Proc. 22nd IEEE Symp. on Foundations of Computer Science*, Nashville, TN, 1981, pp. 364–375.
- Karp, R. M., and R. Tarjan, Linear Expected Time Algorithms for Connectivity Problems, *Proc. 12th ACM Symp. on Theory of Computing*, Los Angeles, CA, 1980, pp. 368–377.
- Kucera, L., Parallel Computation and Conflicts in Memory Access, *Inform. Process. Lett.*, **14**(2), 1982, 93–96.
- Lewis, H. R., and C. Papadimitriou, Symmetric Space Bounded Computation, *Theoret. Comput. Sci.*, **19**, 1982, 161–187.
- Reif, J. H., Symmetric Complementation, *Proc. 14th ACM Symp. on Theory of Computing*, San Francisco, CA, May 1982(a), pp. 201–214; also *J. Assoc. Comput. Mach.*, **31**, 1984, 401–421.
- Reif, J. H., On the Power of Probabilistic Choice in Synchronous Parallel Computations, *Proc. 9th Colloq. on Automata, Languages and Programming*, Aarhus, Denmark, July 1982(b), pp. 442–456.
- Reif, J. H., and P. Spirakis, "Random Matroids," *Proc. 12th ACM Symp. on Theory of Computing*, Los Angeles, CA, 1980, pp. 385–347; also rewritten as Probabilistic Analysis of Random Extension-Rotation Algorithms, TR-28-81, Aiken Computer Laboratory, Harvard University, 1981.
- Reif, J. H., and P. Spirakis, k -Connectivity in Random Undirected Graphs, *J. Discrete Math.*, **54**(2), 1985, 1–18.
- Ruzzo, W., On Uniform Circuit Complexity, *Proc. 20th IEEE Symp. on Foundations of Computer Science*, 1979, pp. 312–318; also *J. Comput. System Sci.*, **22**, 1981, 365–383.
- Ruzzo, W., Personal communication, 1982.
- Savage, C., and J. Ja'Ja', Fast, Efficient Parallel Algorithms for Some Graph Problems, Technical Report, Computer Studies, North Carolina State University, 1978.
- Savage, C., and J. Ja'Ja', Fast Efficient Parallel Algorithms for Some Graph Problems, *SIAM J. Comput.*, **10**(4), 1981, 682–691.
- Savitch, W. J., Relationships Between Nondeterministic and Deterministic Tape Complexities, *J. Comput. System Sci.*, **4**(2), 1970, 177–192.
- Schnorr, C. P., An Algorithm for Transitive Closure with Linear Expected Time, *SIAM J. Comput.*, **7**, 1978, 127–133.
- Shamir, E., and E. Upfal, N -Processors Graphs Distributedly Achieve Perfect Matchings in $O(\log^2 n)$

- Beats, *ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, Ottawa, 1982, pp. 238-241.
- Shiloah, Y., and U. Vishkin, Finding the Maximum, Merging and Sorting in a Parallel Computation Model, *J. Algorithms*, 2(1), 1981, 88-102.
- Shiloah, Y., and U. Vishkin, An $O(\log n)$ Parallel Connectivity Algorithm, *J. Algorithms*, 3(2), 1982, 128-148.
- Spirakis, P., Probabilistic Algorithms, Algorithms with Random Inputs and Random Combinatorial Structures, Ph.D. Thesis, Harvard University, December 1981.
- Wyllie, J., The Complexity of Parallel Computation, Ph.D. Thesis, Cornell University, 1979.