# Topics in Parallel and Distributed Computation

A thesis presented

by

Alexandros Gerbessiotis

to

The Division of Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

January MCMXCIII

*To my parents*

## Abstract

With advances in communication technology, the introduction of multiple-instruction multiple-data parallel computers and the increasing interest in neural networks, the fields of parallel and distributed computation have received increasing attention in recent years.

We study in this work the bulk-synchronous parallel model, that attempts to bridge the software and hardware worlds with respect to parallel computing. It offers a high level abstraction of the hardware with the purpose of allowing parallel programs to run efficiently on diverse hardware platforms. We examine direct algorithms on this model and also give simulations of other models of parallel computation on this one as well as on models that bypass it.

While the term parallel computation refers to the execution of a single or a set of closely coupled tasks by a set of processors, the term distributed computation refers to more loosely coupled or uncoupled tasks being executed at different locations. In a distributed computing environment it is sometimes necessary that one computer send to the remaining ones various pieces of information. The term broadcasting is used to describe the dissemination of information from one computer to the others in such an environment. We examine various classes of random graphs with respect to broadcasting and establish results related to the minimum time required to perform broadcasting from any vertex of such graphs.

Various models of the human brain, as a collection of distributed elements working in parallel, have been proposed. Such elements are connected together in a network. The network in the human brain is sparse. How such sparse networks of simple elements can perform any useful computation is a topic currently little understood. We examine in this work a graph construction problem as it relates to neuron allocation in a model of neural networks recently proposed. We also examine a certain class of random graphs with respect to this problem and establish various results related to the distribution of the sizes of sets of neurons when various learning tasks are performed on this model. Experimental results are also presented and compared to theoretically derived ones.

## Acknowledgements

I would like to express my sincere thanks, appreciation and gratitude to my adviser, Professor Les Valiant. His support, guidance, encouragement, influence and contributions to the completion of this thesis have been invaluable. Chapter 5 of this thesis (except few minor results) is joint work with him.

Thanks to all four members of my thesis committee, Professors Harry Lewis, Michael Rabin, Les Valiant and Alan Yuille for their interest in my work and their comments.

I would also like to thank Professor Thomas E. Cheatham, Jr. for allowing me to use the Connection Machine at Aiken Computation Laboratory. The simulations of Chapter 3 were run on this machine.

I have also greatly benefited from discussions with Thanasis Tsantilas and Christos Kaklamanis in the course of my research.

Professor Christos Papadimitriou and the late Professor Yannis Kollias introduced me to theoretical computer science. My appreciation and thanks go to them for that.

Many thanks to my friends in the United States and in Greece and especially to those in my hometown Kavala, in Macedonia, the most beautiful region of Greece, who made my years in graduate school a wonderful, enjoying and unforgettable experience.

My research was supported in part by the National Science Foundation under grants NSF-CCR-89-02500 and NSF-CCR-92-00884 through Professor Leslie G. Valiant.

# Contents

# Chapter 1

# Introduction

## 1.1 Parallel and Distributed Computation

With rapid advances in communication technology, the introduction of high performance multiple-instruction multiple-data parallel computers and the ever increasing attention to neural networks, the fields of parallel and distributed computation have received increasing attention in recent years.

The Parallel Random Access Machine (PRAM) is one of the most widely studied abstract models of a parallel computer. A PRAM consists of a number of independent processors and a shared memory which these processors can access in unit time. There are many different types of PRAMs which are distinguished from each other mainly in the way they can access the shared memory. Numerous parallel algorithms have been developed for this parallel computer model. It is therefore probable that any successful new proposals for a parallel model would need to be able to simulate the PRAM efficiently. This is similar to the requirement that new programs be compatible with older ones since so much time, effort and money have been spent on the older programs.

The advantage of the PRAM model is its simplicity in capturing parallelism and abstracting away the communication requirements of parallel computing. It is these advantages that make the PRAM easy to program and widely used for algorithm development. The PRAM, by abstracting away communication costs, achieves simplicity at the cost of realism. This advantage, therefore constitutes also a disadvantage, since communication issues do appear to be central to parallel computation.

Other, more realistic, models of parallel computing view a parallel computer as a collection of sequential processors, each one having its own local memory. The processors are interconnected by a network which allows them to communicate by sending and receiving messages. Constraints such as the maximum number of pins on a chip, or the maximum width of a data bus, limit the capacity of a processor to communicate with any other processor. It is only possible for a single processor to communicate directly with a few others, in most cases those physically closer to it. If a message needs to be sent to a distant processor it is relayed through a number of intermediate processors. The limitations of an interconnection network are not much different from those of a road network. Congestion of messages (rather than of vehicles) will occur unless these networks are carefully designed and utilized. It is these real-life aspects of parallel computing that we shall consider in this thesis. We shall, more specifically, study the bulk-synchronous parallel model, that attempts to bridge the software and hardware worlds with respect to parallel computing. It offers a high level abstraction of the hardware with the purpose of allowing parallel programs to run efficiently on diverse hardware platforms, just as the von-Neumann model succeeds in doing so for sequential machines.

While the term *parallel computation* refers to the execution of a single or a set of closely coupled tasks by a set of processors, the term *distributed computation* refers to more loosely coupled or uncoupled tasks being executed at different locations. The computers in a distributed computing environment communicate via a communication network whose limitations are very similar to those of a multi-processor interconnection network. In addition to these limitations the links in such a network are usually physically longer and the delays incurred when messages are sent from one computer to another are larger. In a distributed computing environment it is sometimes necessary that one computer send to the remaining ones various pieces of information. Such pieces of information may describe the current status of the connections linking the computers, how one computer can send messages to others and via which intermediate computers, the sets of approved users and the available peripherals, and so on. The term *broadcasting* is used to describe the dissemination of information from one computer to the others in the network. It is important that the time it takes to broadcast a message be as short as possible. This time depends on the topology of the network, the identities of the pairs of nodes that are linked directly to each other. Since this topology may change with time, one needs to model somehow networks of varying topologies and examine their properties as related to broadcasting.

Various models of the human brain, as a collection of distributed elements working in parallel, have been proposed. Typically a so called *neural network* consists of a collection of simple computing elements that each performs a fairly simple computational operation. The elements are connected together in a network. The links between the elements carry elementary messages, often a 0 or a 1. The information carried by these messages is processed in unit time by the element at which they arrive and the result is transmitted via the network to become inputs to other elements. The network in the human brain is sparse. How such sparse networks of simple elements can perform any useful computation is a topic currently little understood.

## 1.2 Graph Theory and a Neural Network Model

The first topic of this thesis is related to a graph construction problem posed by L. G. Valiant in a model of neural computation he suggested in [50]. The purpose of the model is to serve as a vehicle for investigating the functional capabilities of sparse networks of simplified neurons in accumulating knowledge through interactions with the outside world. The specific learning task is to establish in the network a circuit that computes a Boolean expression. The structure of the underlying network of neurons plays an important role in the various learning tasks considered. In that model a neuron may store information related to some concept or attribute. Let the symbols $A, B, C$ represent the neurons that store information related to say, concepts $c_A, c_B$, and $c_C$ respectively. Various modes of learning require that the network have the property that for every two neurons $A$ and $B$ there exists a third one $C$ which is connected to both $A$ and $B$. For example, when the neural network learns conjunctions, information related to the conjunction $c_C = c_A \wedge c_B$ needs to be stored in a neuron $C$ connected to both $A$ and $B$. Our discussion is mainly directed towards neuron allocation when at most a single concept such as $c_C = c_A \wedge c_B$ is to be associated with any pair, $c_A, c_B$.

It is sometimes desirable that not a single neuron but a set of $k$ neurons, for some integer $k$, store information related to some concept or attribute. This *replication factor* $k$, as we shall call it, links neural networks and graph properties and we shall therefore examine the implications of the appearance of this factor in the neuron allocation problem that arises in various learning tasks. Let the sets of neurons $A, B, C, \ldots$ store information related to concepts or attributes $c_A, c_B, c_C, \ldots$ respectively. Let sets $A$ and $B$ be of size $k$ each, and let us assume we would like to learn concept $c_C = c_A \wedge c_B$, as before. We would like the size of set $C$ to be $k$ or very close to $k$. We require this closeness to $k$ for the size of set $C$ in order to avoid instability in the subsequent spread of information when concept $c_C$ is used for the learning of other new concepts. If we think the neurons as vertices of a graph, we can directly translate this neuron allocation problem into graph theory. Other properties of the neural network in that model are highlighted in the following example. Let us attempt to learn concept $c_G = c_A \wedge c_B \wedge c_D \wedge c_E$, where each of the sets $A$, $B$, $D$, and $E$ has size $k$, by first learning concepts $c_C$ and $c_F$, where $c_C = c_A \wedge c_B$ and $c_F = c_D \wedge c_E$, and then learning $c_G = c_C \wedge c_F$. We require that sets $C$, $F$, and $G$ have sizes as close to $k$ as possible. If the sizes of sets $C$ and $F$ deviate significantly from $k$, then the deviation from $k$ of the size of $G$ may be even greater.

We can formulate the graph theory problem in which we are interested more formally as follows. Let $G = (V, E)$ denote an undirected connected graph on a set of $n$ vertices $V$ and a set of edges $E$. Let $k$ be a constant. Let set $N^*(X, Y)$, where $X, Y \subseteq V$, define the set of vertices that do not belong to $X, Y$ which are adjacent to at least one vertex in $X$ and at least one vertex in $Y$, in other words,

$$N^*(X,Y) = \{v \in V - (X \cup Y) \mid \exists x \in X, y \in Y \text{ such that } (v,x),\ (v,y) \in E\}.$$

**Problem:** *For a constant $k$, we would like to obtain an undirected graph $G = (V, E)$ that exhibits the following connectivity property:*

*For all disjoint sets $A, B \subseteq V$ such that $|A| = |B| = k$, we would like the cardinality of set $C = N^*(A, B)$ to be $k$ or as close to $k$ as possible. Also it is preferable for the graph to have approximately uniform degree.*

We are interested in the behavior of any construction as $n$ tends asymptotically to infinity. We would also like the graph to have uniform degree of about $\sqrt{n}$ or $\sqrt{n/k}$. One reason for this is that in the human brain each one of the approximately $10^{10}$ neurons in cortex is connected to between $10^4$ and $10^5$ others.

We shall prove in this thesis various results related to this problem. We shall first show that for $k > 1$, if any graph $G$ exists so that for all choices of disjoint $A$ and $B$ in the definition of the problem, the set $C = N^*(A, B)$ has size exactly $k$, then graph $G$ must have exactly $3k$ vertices. We go on to present a graph construction based on a graph constructed by Erdos and Renyi which approximately solves the problem, in the sense that for any two sets $A$ and $B$ the size of set $C = N^*(A, B)$ is in the range $[0, k^2]$. We also suggest some alternatives to this construction that yield better results.

The model $G_{n,p}$, $0 \leq p \leq 1$, of random graphs consists of all undirected graphs on $n$ labeled vertices in which the edges are chosen independently and with probability $p$.

We go on to investigate the properties of the class $G_{n,1/(k \cdot n)^{1/2}}$ of random graphs with respect to this problem since for such graphs, as pointed out by Valiant in [50], for random choices of disjoint $A$ and $B$ of size $k$ each, the mean and the variance of the size of set $C = N^*(A, B)$ are both approximately $k$ for $n \to \infty$. We study the properties of such graphs for an iterative process that models the way learning tasks are to be performed. We examine various cases corresponding to different values of the parameter $k$. We conclude this investigation by comparing these theoretical results to ones derived through computer simulations of random graphs.

## 1.3 Broadcasting

A task that needs to be performed frequently in communication networks is that of *broadcasting*, that is, the dissemination of information, held initially at one vertex, to all the vertices of the network. We may view a computer network as a graph, where the vertices correspond to processing units and the edges to communication lines linking these units.

A processing unit can only directly communicate with the ones directly linked to it. Each communication link is bi-directional but at any time only one of the two possible directions can be used. The message to be broadcast traverses such links in unit time. If one unit sends or receives a message from a second unit no other unit can send or receive messages from any of these two units during that communication.

In a complete graph $K_n$ on $n$ vertices it is possible to broadcast a message held by any vertex of the graph in $\lceil \lg n \rceil$ time steps to the remaining vertices by doubling at each step the number of vertices holding the message. Symbol $\lg n$ denotes the base two logarithm of $n$. At the first step of such a broadcast the vertex that originally possesses the message transmits it to a vertex adjacent to it. In the following step the two vertices that now possess the message transmit it to two new vertices. This way, after two time steps the original message has been transmitted to four vertices. This can be repeated until the message has been disseminated to the whole network.

We are interested in the minimum time needed to broadcast a message from a given vertex to all the vertices of $G_{n,p}$ random graphs. We are also interested in the *broadcast number* of such graphs that is, the maximum over all vertices $v$ of the minimum time needed to broadcast a message from vertex $v$.

The problem of broadcasting in random graphs was first studied by E. R. Scheinerman and J. C. Wierman in [46], where they proved that it was possible to perform with high probability, that is with probability tending to one as $n$ approaches infinity, broadcasting in $G_{n,p}$ random graphs for $p = \omega_n \log n/n$ with $\omega_n \to \infty$ as $n \to \infty$, in $\lg n + o(\lg n)$ time steps. In [46] it was conjectured that it was possible to broadcast a message in random graphs with edge probability $p = c \log n/n$, for any constant $c > 1$, in $\lg n + o(\lg n)$ time steps with high probability. We prove this conjecture for $c > 18.4$ and for broadcast time $\lg n + d\,\mathrm{lglg}n$, where $d$ is a positive constant less that one. We then improve this result to time $\lg n + f\,\mathrm{lglglg}\,n$, where $f$ is a positive constant less than one.

It was also proved in [46] that for some broadcast algorithm, it was possible, with high probability, to *optimally broadcast* a message from any given vertex to all the vertices of $G_{n,p}$ random graphs with edge probability $p = c\log^2 n/n$, for any constant $c$ appropriately chosen, in exactly $\lceil \lg n \rceil$ time steps. This implies a broadcast number of $\lceil \lg n \rceil$ for almost all such graphs. It was also conjectured in [46] that it is possible to optimally broadcast a message, with high probability, in $\lceil \lg n \rceil$ time steps in random graphs with edge probability $p = \omega_n \log n/n$, where $\omega_n \to \infty$ as $n \to \infty$. We show here for $p = c\,\mathrm{lglg}n \log n/n$, $c > 16$, that this is feasible. We also show how broadcasting in at most $\lceil \lg n \rceil + 1$ steps can be achieved in $G_{n,p}$ random graphs with $p = c\,\mathrm{lglglg}\,n \log n/n$, $c > 16$. Hence by adding at most one more time step in the minimum required broadcast time we manage to further decrease the necessary value of $p$.

## 1.4 Bulk-synchronous parallel computation

The bulk-synchronous parallel (BSP) model was suggested as a possible "bridging" model to link the software and hardware worlds with respect to parallel computation. Two modes of programming were envisaged in [52]: *automatic mode* where programs are written, say PRAM style, in a high level language that hides memory distribution from the user, and *direct mode* where the programmer retains control of memory allocation.

The bulk-synchronous parallel or BSP model as described in [52] consists of three parts:
(i) a number of processor/memory components. Here we will assume that each component consists of a sequential processor with a block of local memory.
(ii) a router that can deliver messages point to point among the components, and
(iii) facilities for globally synchronizing, in barrier style, all or a subset of the components.

The parameters of such a machine are $p$, the number of processor/components, $L$, the minimal time between successive synchronization operations, and $g$ the ratio of the total throughput of the whole system in terms of basic local computational operations to the throughput of the router in terms of words of information delivered.

Computation on this model proceeds in a succession of supersteps. In each superstep each processor is given a task that can be executed using data already available there locally before the start of the superstep.

This work is mainly concerned with the direct mode of programming since there are circumstances in which the effort of programming in direct mode, as opposed to automatic mode, is warranted. It is advantageous, for example, to program in direct mode when small multiplicative constant factors in runtime are important or smaller problem instances can be run efficiently. Machines with a high $g$ (automatic mode generally requires $g$ to be close to unity) or machines whose parameter $L$ is sufficiently high for direct, but not for automatic mode, for the problem instance in hand, provide two further circumstances in which direct mode programming may be beneficial.

We shall therefore emphasize multiplicative constant factors and in this work we shall restrict ourselves to *one-optimal* algorithms (i.e. where this factor is one asymptotically as the problem size $n \to \infty$). Although it is difficult to measure complexity to this accuracy, since the operations that are counted have to be carefully defined, we can make such measures meaningful by measuring ratios between runtimes on pairs of models that have the same set of local instructions. Thus we shall say that a BSP algorithm is *one-optimal in computation* if it performs the same number of operations, to a multiplicative factor of $1 + o(1)$ as $n \to \infty$, as a specified corresponding sequential or say, PRAM algorithm. Insistence on one-optimality leads to algorithms that only work in restricted ranges of $n$ and $L$ in terms of $p$.

The development and analysis of direct BSP algorithms has a secondary motivation, one relating to computer architecture. An interesting question is that of determining whether, if BSP computers are constructed in currently foreseeable technologies, it is advantageous to implement global operations such as broadcasting or the parallel prefix in hardware. The complexities of some of the most frequently computed problems with respect to these various options need to be understood before such questions can be resolved.

We present in this work results related to simulations of two PRAM models on the BSP as well as simulations (that bypass the BSP) of these PRAM models on a model of optical computing. Direct algorithms are given for a variety of frequently used problems such as, vector matrix multiplication, polynomial evaluation, Gauss-Jordan elimination and Cholesky factorization. Variations of some of these algorithms on various augmented BSP models are also examined. Finally we present a class of direct BSP algorithms for a combinatorial problem, that of sorting.

# Chapter 2

# Some Results from Probability Theory

In this chapter we present some results from probability theory that we will use later in several places. Some further results that are specific to the discussion in just one of the chapters will be introduced when needed.

A function $f(n)$ is $o(g(n))$ and we write $f(n) = o(g(n))$, if $f(n)/g(n) \to 0$ as $n \to \infty$.

A function $f(n)$ is $\omega(g(n))$ and we write $f(n) = \omega(g(n))$, if $f(n)/g(n) \to \infty$ as $n \to \infty$.

A function $f(n)$ is $O(g(n))$ and we write $f(n) = O(g(n))$, if there exist constants $c$ and $n_0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$.

A function $f(n)$ is $\Omega(g(n))$ and we write $f(n) = \Omega(g(n))$, if there exist constants $c$ and $n_0$ such that $f(n) \geq cg(n)$ for all $n \geq n_0$.

A function $f(n)$ is $\Theta(g(n))$ and we write $f(n) = \Theta(g(n))$, if there exist constants $c_1$, $c_2$, and $n_0$ such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.

Most of the Definitions and results to follow can be found in an introductory textbook on probability theory such as [17].

Let $X$ be a random variable and let $\mathcal{E}[X]$ be its expectation and $var(X) = \mathcal{E}[(X - \mathcal{E}[X])^2] = \mathcal{E}[X^2] - (\mathcal{E}[X])^2$ its variance.

**Theorem 1** *Let $X$ be a random variable and $g(\cdot)$ a nonnegative function with domain the real line; then*

$$Prob(g(X) \geq t) \leq \mathcal{E}[g(X)]/t \text{ for every } t > 0.$$

The special case where $g(X) = X$ is sometimes referred to as *Markov* inequality. Chebyshev's inequality then follows.

**Fact 1 (Chebyshev inequality)** *If $X$ is a random variable with finite variance,*

$$Prob(|X - \mathcal{E}[X]| \geq r\,\sqrt{var(X)}) \leq 1/r^2 \quad \text{for every } r > 0.$$

**Fact 2** *A random variable $X$ is defined to have a Binomial distribution if the discrete density function of $X$ is given by*

$$B(i; n, p) = \begin{cases} \binom{n}{i} p^i (1-p)^{n-i} & \text{for } i = 0, 1, \ldots, n \\ 0 & \text{otherwise} \end{cases},$$

*where $0 \leq p \leq 1$ and $n$ ranges over the positive integers.*

**Fact 3 ([17])** *Let $X$ be a random variable that follows a binomial distribution $B(i; n, p)$. Then, for the first two central moments of $X$ we have the following.*

$$\mathcal{E}[X] = np,$$

$$\mathcal{E}[X^2] = np + n(n-1)p^2.$$

*The variance of $X$ is given by*

$$var(X) = np(1-p).$$

**Fact 4** *A random variable $X$ is defined to have a Poisson distribution if the density function of $X$ is given by*

$$P(i, \lambda) = \begin{cases} \frac{e^{-\lambda}\lambda^i}{i!} & \text{for } i = 0, 1, \ldots \\ 0 & \text{otherwise} \end{cases},$$

*where $\lambda > 0$. The mean and the variance of $X$ are given below.*

$$\mathcal{E}[X] = var(X) = \lambda.$$

**Fact 5 (Normal distribution)** *A random variable $X$ is defined to be normally distributed if its density is given by*

$$f(x) = \phi_{\mu,\sigma^2} = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x-\mu)^2/(2\sigma^2)},$$

*where the parameters $\sigma$ and $\mu$ satisfy $-\infty < \mu < \infty$ and $\sigma > 0$. The random variable $X$ has mean and variance that are given by the following expressions.*

$$\mathcal{E}[X] = \mu, \quad var(X) = \sigma^2.$$

We use the notation $X \sim N(\mu, \sigma^2)$ to indicate that random variable $X$ follows a normal distribution with mean $\mu$ and variance $\sigma^2$. We also call $X$ a normal random variable. If $X$ has mean 0 and variance 1 it is called a *standard* or *normalized* normal random variable. Its density function is then $\phi(x)$ (with no indices). We also use the function $\Phi(x)$ to denote $\Phi(x) = \int_{-\infty}^{x} \phi(t)dt$.

The probability of having $S_{n,p}$ successes in $n$ independent Bernoulli trials of individual success probability $p$ is given by a binomial term $B(S_{n,p}; n, p)$. The discussion to follow is taken from [13]. The following is due to Hoeffding.

**Fact 6 (Hoeffding (1963))** *Let the random variables $X_1, \ldots, X_n$ be independent, with $0 \leq X_i \leq 1$, for each $i$. Let $\overline{X} = (1/n)\sum_i X_i$, $p = \mathcal{E}[\overline{X}]$, and $q = 1 - p$. Then for $0 \leq t < q$,*

$$Prob(\overline{X} - p \geq t) \quad \leq \quad \left( (\frac{p}{p+t})^{p+t} (\frac{q}{q-t})^{q-t} \right)^n.$$

The following was proven by Angluin and Valiant for the binomial case that is, when random variable $X_i$, for each $i$, is 1 with probability $p$ and 0 with probability $1 - q$.

**Fact 7 (Angluin and Valiant [4])** *In a binomial distribution with density $B(S_{n,p}; n, p)$ the following bounds hold*

$$Prob(S_{n,p} \geq (1+\epsilon) \cdot n \cdot p) \leq e^{-\frac{1}{3} \cdot \epsilon^2 \cdot n \cdot p},$$

*and*

$$Prob(S_{n,p} \leq (1-\epsilon) \cdot n \cdot p) \leq e^{-\frac{1}{2} \cdot \epsilon^2 \cdot n \cdot p},$$

*where* $0 < \epsilon < 1$.

The following generalization was presented in [13].

**Proposition 1** *Let the random variables $X_1, \ldots, X_n$ be independent, with $0 \leq X_i \leq 1$, for each $i$. Let $\overline{X} = (1/n) \sum_i X_i$, and let $p = \mathcal{E}[\overline{X}]$. For $0 < \epsilon < 1$ we have:*

$$Prob(\overline{X} \geq (1+\epsilon) \cdot p) \leq \exp(-\frac{1}{3} \cdot \epsilon^2 \cdot n \cdot p).$$

$$Prob(\overline{X} \leq (1-\epsilon) \cdot p) \leq \exp(-\frac{1}{2} \cdot \epsilon^2 \cdot n \cdot p).$$

We shall also use the following asymptotic result derived from the DeMoivre-Laplace Theorem given in [10] (Theorem 7, part (ii)).

**Theorem 2 ([10], Theorem 7(ii))** *Let $S_{n,p}$ denote the number of successes in $n$ independent Bernoulli trials with individual probability of success $p$ and let $q = 1 - p$. If $uq \geq 2$ and $pn \geq 1$ then*

$$Prob(S_{n,p} \geq upn) < (e/u)^{upn}.$$

We shall study various properties of various classes of random graphs. A definition of the model of random graphs that will be used is given below.

**Definition 1** *The model $G_{n,p}$ , $0 \leq p \leq 1$, of random graphs consists of all undirected graphs on $n$ labeled vertices in which the edges are chosen independently and with probability $p$.*

# Chapter 3

# A Graph Theoretic Problem

## 3.1 The problem - Motivation.

We deal here with the following graph construction problem posed by L. G. Valiant in [50] and also discussed in the introduction. Let $G = (V, E)$ denote an undirected connected graph on a set of $n$ vertices $V$ and a set of edges $E$. Let $k$ be a constant. Let set $N^*(X, Y)$, where $X, Y \subseteq V$, define the set of vertices that do not belong to $X, Y$ which are adjacent to at least one vertex in $X$ and at least one vertex in $Y$. In other words

$$N^*(X, Y) = \{v \in V - (X \cup Y) \mid \exists x \in X, y \in Y \text{ such that } \ (v, x),\ (v, y) \in E\}.$$

**Problem:** *For a constant $k$, we would like to obtain an undirected graph $G = (V, E)$ that exhibits the following connectivity property:*

*For all disjoint sets $A, B \subseteq V$ such that $|A| = |B| = k$, we would like the cardinality of set $C = N^*(A, B)$ to be $k$ or as close to $k$ as possible. Also it is preferable for the graph to have approximately uniform degree.*

We are interested in the behavior of constructions as $n$ tends asymptotically to infinity. We shall prove the negative result that for $k > 1$, if any graph $G$ exists so that for all choices of $A$ and $B$ in the definition of the problem, the set $C = N^*(A, B)$ has size exactly $k$, then

$G$ must have exactly $3k$ vertices. We then go on to present a graph construction based on a projective plane graph of Erdos and Renyi which in a certain sense explained in section 3.7, solves the given problem approximately. We finally examine the connectivity properties of random graphs with respect to this problem since in these graphs, as pointed out in [50], for random choices of disjoint $A$ and $B$ of size $k$ each, the mean and the variance of the size of set $C$ are both approximately $k$ for $n \to \infty$.

## 3.2 Definitions

We present below the following definitions for distinct vertices $a$, $b$ and $c$ and disjoint sets of vertices $A$, $B$ and $C$ of an undirected graph $G$. Let $k$ be a positive integer.

**Definition 2** *A vertex* a *is defined to be* common *to vertices* b *and* c *if* a *is adjacent to both* b *and* c*.*

Similarly, a vertex $a$ is common to two sets of vertices $B$ and $C$ if there exist vertices $b \in B$, and $c \in C$ such that $a$ is common to $b$ and $c$. Also, a set $A$ is common to two sets of vertices $B$ and $C$, if for every vertex $a \in A$, vertex $a$ is common to $B$ and $C$.

The notion of "commonness" in this treatment is related to the adjacency properties of a vertex or a set of vertices rather than to the intersection of sets of vertices.

**Definition 3** *A graph $G = (V, E)$ has* **connectivity property** *$P^k$ if for every two disjoint sets $A$, $B$ of $G$ of $k$ vertices each, set $\{a \mid a \in V$ is common to $A$ and $B$; $a \notin A$; $a \notin B\}$ is of size $k$.*

We will usually refer to graphs like $G$ that have property $P^k$ as $G_P^k$ graphs.

**Definition 4** *A $G_P^k$ graph is called* **trivial** *if it has exactly $3k$ vertices.*

**Definition 5** *For a specified positive number $a > 1$, a set* A *is* "$a$-bad" *if it is of size zero or of size at least $ak$.*

The term "edge-choice" in the context of a random graph will refer to the decision as to whether a particular edge, among the $n(n-1)/2$ prospective ones, is to be included or not in the graph. All results in this chapter refer to the class of $G_{n,p}$ random graphs, where $p = 1/(kn)^{1/2}$. When we claim that a property holds with high probability we mean that this probability tends to one as $n \to \infty$. In other words, the probability that this property fails to hold is bounded above by a function $\epsilon(n) \to 0$ as $n \to \infty$.

The results in this chapter are asymptotic in the limit $n \to \infty$. We examine first the case where the parameter $k$ is a constant. Later we examine cases that hold asymptotically for large $k$. We shall distinguish clearly between these two kinds of results. All logarithms in this chapter are to base $e$, unless otherwise stated. When we use the approximation $\approx$ symbol, we mean that the ratio of the quantities on the two sides of this symbol tends to one as $n \to \infty$. We shall similarly use the symbol $\preceq$ to denote that the ratio of the left hand side to the right hand side tends to a quantity at most one as $n \to \infty$. In section 3.13 in addition to having $n \to \infty$, we also take $k$ to be large and the use of the $\approx$ symbol takes into account this "largeness" of $k$.

## 3.3 Previous Work on the Problem.

We now examine in finer detail the problem described in section 3.1. The $k = 1$ case of the problem is well studied. It is related to the existence of projective planes, certain combinatorial structures called designs, and, weakly, to a class of graphs called strongly regular graphs. One can find definitions for these constructs in [28]. We shall not get into more details here. For $k = 1$ if we restrict the graph we would like to construct to be regular, that is all its vertices have the same degree, then we can deduce using the properties of strongly regular graphs shown in [28] that the triangular graph, the complete graph on three vertices, is the only regular graph that satisfies property $P^k$ for $k = 1$. Furthermore, Erdos, Renyi and Sos, in [16], proved that for $k = 1$, the only solution of the construction problem exists for odd $n$ and is the windmill-like graph shown in Figure 3-1, where one vertex has degree $n-1$ and each of the remaining $n-1$ vertices has degree two. The disadvantage of this graph is the high

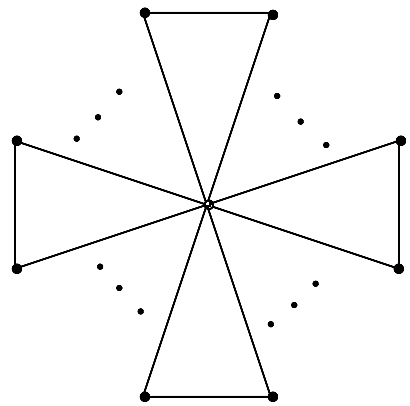

Figure 3-1: The Erdos-Renyi Graph (k=1).

degree of one of its vertices. A secondary problem is that it cannot be used, in the context of neural networks, to store sequences of conjunctions in any generality, since the concept stored in the degree $n-1$ neuron must participate in every member of the sequence. If this neuron is once assigned a concept it cannot be assigned again.

These results imply that a construction of a graph that satisfies property $P^k$ for $k > 1$ and has uniform or near uniform degree may be difficult or impossible. They also raise the question whether there exists a theorem similar to the one stated by Erdos, Renyi and Sos that proves the existence of special graphs that do have property $P^k$ for $k > 1$. A quadrilateral-free graph (i.e. a graph with no cycle of length four) of diameter two and degree $O(\sqrt{n})$ which was constructed by Erdos and Renyi using projective planes (see [16] or section 3.7 for a construction of this graph) has the following property. Any two vertices of the graph are either adjacent or there exists exactly one vertex adjacent to both of them. This graph approximately solves the $k = 1$ case, in the sense that for arbitrary choices of two distinct vertices $a$ and $b$, either there exists a vertex $c$ distinct from $a$ and $b$ which is connected to both $a$ and $b$, or vertices $a$ and $b$ are adjacent. This graph has uniform degree $O(\sqrt{n})$.

## 3.4 Results

We first prove that for any $k > 1$ if a graph exists that has property $P^k$ then the graph has to be trivial. The proof consists of two Propositions. The first, Proposition 6, shows that in a graph with property $P^k$, $k > 1$, unless the graph is trivial, any vertex has degree either one or at least $2k$. The second, Proposition 7, states that a vertex in a $G_P^k$ graph can not have degree at least $2k$ unless the graph is trivial. The only case left, other than having a trivial $G_P^k$ graph, is that every vertex has degree exactly one, which also contradicts property $P^k$. This non-existence result shows that there is no exact solution to our problem for graphs with $n$ vertices where $n > 3k$. We conclude that the best we can aim for in general is to have graphs in which two disjoint sets of vertices, each of cardinality $k$, have in common, not $k$, but $l$ vertices where $l$ is close to $k$.

With this motivation we go on to give an explicit construction based on the projective plane graph of Erdos and Renyi which satisfies this latter requirement. We claim that the properties of this graph with respect to the graph problem examined are slightly more tolerable than those of random graphs.

We also examine in this chapter the problem presented in the introduction in the context of random graphs. For a given replication factor $k$ we select the class of random graphs $G_{n,1/(k \cdot n)^{1/2}}$ proposed by Valiant and examine their properties as $k$ varies. The case of constant $k$ is discussed first, while asymptotic results for large $k$ are examined in two of the last sections of this chapter. Experimental results obtained through simulations are compared to the derived theoretical results.

In the remainder of this section we shall define and explain the properties of these random graphs that we investigate. The properties all relate to the following iterative procedure that is motivated by *sequences* of, rather than *single*, memory assignments in the neural network model. Let us pick uniformly at random $k$ vertices from this graph to form set $A$ and similarly, another set of $k$ vertices distinct from the vertices of the first set to form set $B$. Valiant in [50] mentioned that for such a random graph the expected value and variance of the size of the set of vertices common to sets $A$ and $B$ so chosen, for large $n$, are both approximately equal to $k$.

We present below an iterative procedure that defines how a set of vertices of the random graph is formed at iteration $r \geq 0$.

**Definition 6** *The following iterative procedure forms a set of vertices at iteration $r \geq 0$.*

- *At iteration 0, a set of vertices is formed by choosing arbitrarily at random a subset of size $k$ among the vertices of the graph not previously selected. We need to form $2^r$ such sets.*

- *A set of vertices is formed at iteration $i$ by including in this set the vertices common to two sets, not previously selected this way, formed at iteration $i-1$. The number of sets formed this way at iteration $i$ is $2^{r-i}$.*

We investigate for any constant $r$ the distribution of the size of the set of vertices formed at this iteration. We can relax the constraint of constant $r$ by assuming that $r$ depends on $k$ only and not on $n$. A set formed at iteration $r$ can be viewed as the root of a complete binary tree of depth $r$, where the vertices of the tree correspond to sets of vertices of the graph and each set formed at some iteration $i$, $1 \leq i \leq r$, is common to two sets formed at iteration $i-1$. There are also $2^r$ leaves that correspond to sets of size $k$ formed at iteration zero. We call such a tree, *a tree of iteration $r$*. In Figure 3-2, a tree of iteration three is depicted.

Proposition 2 shows that for a constant $r$ the mean of the random variable that represents the size of the set formed at iteration $r$ is approximately $k$ for $n \to \infty$. We show this proposition in the following more general form. Let $A$, $B$ be two sets formed at iteration $r-1$ whose sizes are random variables with means $\mu_1$, $\mu_2$ respectively. Then $Y_r$, the size of set $C$ common to $A$ and $B$, has mean approximately equal to $\mu_1\mu_2/k$. This is approximately $k$.

**Proposition 2** *Consider the class of $G_{n,1/(k\cdot n)^{1/2}}$ random graphs, where $k$ is constant. Let the random variables that represent the sizes of two sets $A$, $B$ formed at iteration $r-1 \geq 0$ have expectations $\mu_1$, and $\mu_2(=\mu_1)$ respectively. Then the expectation of the random variable $Y_r$, for constant $r$, that represents the size of the set of vertices $C$ common to $A$ and $B$ is*

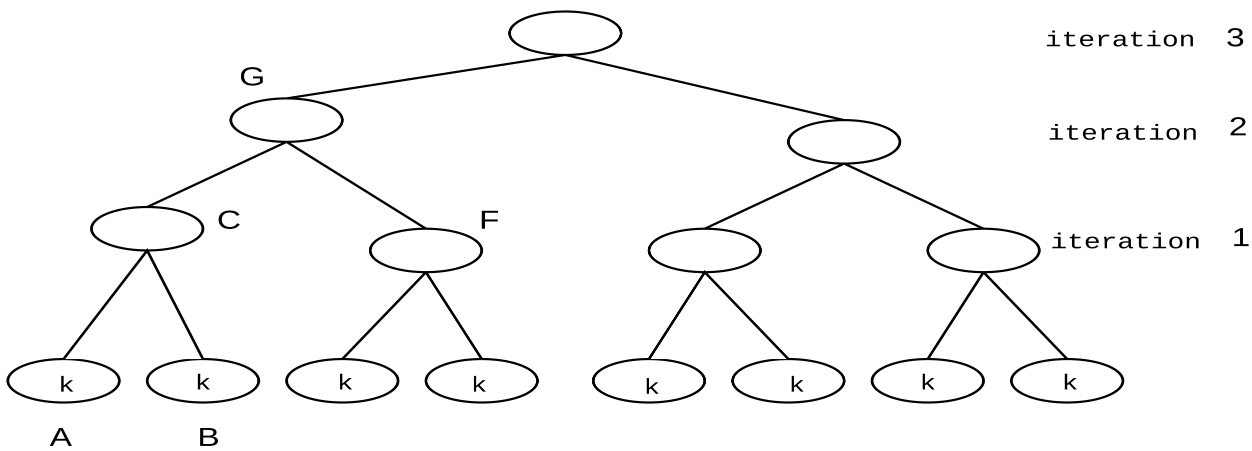


Figure 3-2: A Tree of iteration three.

*asymptotically as* $n \to \infty$

$$\mathcal{E}[Y_r] \approx \frac{\mathcal{E}[Y_{r-1}]\mathcal{E}[Y_{r-1}]}{k} = \frac{\mu_1\mu_2}{k} = \frac{\mu_1^2}{k}.$$

*Consequently,* $\mathcal{E}[Y_r] \approx k$ *for any constant* $r \geq 1$.

Proposition 3 shows a weak corresponding result for the variance, in the form of an upper bound only. A tight bound, as expressed by Proposition 4, will be deduced later using Theorem 8 in section 3.11.

**Proposition 3** *Consider the class of* $G_{n,1/(k\cdot n)^{1/2}}$ *random graphs, where* $k$ *is constant. The variance* $var(Y_r)$ *of the size of set* $C$ *common to two sets* $A$, $B$, *formed at iteration* $r-1 \geq 0$ *whose sizes have means and variances* $\mu_1, v_1$, *and* $\mu_2, v_2$ *respectively, is bounded above as* $n \to \infty$ *by*

$$var(Y_r) = \mathcal{E}[Y_r^2] - \mathcal{E}^2[Y_r] \preceq \frac{\mu_1\mu_2}{k} + \frac{4v_1v_2}{k^2} + \frac{2{\mu_1}^2v_2 + 2{\mu_2}^2v_1}{k^2}.$$

*Consequently, as $n \to \infty$,*

$$var(Y_r) \preceq k + \frac{4v^2}{k^2} + 4v.$$

**Proposition 4** *Consider the class of $G_{n,1/(k \cdot n)^{1/2}}$ random graphs, where $k$ is constant. The variance $var(Y_r)$ of the size of set $C$ common to two sets $A$, $B$ formed at iteration $r-1$, whose sizes have means and variances $\mu_1, v_1$, and $\mu_2, v_2$ respectively, is given by*

$$var(Y_r) = \mathcal{E}[Y_r^2] - \mathcal{E}^2[Y_r] \approx \frac{\mu_1 \mu_2}{k} + \frac{v_1 v_2}{k^2} + \frac{{\mu_1}^2 v_2 + {\mu_2}^2 v_1}{k^2}.$$

*Consequently,*

$$var(Y_r) \approx k + 2v + v^2/k^2.$$

In section 3.15 we describe the results of computer simulation experiments that confirm Propositions 2 and 4 to high accuracy.

In section 3.9, we show how one can use Propositions 2 and 3 to get some rough estimates on the probability that the set $C$ formed at iteration $r$ is $a$-bad. These bounds are direct applications of Markov's and Cantelli's inequalities.

In section 3.10 we obtain for iterations one and two, much more accurate bounds than these. In particular accurate bounds on the probability of having a set of size zero or more than $ek$ at iteration one, and of size zero or more than $e^2k$ at iteration two are derived. In deriving these bounds we need to solve numerically a number of equations. The accuracy of the numerical approximations affects the accuracy of the bounds obtained.

**Theorem 3** *In the class of $G_{n,1/(k \cdot n)^{1/2}}$ random graphs, where $k$ is constant, the probabilities of having a set of vertices of size $Y_1 = 0$ or $Y_1 > ek$ at iteration one are, in the limit $n \to \infty$*

$$Prob(Y_1 = 0) \approx e^{-k},$$

*and*

$$Prob(Y_1 \geq ek) \preceq \frac{e}{e-1} \frac{e^{-k}}{\sqrt{2\pi ek}}.$$

The exponents of $e$ that appear in the upper bounds in the Theorems to follow are numerical approximations to solutions of various equations that do not depend on $n$. We indicate this fact in the statements of the following two theorems by using an $\tilde{O}(x)$ notation to indicate that the value of $x$ given was obtained by making numerical approximations, and better approximations may be obtained by solving to high precision. For the same reason we use the $O_k$ notation to absorb various constants that are attributed to such numerical approximations. Thus $f(k) = O_k(g(k,n))$ if there exist constants $c$, $k_0$ and $n_0$ such that $f(k) \leq cg(k,n)$ for any $k \geq k_0$ and any $n \geq n_0$.

**Theorem 4** *In the class of* $G_{n,1/(k\cdot n)^{1/2}}$ *random graphs, where* $k$ *is constant, the probability of having a set of vertices at iteration two of size zero is,*

$$O_k\left(k^{1/2}\right)\ e^{-\tilde{O}(0.544)\ k} + O_k\left(k^{1/2}\right)\ e^{-\tilde{O}(0.632)\ k} + 2e^{-k}.$$

**Theorem 5** *In the class of* $G_{n,1/(k\cdot n)^{1/2}}$ *random graphs, where* $k$ *is constant,*

$$\begin{aligned} Pr(Y_2 > e^2 k) &\preceq O_k\left(k\ \log k\right)\ e^{-\tilde{O}(1.50512)\ k} + O_k\left(k^{1/2}\ \log k\right)\ e^{-\tilde{O}(2.10)\ k} \\ &+ \frac{2e^{-2k}}{\pi e k} + \frac{4e^{-k}}{\sqrt{2\pi e k}}, \end{aligned}$$

*where* $Y_2$ *is a random variable that represents the size of a set of vertices at iteration two.*

The bounds on the tails of the Binomial Distribution due to Angluin and Valiant presented in Chapter 2 are used in section 3.11 to derive bounds on the tails of random variable $Y_r$. These results are summarized in Theorems 6 and 7. For small values of $k$ the bounds obtained in these results are not very useful. If we use the proof technique that was used in the derivation of these bounds and claim the DeMoivre-Laplace bound presented in Chapter 2 instead of the bounds on the right tail of the binomial distribution, then we get the bound expressed in Theorem 8 below. We can then use this result to show Proposition 4.

**Theorem 6** *In the class of* $G_{n,1/(k\cdot n)^{1/2}}$ *random graphs, where* $k$ *is constant, a set formed at some constant iteration* $r$ *has size at most* $(1-a)^{2^r-1}\cdot k$, $0 < a < 1$, *with probability, in the* *limit $n \to \infty$, at most*

$$\sum_{\lambda=0}^{r-1} 2^{\lambda} e^{-\frac{1}{2}a^2(1-a)^{2^{r-\lambda}-2}k}.$$

**Theorem 7** *In the class of $G_{n,1/(k\cdot n)^{1/2}}$ random graphs, where $k$ is constant, a set formed at some constant iteration $r$ has size at least $(1+a)^{2^r-1}\cdot k$, $0<a<1$, with probability, in the limit $n \to \infty$, at most*

$$\sum_{\lambda=0}^{r-1} 2^{\lambda} e^{-\frac{1}{3}a^2(1+a)^{2^{r-\lambda}-2}k}.$$

**Theorem 8** *In the class of $G_{n,1/(k\cdot n)^{1/2}}$ random graphs, where $k$ is constant, a set formed at some constant iteration $r$ has size at least $u^{2^r-1}\cdot k$, with probability, in the limit $n \to \infty$, at most*

$$\sum_{\lambda=0}^{r-1} 2^{\lambda} (e/u)^{u^{2^{r-\lambda}-1}k},$$

*where $u(1-\frac{u^{2^l-2}k}{n}) > 2$ and $\frac{u^{2^l-2}k}{n}n > 1$, for any $l$ such that $1 \le l \le r$.*

Sections 3.8 through 3.11 hold for constant $k$ and $n \to \infty$. In sections 3.12, 3.13 we examine the case where in addition to having $n \to \infty$, $k$ is also large but independent of $n$. We are then able to show, through the use of exponential generating functions, that the size of a set formed at some constant iteration $r$ can be approximated by a Normal random variable with mean $k$ and variance approximately $(2^r-1)k+O(1)$. More precisely, the value of the variance is that obtained, though through other techniques, in Proposition 4. In order to achieve this result we develop a new way to examine the variance of the size of set $C$ formed at some constant iteration $r$ and common to two sets $A$, $B$ formed at iteration $r-1$. We first approximate the product $|A||B|$ by a normal random variable and then express the probability of having set $C$ of some size with respect to this newly introduced random variable.

**Theorem 9** *If $X_1, X_2 \sim N(\mu,\sigma^2)$ are independent, and $\mu=\Theta(k)$ and $\sigma^2=\Theta(k)$ for some* *parameter $k$ and in the limit for large $k$, then the random variable $X = X_1 \cdot X_2$ follows a normal distribution $N(\mu^2, \sigma^4 + 2\,\mu^2\,\sigma^2)$.*

**Theorem 10** *Consider the class of $G_{n,1/(k\cdot n)^{1/2}}$ random graphs, where $k$ is independent of $n$. At some constant iteration $r$ the random variable $Y_r$ that represents the size of a set of vertices formed at iteration $r$ follows a normal distribution , $N(k, (2^r - 1)k + O(1))$, for large $k$ and $n \to \infty$.*

Section 3.14, describes the experimental results obtained through random graph simulations for $k = 25, 50$. For iterations $r = 1, \ldots, 5$, we give the statistical mean and variance of the size of set $C$ formed at iteration $r$ during the simulations. We also present for these iterations the relative frequencies of sets having size 0, size less than $k/2$, greater than $2k$, greater than $3k$, as well as fractions of the sets formed at each iteration whose sizes were within $\pm\sqrt{k}$, $\pm 2\sqrt{k}$, $\pm 3\sqrt{k}$ of the mean.

## 3.5 A Negative result for the $k > 1$ case

Let us fix a $k > 1$. Let us assume that a $G_P^k$ graph exists and is not trivial. Let the number of vertices of this graph be $n$. The following is immediate:

**Proposition 5** *Any two vertices of $G_P^k$ can not have more than $k$ vertices in common.*

**Proof:** If two such vertices exist we can obviously find two sets $A$ and $B$ of $k$ vertices each, that have at least $k+1$ vertices in common. The only case in which we cannot find such sets $A$ and $B$ is when $G_P^k$ has $3k$ vertices. ■

**Proposition 6** *If there exists a vertex $v$ in a $G_P^k$ graph that is not trivial such that the degree of $v$ is at most $2k - 1$, then the degree of $v$ is equal to one.*

**Proof:** Suppose that there exists a vertex of degree at most $2k - 1$ and let us call it $v$. We shall call a vertex $a$ a neighbor of $b$ if $a$ is adjacent to $b$. The vertices adjacent to $v$ and vertex $v$ are thus at most $2k$ in number. We can split these vertices into two sets and let us call them $A$ and $B$. Each of the two sets will be of size at most $k$, will contain at least one neighbor of $v$, and all neighbors of $v$ including vertex $v$ itself will be in one of these two sets. This is possible whenever $v$ has at least two neighbors, otherwise $v$ is of degree one and we are done.

Without loss of generality let $A$ be the set containing $v$. If the neighbors of $v$ are less than $2k-1$, then for the cardinalities of each of the sets $A$ and $B$ to be $k$ we need to add vertices to these two sets from the remaining vertices of the graph. We consider the consequences of property $P^k$ for sets $A$ and $B$ of $G_P^k$. Let $C$ be the set of $k$ vertices common to $A$ and $B$. There exists a vertex and let it be $x$ that does not belong to either set $A$ or $B$ or $C$, or else graph $G_P^k$ is trivial. Note that $v$ cannot be connected to a vertex in $C$ because all neighbors of $v$ are in $A$ or $B$. Hence we can remove $v$ from $A$ and sets $A - \{v\}$ and $B$ still have $C$ as the set of their common vertices. We thus replace $v$ in $A$ by $x$ and claim property $P^k$ for sets $A - \{v\} \cup \{x\}$ and $B$. They have in common not only set $C$ but also $v$. This contradicts the $P^k$ property. ■

**Proposition 7** *If there exists in $G_P^k$ a vertex of degree at least $2k$, then $G_P^k$ must be trivial.*

**Proof:** Let $v$ be a vertex of degree at least $2k$ and let us take $2k$ of these vertices adjacent to $v$ and split them into two sets $A$ and $B$ of size $k$ each. We then consider the consequences of property $P^k$ for these two sets. These sets must have $v$ and $k-1$ other vertices in common. Let us call the set of the $k-1$ vertices other than $v$ set $C$. Let $A' \subseteq A$, $B' \subseteq B$ be all the vertices in $A$ and $B$ respectively connected to at least one vertex in $C$. We distinguish the following two cases.

**Case 1:** $|A'| + |B'| > k$. Set $C$ and vertex $v$ have more than $k$ vertices in common, namely $A' \cup B'$. Unless $G_P^k$ has only $3k$ vertices we can form sets $C'$, $D'$, each of size $k$, such that $C \subseteq C'$ and $v \in D'$. Sets $C'$ and $D'$ will then have more than $k$ vertices in common. This would contradict property $P^k$ of $G_P^k$.

**Case 2:** $|A'| + |B'| \leq k$. Let us then form set $X$, where $X \subseteq (A - A') \cup (B - B')$, of size $k$ and let $X' = (A \cup B) - X$. We then consider the consequences of property $P^k$ for $X$ and $X'$.

By construction, these two sets cannot have a vertex in $C$ as a common vertex. Therefore, there must exist at least one vertex outside $A \cup B \cup C \cup \{v\}$ that is common to $X$ and $X'$. Let us call this vertex $z$.

We define vertex $x$ adjacent to $z$ to be of Type $A$ if $x \in A$, and to be of Type $B$ if $x \in B$.

Since $z$ is common to $X$ and $X'$ it must be connected to a vertex in $X$ and to a vertex in $X'$. These two vertices of $X$ and $X'$ must be of the same Type, otherwise the original sets $A$ and $B$ have in common $C \cup \{v\} \cup \{z\}$, a contradiction to property $P^k$ of $G_P^k$. In any case, $|B - B'| \geq 1$ and $|A - A'| \geq 1$. Let these two vertices be, without loss of generality, of Type $A$, and let us call them $a_1 \in X$ and $a_2 \in X'$. Let $b$ be an arbitrary vertex in $B - B'$. We then form sets $Y$ and $Y'$ as follows. Set $Y$ will be $Y = B \cup \{a_1\} - \{b\}$ and set $Y'$ will be $Y' = A - \{a_1\} \cup \{b\}$. Because of their choice, neither $a_1$ nor $b$ are connected to a vertex in $C$. We then consider the consequences of property $P^k$ for these two sets $Y$ and $Y'$. They have in common $C \cup \{v\} \cup \{z\}$ that is $k+1$ vertices in $G_P^k$, a contradiction to property $P^k$ of $G_P^k$.

This proves the validity of the Proposition. ■

We now prove,

**Theorem 11** *For $k > 1$, every $G_P^k$ graph has exactly $3k$ vertices.*

**Proof:** From Propositions 6 and 7, a $G_P^k$ graph is either trivial, or every vertex of it has degree equal to one. If every vertex has degree one, it cannot be the case that a vertex is connected to two other vertices. Hence the graph cannot have property $P^k$. Therefore a $G_P^k$ graph must be trivial and thus has $3k$ vertices. ■

## 3.6 Projective Spaces and Planes

We mention in this section without proof some introductory material related to finite projective spaces/planes based on discussions in [28], [5], and [6].

A geometry is a kind of incidence system whose undefined elements are *points* and *lines* and the fundamental relation is the incidence relation $P \in L$ read as "the point $P$ is on line $L$". We may also use the term "$L$ contains $P$". A finite geometry is one that contains a finite number of points. We shall use the symbols $A$, $B$, $C$ in this discussion to interchangeably denote points in projective spaces or sets of points or graph vertices. We shall clearly distinguish between the one or the other use of these symbols.

**Axioms for Projective Geometry**

**PG 1.** There is one and only one line containing two distinct points.

**PG 2.** If $A$, $B$, and $C$ are three points not on a line, and if $D \neq A$ is a point on the line through $A$ and $B$, and if $E \neq A$ is a point on the line through $A$ and $C$, then there is a point $F$ on a line with $D$ and $E$ and also on a line with $B$ and $C$.

**PG 3.** Every line contains at least three distinct points.

The *dimension* of a projective space, for which the axioms for projective geometry hold, can be defined recursively by defining that a point is a space of dimension zero, a line is a space of dimension one; and recursively, if $X_{l-1}$ is a space of dimension $l-1$, $P$ a point not contained in $X_{l-1}$, the set of all points in all lines $PB$, $B$ a point of $X_{l-1}$, is a space $X_l$ of dimension $l$.

For projective planes, that is projective spaces of dimension two, the previous axioms may be put in a simpler form:

- There is one and only one line containing two distinct point.
- There is one and only one point common to two distinct lines.
- There exist four points, no three of which are on a line.

Let $m \geq 2$ be an integer. In a projective plane $\pi$ any of the following properties implies the rest:

- One point is on exactly $m+1$ lines.
- One line contains exactly $m+1$ points.
- Every point is on exactly $m+1$ lines.
- Every line contains exactly $m+1$ points.
- There are exactly $m^2+m+1$ points in $\pi$.

- There are exactly $m^2 + m + 1$ lines in $\pi$.

A finite projective plane is then said to be of *order* $m$ if a line contains exactly $m + 1$ points.

The following theorem is due to Desargues.

**Definition 7 (Theorem of Desargues)** *A plane is Desarguesian if $O$, $A_1$, $B_1$, $C_1$, $A_2$, $B_2$, $C_2$ are distinct points and if $OA_1A_2$, $OB_1B_2$, $OC_1C_2$ are distinct lines, then $A_1B_1$ and $A_2B_2$ meet in a point $C_3$, $A_1C_1$ and $A_2C_2$ meet in a point $B_3$ and $B_1C_1$ and $B_2C_2$ meet in a point $A_3$, and the points $A_3$, $B_3$, $C_3$ lie on a line.*

**Theorem 12 (Introduction of coordinates)** *A Desarguesian plane $S_2$ or projective space $S_l, l \geq 3$, may be coordinated by a division ring $R$ so that a point $W$ is represented by the left scalar multiples of a non-zero $l + 1$ vector over $R$ so that:*

$$W = u(x_1, \ldots, x_{l+1}), \quad (x_1, \ldots, x_{l+1}) \neq (0, \ldots, 0).$$

Let $S_l$ be a projective geometry of dimension $l \geq 2$ over the finite field $F_q = GF(q)$, with $q = p^r$ elements, where $p$ is a prime. We may write $S_l = PG(l, q)$ and we may refer to $S_2$ as $\pi$ as well. The $(q^{l+1} - 1)$ vectors $(x_1, \ldots, x_{l+1}) \neq (0, \ldots, 0)$ , $x_i \in GF(q)$, fall into sets of $q - 1$ elements each, each such set representing a point. Consequently, $PG(l, q)$ contains $(q^{l+1} - 1)/(q - 1)$ points. The parameter $q$ is also known as the order of the geometry.

Projective planes are known to exist only for prime or prime power order.

If $S_l$, $l \geq 2$, is an abstract $l$-dimensional projective space, then we define a **polarity** in $S_l$ as a mapping $P$ associating with every point $A$ in $S_l$ a hyperplane $P(A)$ and with every hyperplane $h$ a point $P(h)$ in such a way that:

1. $A = P(P(A))$ for every point $A$, and $h = P(P(h))$ for every hyperplane $h$.

2. The point $A$ is on the hyperplane $h$ iff the hyperplane $P(A)$ passes through the point $P(h)$.

It is an immediate consequence of Property 1 above that polarities are 1-1 correspondences.

An easily proved lemma (see also [5], [6], [34] for a proof) is the following:

**Lemma 1** *If $P$ is a polarity of the projective plane $\pi$ and $A$, $B$ are different points in $\pi$ such that $A$ is on the line $P(A)$ and $B$ is on the line $P(B)$ then $A$ is not on $P(B)$ (nor is $B$ on $P(A)$).*

We will refer to A and $P(A)$ as the *pole* and the *polar line* (or simply polar) of A respectively.

A line is called *absolute* if it passes through its pole. The lemma above states that an absolute line carries one and only one absolute point, that is a point that lies on its polar line.

## 3.7 Construction

We first present below a construction due to Erdos and Renyi [16] related to graphs of diameter two that are quadrilateral free. We will later use this construction to construct graphs that have approximately our connectivity property in the sense that if in this graph we pick uniformly at random two disjoint sets $A$ and $B$ of size $k$ each then the size of the set $C$ common to $A$ and $B$ can be represented by a random variable whose expectation is $k$ and whose variance depends on $k$.

### 3.7.1 Erdos-Renyi Graph

The graph $G = (V, E)$ constructed by Erdos and Renyi is defined as follows. Set $V$ of $G$ consists of the $q^2 + q + 1$ points of $PG(2, q)$, where $q$ is an odd power of an odd prime. We thus have $|V| = n = q^2 + q + 1$. We restrict the choice of $q$ for reasons that will become apparent when we present our construction in the next section. We then define a polarity $P$ that maps a point $(a, b, c) \in PG(2, q)$ to the line $ax + by + cz = 0$ and vice-versa. Two vertices $A$ and $B$ in $G$ are joined by an edge if $A$ is on the polar of $B$. By the definition of a polarity, $B$ is then on the polar of $A$ also. A point of $PG(2, q)$ is on its polar line if and only if it is on the conic $N$, $N : x^2 + y^2 + z^2 = 0$. A line is tangent to the conic $N$ if the common intersection of the line and $N$ is exactly one point. We shall also use the term point when we refer to vertices of the graph $G$ below.

It is shown in [34] that the number of absolute points is exactly $q+1$ if $q$ is not the square of an integer. The choice of $q$ fulfills this condition. Each point on the conic is an absolute point since the tangent line to the conic at that point is the polar line of that point. This implies that two vertices on the conic are not joined by an edge in $G$ since an absolute line contains one and only one absolute point.

Let $A$ be a point on the conic $N$. Let $P(A)$ be its polar line that passes through $A$. Since $P(A)$ contains $q$ other points besides $A$ and since all these points are on the polar of A, these are the only vertices to which A is connected to in $G$. Therefore in $G$, points on the conic have degree $q$. Points not on the conic have degree $q + 1$ since the polar lines of non-conic points include $q + 1$ points that these non-conic points can be connected to.

The following was proven by Erdos and Renyi.

**Claim 1 (see [16])** *Graph $G = (V, E)$ as constructed above has diameter two.*

Hence in graph $G$ every pair of vertices is either adjacent or there exists a third vertex that is adjacent to each vertex of the pair, or both. It is then easy to show the following.

**Claim 2** *There are only $q^2 + q$ sets of two vertices $a$ and $b$, $a \neq b$, in $G$ that do not have a third vertex $c$, $c \neq a$ and $c \neq b$, adjacent to both $a$ and $b$.*

**Proof:** An absolute point does not have in $G$ any vertex in common with any of the points of its polar line, that is a total of $q$ sets of two vertices for every such absolute point. Since there are $q + 1$ distinct absolute points the claim follows. ■

Such sets of two vertices represent only a small fraction of the total number of such sets, actually $O(1/q^2) = O(1/n)$. For any other set of two vertices there always exists, by the construction of graph $G$, a third vertex adjacent to both vertices.

We then proceed to showing the following claim.

**Claim 3** *Two vertices in $G$ cannot have more than one vertex in common.*

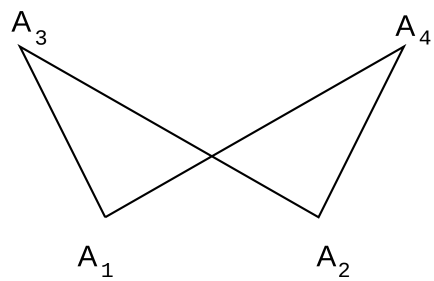


Figure 3-3: A configuration of four points.

**Proof:** (By contradiction). Suppose that vertices $A_1, A_2$ are distinct and have in common at least two vertices. Let $A_3, A_4$ be two of these common vertices. By way of Figure 3-3 we have the following incidence relationships:

$$\begin{aligned} A_3 \in P(A_1) &\quad A_1 \in P(A_3), \\ A_3 \in P(A_2) &\quad A_2 \in P(A_3), \\ A_4 \in P(A_1) &\quad A_1 \in P(A_4), \\ A_4 \in P(A_2) &\quad A_2 \in P(A_4). \end{aligned}$$

We then deduce that $A_3$ and $A_4$ have the same polar lines and are hence identical, contradicting the assumption that $A_3$ and $A_4$ are distinct. ■

### 3.7.2 Remarks

The discussion and the claims presented above lead to the following conclusions related to the Erdos-Renyi graph $G$ so constructed.

- Almost all pairs of vertices of graph $G$ have exactly one common vertex. The probability that two vertices of $G$, chosen uniformly at random among its $n$ vertices, have no common vertex is $O(1/n)$ by Claim 2.

- The number of vertices common to two uniformly at random chosen disjoint sets $A$ and $B$ in $G$ is $\approx |A||B|$ for constant $|A|$ and $|B|$ and for large $n$. This is a consequence of the fact that in $G$ two vertices are adjacent to a third common vertex, or share an edge, or both.

### 3.7.3 The Multinomial distribution

We present below some facts related to the Multinomial Distribution, as they can also be found in [17]. Suppose we have an event that may have $k$ possible outcomes and each outcome $i$ has the same probability $p_i = 1/k$ to occur. Let us perform an experiment by repeating this event $n$ times. The probability that we have a given pattern of outcomes follows a multinomial distribution. Let $o_i$ be a random variable that represents the number of times outcome $i$ occurred in our experiment.

Then, the expected value and variance of the random variable $o_i$ are $E[o_i] = np_i = n(1/k)$ and $var[o_i] = np_i(1 - p_i) = n(1/k)(1 - (1/k))$ respectively, for all $i$. The covariance between $o_i$ and $o_j$ is equal to $cov[o_i, o_j] = -np_ip_j = -n(1/k^2)$.

### 3.7.4 Graph construction

We proceed to presenting our construction for the problem under consideration. We will show that the graph to be constructed below is such that if we pick uniformly at random two distinct sets of $k$ vertices each, the size of the set of vertices in that graph common to the two sets is a random variable with mean $k$ and variance $k - 1$.

This explicit construction of the graph, if we allow a comparison with random graphs, matches (is slightly better than) the random graph construction suggested by Valiant which achieves the same mean, but variance $k$, for all possible choices of two sets $A$, $B$, though the means and variances have different meanings in both cases. In the deterministically constructed graph below, it might be the case that two given sets of vertices have no vertices in common, while in a random graph, due to the randomization of its edges, this is highly unlikely.

Let us first fix an integer $k > 1$. Parameter $k$ is considered constant for the remainder of this discussion. Let us assume that the number of vertices $n$ of the graph to be constructed is such that $n = k(q^2 + q + 1)$, where $q$ is an odd power of an odd prime, that is $q = p^r$, $p$ an odd prime and $r$ an odd integer. Since $n$ is of that form, we get that $n/k = q^2 + q + 1$. We then build $k$ copies of the Erdos-Renyi graph constructed in the previous section. Each of these $k$ copies has $n/k$ vertices and therefore degree $O(\sqrt{n/k})$. Let us call the $i$-th copy of such a graph, $G_i$, $i = 1, \ldots, k$, and for a vertex $v$ in the Erdos-Renyi graph $G$ of the previous section let $v_i$ denote the copy of $v$ in $G_i$. Let $G_i = (V_i, E_i)$, where $V_i$, $E_i$ are the set of vertices and edges respectively of $G_i$.

The graph to be constructed may consist only of these $k$ unconnected components. If this looks unnatural, we may connect these components so that the resulting graph is connected. In this case, we can connect these $k$ subgraphs as follows. We connect these copies in the following array-like fashion. For each $v \in G$ and $v$ an absolute point of the projective plane used to construct $G$, vertex $v_i$ of $G_i$ is connected to vertex $v_{i+1}$ of $G_{i+1}$, $1 \leq i < k$. Let the edges added during this step form set $E'$. These edges add to the degree of any vertex in any of the $k$ subgraphs at most two, an asymptotically negligible constant. This way the graph becomes $G^k = (V^k, E^k)$ where the set of vertices $V^k = V_1 \cup \ldots \cup V_k$ and the set of edges $E^k = E_1 \cup \ldots \cup E_k \cup E'$. We will now show that $G^k$ has the properties claimed in the beginning of this section (with or without the edges $E'$).

### 3.7.5 Statistical Properties of graph $G^k$

We use the symbol $\approx$ below to denote that the ratio of the quantities on the left and right hand side of that symbol tends to one as $n \to \infty$, for constant $k$.

Let two disjoint sets of vertices of graph $G^k_P$ be of cardinality $k$ each and let us call them $A$ and $B$ respectively. The vertices of these two sets will be selected among the $n$ vertices of the $G^k$ uniformly at random and without replacement. Each such selected vertex belongs for large $n$ and constant $k$ with probability approximately $1/k$ to set $V_i$ of $G_i$. This is because all sets $V_i$ are of the same size $n/k$ and the probability that a selected vertex belongs to some set $V_j$, for some $j$, during this selection process is at least $(n/k - 2k)/n$ and at most $1/k$ Let $a_i$ and $b_i$, $i = 1, \ldots, k$, be random variables that represent the number of vertices of $A$ and $B$ respectively so selected that belong to set $V_i$ of graph $G_i$.

This selection procedure can be rephrased as follows. Let us perform the following procedure twice, once for set $A$ and once for set $B$. We repeat $k$ times an experiment that has $k$, asymptotically equally probable, possible outcomes each outcome corresponding to an index $i$ of a graph $G_i$. If the outcome of an experiment is $i$ we choose uniformly at random and without replacement one vertex from set $V_i$ of $G_i$.

Given that sets $A$ and $B$ have size $k$, a constant, for $n$ large enough the experiments can be considered independent so that the outcome of an experiment is as likely to occur as any other outcome, in spite of the fact that the outcome of say, the $j$th experiment depends on the outcomes of all previous $j-1$ experiments. This is due to the selection without replacement of sets $A$ and $B$.

Since a constant number of $2k$ vertices are drawn to form sets $A$ and $B$ for large $n$ we may assume that the probability that the outcome of an experiment is say, $i$ is $\approx 1/k$.

**Theorem 13** *In graph $G^k$, where $k$ is a constant, if disjoint sets $A$ and $B$ of size $k$, are chosen uniformly at random and without replacement from the set of vertices of $G^k$, then the size of set $C = N^*(A, B)$ in $G^k$ is given by a random variable with expectation and variance $\approx k$ and $\approx k - 1$ respectively.*

**Proof:** Let sets $A$ and $B$ be so chosen. From the discussion prior to the statement of the Theorem we get that:

$$\mathcal{E}[a_i] = \mathcal{E}[b_i] \approx k \cdot \frac{1}{k} = 1 \ ,$$

$$var[a_i] = var[b_i] \approx k \cdot \frac{1}{k} \cdot (1 - \frac{1}{k}) = (1 - \frac{1}{k}) \ ,$$

$$cov[a_i, a_j] = cov[b_i, b_j] \approx -k \cdot \frac{1}{k} \cdot \frac{1}{k} = -\frac{1}{k}.$$

We would be interested in the behavior of the random variable $X$ that gives the number of vertices of set $C$, that is the number of vertices common to $A$ and $B$. This random variable, by taking into consideration the conclusions drawn earlier in this section related to the construction of the Erdos-Renyi graph, is approximately equal to

$$X \approx a_1 b_1 + \ldots + a_k b_k = \sum_{i=1}^{k} a_i b_i.$$

We compute the expectation $\mathcal{E}[X]$ and variance $var[X]$ of $X$.

$$\mathcal{E}[X] \approx \mathcal{E}[\sum_{i=1}^{k} a_i \cdot b_i] = k\mathcal{E}[a_i \cdot b_i] = k\mathcal{E}[a_i]\mathcal{E}[b_i] \approx k \cdot 1 \cdot 1 = k \ ,$$

since the random variables $a_i,\ b_i$ are considered to be mutually independent.
We know that the variance of the sum of $k$ random variables $c_i$ is given by:

$$var[\sum_{i=1}^{k} c_i] = \sum_{i=1}^{k} var[c_i] + \sum_{i \neq j} cov[c_i, c_j].$$

We let $c_i$ denote the product $c_i = a_i b_i$. Then the mean and the variance of $c_i$ for mutually independent $a_i$ and $b_i$ are given by:

$$\mathcal{E}[c_i] = \mathcal{E}[a_i b_i] = \mathcal{E}[a_i]\mathcal{E}[b_i] + cov[a_i, b_i] = \mathcal{E}[a_i]\mathcal{E}[b_i] \approx 1,$$

and

$$\begin{aligned} var[c_i] &= var[a_i b_i] \\ &= E^2[a_i]var[b_i] + E^2[b_i]var[a_i] + var[a_i]var[b_i] \\ &\approx 1^2 \cdot var[a_i] + 1^2 \cdot var[b_i] + var[a_i]var[b_i] \\ &\approx 2(1 - \frac{1}{k}) + (1 - \frac{1}{k})^2. \end{aligned}$$

The covariance of $cov[c_i, c_j] = cov[a_i \cdot b_i, a_j \cdot b_j]$ can be computed from the expression:

$$cov[c_i, c_j] = \mathcal{E}[c_i \cdot c_j] - \mathcal{E}[c_i]\mathcal{E}[c_j]$$

and the fact that,

$$\mathcal{E}[c_i \cdot c_j] = \mathcal{E}[a_i \cdot b_i \cdot a_j \cdot b_j] = \mathcal{E}[(a_i \cdot a_j) \cdot (b_i \cdot b_j)] = \mathcal{E}[a_i \cdot a_j] \cdot \mathcal{E}[b_i \cdot b_j].$$

Similarly, we get

$$\mathcal{E}[a_i \cdot a_j] = \mathcal{E}[a_i] \cdot \mathcal{E}[a_j] + cov[a_i, a_j] \approx 1 - \frac{1}{k}.$$

This is also approximately equal to $\mathcal{E}[b_i \cdot b_j]$ as well. Therefore, we get that:

$$\mathcal{E}[c_i \cdot c_j] \approx (1 - \frac{1}{k})^2$$

and then,

$$cov[c_i, c_j] \approx (1 - \frac{1}{k})^2 - 1.$$

From the last equation and the expression for $var[c_i]$ we get that:

$$\begin{aligned}
var[X] &\approx var[\sum_{i=1}^{k} c_i] = var[\sum_{i=1}^{k} a_i \cdot b_i] \\
&= k \cdot var[c_i] + k \cdot (k-1) \cdot cov[c_i, c_j] \\
&\approx k \cdot (2(1 - \frac{1}{k}) + (1 - \frac{1}{k})^2) + k(k-1)((1 - \frac{1}{k})^2 - 1) \\
&= k - 1,
\end{aligned}$$

and thus our claims are proved. ■

### 3.7.6 Remarks on the construction of $G^k$

The construction of graph $G^k$ gives high variance for the size of set $C$ common to $A$ and $B$. We can reduce this variance if we somehow restrict the way sets $A$ and $B$ are formed. Suppose we pick the first vertex of set $A$ (the discussion for $B$ follows similarly) uniformly at random among the vertices of subgraph $G_1$ of $G^k$. Let this vertex be $v_1$. We then choose the other $k-1$ vertices of set $A$ to be the vertices in the graphs $G_2, \ldots, G_k$ that are isomorphic to $v_1$, that is, following a previous convention, we select vertices $v_2, \ldots, v_k$ from these graphs respectively. Following this selection each of the two sets $A$ and $B$ has exactly one vertex in each graph $G_i$, $i = 1, \ldots, k$ thus getting a set $C$ in $G^k$ common to $A$ and $B$ of cardinality $k$ with high probability. The probability that set $C$ is of size zero, the only other alternative, is $O(1/n)$ by Claim 2. The advantage of this selection method is that it allows us to exploit the properties of the graph when we use this graph to form a set at some iteration $r$, just as we are going to do for random graphs in the sections to follow.

In another variation of this set selection scheme instead of picking the vertices in the graphs $G_2, \ldots, G_k$ to be isomorphic to the vertex picked in $G_1$, we pick the vertices in any $G_i$ independently of the other choices.

## 3.8 Iterated Mean and Variance for Random Graphs

We now turn our attention to studying the connectivity properties of the class of random graphs $G_{n,1/(k\cdot n)^{1/2}}$. For the discussion to follow we assume that $k$ is a constant. Later, we shall also examine cases where $k$ grows.

We first examine the mean and the variance of the size of a set formed at some constant iteration $r$, as defined in section 3.4, in $G_{n,1/(k\cdot n)^{1/2}}$ random graphs. Suppose we have two distinct sets of vertices one of size $i$ and another of size $j$. The probability that a vertex $c$ of the graph is connected to some vertex $a$ ($a \neq c$) of the first set and some vertex $b$ ($b \neq c$) of the second set is equal to

$$r_{ij} = (1-(1-p)^i)(1-(1-p)^j) \approx\ ijp^2 = \frac{ij}{kn}, \tag{3.1}$$

where $p = 1/\sqrt{nk}$ denotes the edge probability of the random graph. The approximation in Equation (3.1) holds provided $pi = o(1)$ (similarly for $pj$).

We shall select sets of common vertices according to a process outlined in [1] and described below. For constant $r$, a set formed at iteration $r$ will not intersect with sets previously formed, except with small probability as $n \to \infty$.

Suppose a set is to be formed at some constant iteration $r$. Let $V$ be the set of vertices of the class $G_{n,1/(k\cdot n)^{1/2}}$ of random graphs, $F$ an arbitrary subset of it of size say $n^{1/5}$, $P = V - F - \{v \mid v \in V$ belongs to one of the $2^r$ sets of size $k$ formed at iteration 0 $\}$, and $n_0 = |P| = n - n^{1/5} - 2^r k$. The term $2^r k$ in the expression for $|P|$ counts the vertices of $2^r$ sets of $k$ vertices each at iteration zero. In the steps of Definition 6 in section 3.4 we are going to select the vertices that will form a set $C$ at some iteration $m \le r$ among the vertices of set $P$ rather than of $V$. This set $C$ will be common to two sets $A$ and $B$ formed at iteration $m-1$. If sets $A$, $B$ have sizes $i$ and $j$ respectively then, the size of $C$ is equal to $l$ with probability, given by a binomial distribution $B(l; n_0, r_{i,j}) = \binom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l}$. If the value $l$ is chosen for the size of set $C$, then we choose $l$ vertices uniformly at random among the $\binom{n_0}{l}$ sets of vertices of size $l$ of $P$. We select and remove the vertices so chosen from $P$. We restore the size of set $P$ to $n_0$ by moving $l$ arbitrary vertices from $F$ to $P$. If the size of $F$ is less than $l$ then we declare failure of this selection process and subsequently of the formation of a set at iteration $r$. One can prove that with high probability in the construction of a tree of iteration $r$ the total number of vertices that will be selected from $P$ will be less than the cardinality of set $F$. Thus this selection procedure will succeed with high probability. This is proved in Claim 4.

We call this method of selecting common vertices **Select**.

One needs to show that any result that will be obtained in this chapter by restricting the choice of vertices in Select to set $P$ rather than $V$ could have been obtained, with probability

$1 - o(1)$, without the use of Select. This is proved as follows. The probability that any vertex, including vertices in $F$, involved in the construction of a tree of iteration $r$ (there are at most $n^{1/5} + 2^r k$ such vertices) is connected to any vertex of $F$ is at most is at most $(2^r k + n^{1/5})(1 - (1-p)^{n^{1/5}}) \leq 2n^{1/5}(1 - e^{-n^{1/5}/((1/p)-1)}) \leq 2n^{2/5}/(\sqrt{kn} - 1) = o(1)$. We used here the inequalities $(1 - 1/n)^n \leq 1/e \leq (1 - 1/n)^{n-1}$, $e^{-x} \geq 1 - x \quad (x \geq 0)$. Hence, with probability $1 - o(1)$, no two vertices among the ones included in a set formed during the construction of a tree of iteration $r$ will have a vertex in common in $F$.

The claim below summarizes the properties of Select that will be of interest in this discussion.

**Claim 4** *When Select is executed in the steps of Definition 6 in section 3.4 to form a set of vertices at some constant iteration $r$ in the class of $G_{n,1/(k \cdot n)^{1/2}}$ random graphs, where $k$ is constant, the number of vertices chosen in Select up to iteration $r$ is with high probability at most $n^{1/5}$ altogether.*

This claim is to be deduced at the end of this section using Propositions 2 and 3. We note that the requirement that $r$ be a constant can be replaced by one that restricts $r$ to be independent of $n$ only.

### 3.8.1 Calculating the mean

Let the size of a set of vertices formed at iteration $r$ be represented by random variable $Y_r$. We are interested in the mean and variance of $Y_r$ when the mean and variance of $Y_{r-1}$ are known.

With reference to the example presented in the introduction, let $A, B, D$, and $E$ be mutually disjoint sets of vertices of $G_{n,1/(k \cdot n)^{1/2}}$, each of size $k$. We shall study the distribution of the size of sets $C$ and $F$ formed at iteration one and that of set $G$ formed at iteration two. Sets $C$ and $F$ are common to $A$, $B$ and $D$, $E$ respectively, and set $G$ to $C$ and $F$.

Let set $A$ formed at some iteration $r - 1 \geq 0$ be of size $i$ with probability $p_i = Prob(Y_{r-1} = i)$ and set $B$ formed at the same iteration be of size $j$ with probability $p_j$. We shall use for the latter probability $p_j$ the symbol $q_j$ to make explicit the respective contributions in the proofs to follow of $A$ and $B$ clearly. Clearly $\sum_i p_i = 1$ and $\sum_j q_j = 1$ . The probability that a set of vertices $C$ formed at iteration $r$ and common to $A$ and $B$ has size $l$ is given by

$$\sum_{i,j} p_i q_j \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l} = \sum_{i,j} p_i p_j \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l}.$$

The vertices of set $C$ are chosen by process Select. Results related to the mean and variance of $Y_r$, the random variable representing the size of set $C$, are shown below. We split the related proofs for the mean and variance into two Propositions for convenience only. Each result uses the other one in the sense that the $i$th inductive step of the proof of either result uses the results deduced for the $(i-1)$th step. We first show that at iteration one random variable $Y_1$ has mean and variance both approximately equal to $k$, asymptotically for large $n$. This result will be the basis of the inductive proofs (induction on the iteration) of Propositions 2 and 3.

Let the means and variances of the two random variables representing the sizes of $A$ and $B$ be $\mu_1, v_1$ and $\mu_2, v_2$ respectively. Since the two sets are formed at the same iteration the two random variables are identically distributed but independent so that $\mu_1 = \mu_2$ and $v_1 = v_2$.

We then get the following proof of Proposition 2 stated in section 3.2.

**Proof of Proposition 2:** In the base case $r = 1$, two sets $A$ and $B$ of size $k$ each have a vertex in common with probability $r_{kk}$ given by Equation (3.1). Because of Select, the number of vertices in $P$ common to these two sets follows a binomial distribution $B(k; n_0, r_{kk})$. Since $r_{kk} \to 0$ and $n_0 r_{kk} \to k$ as $n \to \infty$,the Binomial distribution can be approximated by a Poisson distribution with mean $k$. Then the random variable $Y_1$ that represents the size of set $C$ formed at iteration one and common to $A$ and $B$ is such that $\mathcal{E}[Y_1] \approx k = k^2/k$ and $var(Y_1) \approx k$, as required.

Then for iterations $t$, $1 < t \leq r - 1$, we inductively assume Proposition 3, $\mathcal{E}[Y_t] \approx \mathcal{E}[Y_{t-1}]\mathcal{E}[Y_{t-1}]/k$, and $\mathcal{E}[Y_t] \approx k$ since $\mathcal{E}[Y_{t-1}] = \mu_1 = \mu_2 \approx k$.

We prove the claim for two sets $A$, $B$ formed at iteration $r-1$ so that set $C$ common to $A$ and $B$ is formed at iteration $r$. Let the expectations and variances of $A$ and $B$ be $\mu_1, v_1$ and $\mu_2, v_2$ respectively. By the inductive hypothesis $\mu_1 = \mu_2 = \mathcal{E}[Y_{r-1}]$ are approximately $k$

and by Proposition 3, $v_1 = v_2 = var(Y_{r-1})$ is also bounded above by a term dependent on $k$ and the iteration and independent of $n$.

Let set $A$ be of size $i$ with probability $p_i$ and $B$ be of size $j$ with probability $q_j$. We then get the following.

$$\mathcal{E}[Y_r] = \sum_{l=0}^{n_0} l \sum_{i,j} p_i q_j \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l}.$$

We now take into consideration the properties of the binomial distribution and get the following three Cases, depending on the range of the indices $i, j$ with respect to $\sqrt{nk}$:

1. $i$ and $j$ are less than $\sqrt{nk}$,
2. $i$ and $j$ are both at least $\sqrt{nk}$,
3. $i$ or $j$ are such that one of them follows Case 2 and the other follows Case 1.

Let $\mathcal{E}_m[Y_r]$ $(m = 1, 2, 3)$ be the contribution to the expectation $\mathcal{E}[Y_r]$ of each of the first two Cases, and for Case 3, of each of its two subcases so that

$$\mathcal{E}[Y_r] = \mathcal{E}_1[Y_r] + \mathcal{E}_2[Y_r] + 2\mathcal{E}_3[Y_r].$$

The following inequalities are known to be true.

$$(1 - 1/n)^n \leq 1/e \leq (1 - 1/n)^{n-1} \ ,$$

$$e^{-x} \geq 1 - x \quad (x \geq 0) \ ,$$

$$e^{-x} \leq 1 - x + x^2/2 \quad (0 \leq x < 1).$$

**Case 1:** Since

$$\begin{aligned} r_{ij} &= (1 - (1-p)^i)(1 - (1-p)^j) \\ &\leq \frac{(ip)}{(1-p)} \frac{(jp)}{(1-p)} \\ &= \frac{ij}{kn} \frac{1}{(1-p)^2} \end{aligned}$$

$$\approx \frac{ij}{kn},$$

and

$$\begin{aligned}
r_{ij} &= (1-(1-p)^i)(1-(1-p)^j) \\
&\geq (1-e^{-pi})(1-e^{-pj}) \\
&\geq (ip - i^2p^2/2)(jp - j^2p^2/2) \\
&= \left(\frac{ij}{kn} - \frac{i^2 j}{2(kn)^{3/2}} - \frac{j^2 i}{2(kn)^{3/2}} + \frac{i^2 j^2}{4(kn)^2}\right),
\end{aligned}$$

we get

$$\begin{aligned}
\mathcal{E}_1[Y_r] &= \sum_{i<\sqrt{nk}, j<\sqrt{nk}} p_i q_j \sum_l l \tbinom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} \\
&= \sum_{i,j} p_i q_j n_0 r_{ij} \\
&\leq \sum_{i,j} p_i q_j n_0 \frac{ij}{kn} \frac{1}{(1-p)^2} \\
&\approx \sum_{i,j} p_i q_j \frac{ij}{k} \\
&= \frac{1}{k} \sum_i i p_i \sum_j j q_j \\
&\leq \frac{\mu_1 \mu_2}{k}.
\end{aligned}$$

We examine the lower bound for $r_{ij}$ and get that

$$\begin{aligned}
\mathcal{E}_1[Y_r] &= \sum_{i<\sqrt{nk}, j<\sqrt{nk}} p_i q_j \sum_l l \tbinom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} \\
&= \sum_{i,j<\sqrt{kn}} p_i q_j n_0 r_{ij} \\
&\geq \sum_{i,j<\sqrt{kn}} p_i q_j n_0 \left(\frac{ij}{kn} - \frac{i^2 j}{2(kn)^{3/2}} - \frac{j^2 i}{2(kn)^{3/2}} + \frac{i^2 j^2}{4(kn)^2}\right)
\end{aligned}$$

$$
\begin{aligned}
&\approx \sum_{i,j<\sqrt{kn}} p_i q_j \left(\frac{ij}{k} - O(1/\sqrt{n})\right) \\
&\geq \sum_{i,j} p_i q_j \left(\frac{ij}{k} - O(1/n^{1/4})\right) \\
&\approx \frac{1}{k}\sum_i ip_i \sum_j jq_j \\
&= \frac{\mu_1\mu_2}{k}.
\end{aligned}
$$

We explain below how the various inequalities were derived. We used the inductive assumption for the mean and the variance to get

$$
\begin{aligned}
\sum_{i<\sqrt{nk},\ j<\sqrt{kn}} ijp_iq_j &= \sum_{i,j} ijp_iq_j - \sum_{i\geq\sqrt{kn},j\geq\sqrt{kn}} ijp_iq_j \\
&- \sum_{i\geq\sqrt{kn},j<\sqrt{kn}} ijp_iq_j - \sum_{i<\sqrt{kn},j\geq\sqrt{kn}} ijp_iq_j \\
&\geq \sum_{i,j} ijp_iq_j - O(1/n^{1/4}).
\end{aligned}
$$

In order to show this last derivation we need to show the case for

$$
\begin{aligned}
\sum_{i\geq\sqrt{kn},j<\sqrt{kn}} ijp_iq_j &= \sum_{i\geq\sqrt{kn}} ip_i \sum_{j<\sqrt{kn}} jq_j \\
&\leq \sum_{i\geq\sqrt{kn}} ip_i \sum_j jq_j \\
&\leq \sum_{i\geq\sqrt{kn}} ip_i\mu_2 \\
&= \sum_{n\geq i>n^{3/4}} ip_i\mu_2 + \sum_{n^{3/4}\geq i\geq\sqrt{kn}} ip_i\mu_2 \\
&\leq n\sum_{i\geq n^{3/4}} p_i\mu_2 + n^{3/4}\sum_{i\geq\sqrt{kn}} p_i\mu_2 \\
&\leq n\frac{v_1+\mu_1^2}{n^{1.5}}\mu_2 + n^{3/4}\frac{v_1+\mu_1^2}{kn}\mu_2 \\
&\leq O(1/n^{1/2}) + O(1/n^{1/4}),
\end{aligned}
$$

where we claimed Theorem 1 (with $g(X) = X^2$) and $Prob(X \geq t) = Prob(X^2 \geq t^2)$ for a non-negative random variable $X$, to get the fourth inequality.

The other two cases can be derived similarly. The following is also true.

$$\sum_{i,j<\sqrt{kn}} p_i q_j n_0 \left(-\frac{i^2 j}{2(kn)^{3/2}} - \frac{j^2 i}{2(kn)^{3/2}} + \frac{i^2 j^2}{4(kn)^2}\right) \geq -O(1/\sqrt{n}). \tag{3.2}$$

In order to prove this claim we need to use the inductive hypothesis and the definition $var(X) = \mathcal{E}[X^2] - (\mathcal{E}[X])^2$ so that $\sum_i i^2 p_i$ is bounded above by a term dependent on $\mu_1$ and $v_1$. A similar claim can be made for the sum $\sum_j j^2 q_j$. Then the first term of the sum in Equation (3.2) gives

$$\begin{aligned}
\sum_{i,j<\sqrt{kn}} p_i q_j n_0 \left(-\frac{i^2 j}{2(kn)^{1/2}}\right) &= \left(-\sum_{i<\sqrt{nk}} p_i i^2\right)\left(\frac{1}{2(kn)^{3/2}}\sum_{j<\sqrt{kn}} j q_j\right) \\
&\geq \left(-\sum_{i<\sqrt{nk}} p_i i^2\right)\left(\frac{1}{2(k^3 n)^{1/2}}\right)\mu_2 \\
&\geq -O(1/\sqrt{n}) \\
&\approx 0.
\end{aligned}$$

The second term of the sum in Equation (3.2) is treated similarly and the last one is at least zero.

**Case 2:** We now consider the expression

$$\begin{aligned}
\mathcal{E}_2[Y_r] &= \sum_l l \sum_{i\geq\sqrt{nk}, j\geq\sqrt{nk}} p_i q_j \binom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} \\
&= \sum_{i,j} p_i q_j n_0 r_{ij} \\
&\leq \sum_{i,j\geq\sqrt{nk}} p_i q_j r_{ij} n_0 \\
&= n_0 \sum_{i\geq\sqrt{nk}} p_i \sum_{j\geq\sqrt{nk}} q_j \\
&\leq \frac{{\mu_1}^2 + v_1}{nk}\frac{{\mu_2}^2 + v_2}{nk} n_0
\end{aligned}$$

$$\begin{aligned} &\leq c_1 \frac{1}{n} \text{ , for some constant } c_1, \\ &\leq c_1 \frac{1}{n} \to 0 \text{ as } n \to \infty, \end{aligned}$$

where we claimed Theorem 1 (with $g(X) = X^2$) and $Prob(X \geq t) = Prob(X^2 \geq t^2)$ for a non-negative random variable $X$, to get an upper bound for $\sum_{i \geq \sqrt{nk}} p_i = Pr(i \geq \sqrt{nk})$. By the inductive hypothesis, the last two inequalities above were derived.

We now show that the contribution to the expectation when the indices are constrained as in Case 3 is also negligible.

**Case 3:** For Case 3 suppose $i < \sqrt{kn}$ and $j \geq \sqrt{nk}$. The other case is the one with $j < \sqrt{kn}$ and $i \geq \sqrt{nk}$, which can be treated similarly. Then we have

$$pi - p^2 i^2/2 \leq r_{ij} = (1-(1-p)^i)(1-(1-p)^j) \leq pi \frac{1}{1-p} = \frac{i}{\sqrt{nk}} \frac{1}{1-p},$$

and therefore, following the steps of the previous cases we get

$$\begin{aligned} \mathcal{E}_3[Y_r] &= \sum_l l \sum_{i,j} p_i q_j \tbinom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} \\ &\leq \sum_{i<\sqrt{nk}, j \geq \sqrt{nk}} p_i q_j n_0 \frac{i}{\sqrt{nk}} \frac{1}{1-p} \\ &= \sum_{i<\sqrt{nk}} i p_i \sum_{j \geq \sqrt{nk}} q_j \frac{1}{\sqrt{nk}} n_0 \frac{1}{1-p} \\ &\leq \mu_1 \frac{v_2 + {\mu_2}^2}{kn} n_0 \frac{1}{\sqrt{nk}} \frac{1}{1-p} \to 0 \text{ as } n \to \infty. \end{aligned}$$

We then combine the various upper and lower bounds for $\mathcal{E}_i[Y_r]$, $i = 1, 2, 3$, to get that the iterated mean is asymptotically for large $n$

$$\mathcal{E}[Y_r] \approx \frac{\mu_1 \mu_2}{k}.$$

We also have by the inductive hypothesis that $\mu_1 = \mu_2 \approx k$ so that $\mathcal{E}[Y_r] \approx k$, asymptotically for large $n$, for every constant iteration $r$. ■

### 3.8.2 Calculating the variance

We follow the terminology and techniques of the previous section to find the variance of $Y_r$, $r \geq 1$. The proof of Proposition 3 follows.

**Proof of Proposition 3:** It is true by definition ([17]) that

$$var(Y_r) = \mathcal{E}[Y_r^2] - \mathcal{E}^2[Y_r].$$

We thus estimate $\mathcal{E}[Y_r^2]$ using the second moment of the binomial distribution to estimate $var(Y_r)$.

The three cases distinguished in the Proof of Proposition 2 depending on the ranges of $i$, $j$ are also considered here. Case 1, following the steps of the proof of Proposition 2 and using the same approximations for $r_{ij}$ gives

$$\begin{aligned}
\mathcal{E}_1[Y_r^2] &= \sum_l l^2 \sum_{i,j<\sqrt{kn}} \tbinom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} p_i q_j \\
&= \sum_{i,j<\sqrt{kn}} p_i q_j \sum_l l^2 \tbinom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} \\
&\approx \sum_{i,j<\sqrt{kn}} p_i q_j (n_0^2 r_{ij}^2 + n_0 r_{ij}(1-r_{ij})) \\
&\leq \sum_{i,j} p_i q_j \left( \frac{n_0^2 i^2 j^2}{(k^2 n^2)(1-p)^4} + \frac{n_0 ij}{(kn)(1-p)^2} \right) \\
&\approx \frac{\sum i^2 p_i \sum j^2 q_j}{k^2} + \frac{\sum i p_i \sum j q_j}{k} \\
&\approx \frac{(v_1 + {\mu_1}^2)(v_2 + {\mu_2}^2)}{k^2} + \frac{\mu_1 \mu_2}{k}.
\end{aligned}$$

Cantelli's ([17]) inequality yields for a non-negative random variable $X$ with mean $\mu$ and variance $v$ the following bound:

$$Pr(X > a + \mu) \leq \frac{v}{v + a^2} \qquad (a > 0)\ .$$

We apply Cantelli's inequality in Case 2 to get the second inequality below.

$$\begin{aligned}
\mathcal{E}_2[Y_r^2] &= \sum_l l^2 \sum_{i,j\geq\sqrt{kn}} \tbinom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} p_i q_j \\
&\approx \sum_{i,j\geq\sqrt{kn}} p_i q_j (n_0^2 r_{ij}^2 + n_0 r_{ij}(1-r_{ij})) \\
&\leq \sum_{i\geq\sqrt{nk}, j\geq\sqrt{kn}} p_i q_j (n_0^2 + n_0) \\
&= \sum_{i\geq\sqrt{nk}} p_i \sum_{j\geq\sqrt{nk}} q_j n_0^2 + \sum_{i\geq\sqrt{nk}} p_i \sum_{j\geq\sqrt{nk}} q_j n_0 \\
&\leq \frac{v_1}{v_1+kn}\frac{v_2}{v_2+kn} n_0^2 + \frac{v_1}{v_1+kn}\frac{v_2}{v_2+kn} n_0 \\
&\approx \frac{v_1 v_2}{k^2}.
\end{aligned}$$

For $i < \sqrt{nk}$ and $j \geq \sqrt{kn}$ of Case 3 we have that $r_{ij} \leq i/\sqrt{kn}$ and subsequently

$$\begin{aligned}
\mathcal{E}_3[Y_r^2] &= \sum_l l^2 \sum_{i<\sqrt{kn}, j\geq\sqrt{kn}} \tbinom{n_0}{l} {r_{ij}}^l (1-r_{ij})^{n_0-l} p_i q_j \\
&\approx \sum_{i<\sqrt{kn}, j\geq\sqrt{kn}} p_i q_j (n_0^2 r_{ij}^2 + n_0 r_{ij}(1-r_{ij})) \\
&\leq \sum_{i<\sqrt{nk}} p_i \left(\frac{i}{\sqrt{nk}}\right)^2 \sum_{j\geq\sqrt{nk}} q_j n_0^2 + \sum_{i<\sqrt{nk}} p_i \frac{i}{\sqrt{nk}} \sum_{j\geq\sqrt{nk}} q_j n_0 \\
&\leq \frac{v_1+\mu_1^2}{kn} n_0^2 \frac{v_2}{v_2+kn} + \frac{v_1+\mu_1^2}{kn} n_0 \frac{v_2}{v_2+kn} \\
&\approx \frac{(v_1+\mu_1^2)v_2}{k^2}.
\end{aligned}$$

The symmetric case $j < \sqrt{nk}$ and $i \geq \sqrt{kn}$ of Case 3 yields a similar bound.

Adding all contributions in the limit $n \to \infty$ we get

$$\begin{aligned}
var(Y_r) &= \mathcal{E}[Y_r^2] - \mathcal{E}^2[Y_r] \\
&\preceq \frac{\mu_1\mu_2}{k} + \frac{4v_1v_2}{k^2} + \frac{2{\mu_1}^2 v_2 + 2{\mu_2}^2 v_1}{k^2},
\end{aligned}$$

as desired. By the inductive hypothesis of Propositions 2 and 3, $\mu_1, \mu_2$ and $v_1, v_2$ depend on

$k$ and the iteration only and not on $n$ therefore $var(Y_r)$ is bounded above by a term that also depends on $k$ and the iteration only and not on $n$.

We note here that we can get a better bound for $var(Y_r)$ after our discussion in section 3.11 where it will be inferred that $\sum_{j>f(n)} p_j < 1/n^4$ for $n \to \infty$, where $f(n)$ is any function of $n$ and $p_j$ gives the probability that a set $A$ formed at iteration $r-1$ has size $j$. Then the formula for *var*($Y_r$), with Cases 2 and 3 contributing $o(1)$, becomes

$$var(Y_r) \approx \frac{\mu_1\mu_2}{k} + \frac{v_1v_2}{k^2} + \frac{{\mu_1}^2 v_2 + {\mu_2}^2 v_1}{k^2}.$$

■

The Proof of Claim 4 which utilizes Propositions 2 and 3 follows.

**Proof of Claim 4:** In Propositions 2 and 3 we showed that the mean and variance of $Y_r$, for constant $r$, depends on $k$ and the iteration only and is thus constant for a constant $r$.

The probability of having a set of common vertices formed at some iteration $i \leq r$ of size greater than $n^{1/5}/2^{r+1}$ by Theorem 1 (with $g(X) = X^2$), the fact that $Prob(X \geq t) = Prob(X^2 \geq t^2)$ for a non-negative random variable $X$, and Propositions 2 and 3 is at most $1/\Theta(n^{2/5})$. There are at most $2^r$ sets formed at iterations $1 \leq i \leq r$. Therefore the probability that the vertices of all these sets is more than $(n^{1/5}/2^{r+1})\ 2^r \leq n^{1/5}$ overall, is also by subadditivity $1/\Theta(n^{2/5})$. This proves Claim 4. ■

## 3.9 Upper bounds on probabilities when the mean and the variance of a random variable are given

We use Markov's (Theorem 1) and Cantelli's inequalities to give upper bounds on the probability of having a set of "bad" size formed at some iteration. The computation of these bounds requires the knowledge of the first two moments of random variable $Y_r$. We can also obtain sharper upper bounds by examining the distribution of the size of a set of vertices formed at some iteration through the convergence of *Bernstein* polynomials ([44], [9], [17]). This case is treated elsewhere. Although problems related to this one have appeared in populational genetics and solved using Markov chains or Branching processes those techniques seem to be of little use here.

We give below a rather easily obtainable upper bound [10] on the probability of getting a set of size zero formed at iteration $r$, given that the mean and the variance of the random variable $Y_r$ representing the size of such a set are known. Let $Y_r$ be a random variable with $\mathcal{E}[Y_r] = \mu$ and variance $\mathcal{E}[(Y_r - \mu)^2] = v$. The following is then true.

$$\begin{aligned} Pr(Y_r = 0) &\leq 1 - \frac{(\mathcal{E}^2[Y_r])}{\mathcal{E}[Y_r^2]} \\ &= 1 - \frac{\mu^2}{\mu^2 + v} \\ &= \frac{v}{\mu^2 + v}. \end{aligned}$$

Suppose we want to find upper bounds on the probability $Pr(Y_r > ak)$, where $a > 1$ is some given positive real number. A straightforward bound is given by an application of Theorem 1 (with $g(X) = X^2$) and the fact that $Prob(Y_r > ak) = Prob(Y_r^2 \geq (ak)^2)$ for a non-negative random variable $Y_r$:

$$Pr(Y_r > ak) \leq \frac{\mathcal{E}[Y_r^2]}{a^2k^2} = \frac{v + \mathcal{E}^2[Y_r]}{a^2k^2} = \frac{v + \mu^2}{a^2k^2}.$$

We can get a better bound by applying Cantelli's inequality (see [9]), which gives that

$$Pr(Y_r > ak + \mu) \leq \frac{v}{v + a^2k^2},$$

where $\mu$, $v$ is the mean and variance of $Y_r$.

Substituting the expressions for $\mu, v$ derived in Propositions 2 and 3 yields some rough upper bounds on the probability of having $Y_r$ equal to zero or at least $ak$, for some constant $a > 1$.

## 3.10 Tails of the Distribution of the sizes of sets formed at various iterations

In this section we find better bounds than the ones previously suggested for the left (i.e. $Prob(Y_r = 0)$) and the right tail (i.e. $Prob(Y_r > ak)$, for $a > 1$) of random variable $Y_r$, $r = 1, 2$.

In the discussion to follow we assume that $k \geq 20$ so that Stirling's approximation formula for the factorial will be applicable to say, $k!$.

### 3.10.1 Iteration one

A set $C$ formed at iteration one is common to two sets $A$ and $B$ each of size $k$ formed at iteration zero. Hence, we get from Equation (3.1) that $r_{kk} \approx k/n$. The probability that set $C$ is of size $i$ is given by a binomial distribution and is equal, asymptotically for large $n$, to

$$Prob(Y_r = i) \approx \tbinom{n_0}{i} {r_{kk}}^i (1 - r_{kk})^{n_0 - i}.$$

Since for $n \to \infty$ we have $r_{kk} \to 0$ and $r_{kk}\ n_0 \to k$, a constant, we can approximate the Binomial distribution by a Poisson distribution of mean $k$. This way the probability that a set formed at iteration one is of size $i$ is approximately equal, for large $n$, to $e^{-k} k^i / i!$.

We thus get that

$$Prob(Y_1 = 0) \approx e^{-k},$$

and

$$Prob(Y_1 \geq ek) \preceq \frac{e}{e-1} \frac{e^{-k}}{\sqrt{2\pi ek}}.$$

The former bound is straightforward to derive. The terms of the Poisson distribution with $i > ek$ give a geometric sequence with ratio less than one, as we can derive from,

$$\frac{\frac{e^{(-k)} k^i}{i!}}{\frac{e^{(-k)} k^{i+1}}{i+1!}} = \frac{i+1}{k} \geq \frac{ek+1}{k}.$$

For $k \geq 10$ we use Stirling's approximation formula for the factorial (i.e. $n! \approx (\frac{n}{e})^n \sqrt{2\pi n}$) to get the desired result. This concludes the proof of Theorem 3 stated in section 3.2.

### 3.10.2 Iteration two

#### Left tail

A set of vertices formed at iteration two is common to two sets formed at iteration one. Each of these two sets has size $i$ with probability given for large $n$ by a Poisson distribution of mean approximately equal to $k$.

We are ready to prove Theorem 4 stated in section 3.2.

**Proof of Theorem 4:** Let $A,\ B$ be two sets of sizes $i$ and $j$ respectively, formed at iteration one. A set of vertices $C$ formed at iteration two and common to $A$ and $B$ is of size $l$ with probability $q_l$ given by

$$q_l = \sum_{i,j} p_i p_j \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l}, \tag{3.3}$$

where $p_i$ and $p_j$ are the probabilities of having a set of size $i,\ j$ respectively at iteration one, and the set of common vertices is drawn from set $P$ according to procedure Select. We would first like to find a non-trivial upper bound for $q_0$, which is thus equal to

$$q_0 = \sum_{i,j} p_i p_j (1 - r_{ij})^{n_0} = \sum_{i,j<W} p_i p_j (1 - r_{ij})^{n_0} + \sum_{\text{either } i \text{ or } j \geq W} p_i p_j (1 - r_{ij})^{n_0}$$

We claim that we can restrict the range of the indices $i, j$ up to a quantity which is $o(\sqrt{nk})$ say, $W = \Theta(n^{1/3})$. This can be proven as follows. For such a value of $W$, the contribution of the second of the two sums into which $q_0$ was split is at most $1/\Theta(n^{2/3})$. This is due to Theorem 1 (with $g(X) = X^2$), the fact that $Prob(Y > t) = Prob(Y^2 \geq t^2)$ for a non-negative random variable $Y$, Propositions 2 and 3 and the bound $(1 - r_{ij}) \leq 1$. We get

$$\sum_{i \geq W} p_i \leq \frac{\mu^2 + v}{\Theta(n^{2/3})}.$$

Since by Propositions 2 and 3 we have that $\mu = \sum_i ip_i$ and $v = \sum_i (i-\mu)^2 p_i$ are dependent on $k$ and the iteration and independent of $n$ the claim follows. For $i, j < W$ we have that $r_{ij} \approx ij/(kn)$ and since $(1-1/n)^n \to 1/e$ we get that

$$\begin{aligned} q_0 &\approx \sum_{i,j<W} p_i p_j e^{\frac{-ij}{k}} \\ &\leq 2p_0 + \sum_{1\leq i,j<W} p_i p_j e^{\frac{-ij}{k}} \\ &\leq 2e^{-k} + \sum_{i,j\geq 1} p_i p_j e^{\frac{-ij}{k}}. \end{aligned}$$

This upper bound for $q_0$ is tight since for $i, j < W = o(\sqrt{kn})$ we have that

$$(1-r_{ij})^{n_0} \geq (1/e)^{\frac{n_0}{1/r_{ij}-1}} \approx e^{-ij/k}.$$

Since each of $p_i,\ p_j$ are given approximately, for large $n$, by a Poisson distribution of mean $k$ we have for large $n$ that

$$\begin{aligned} q_0 &\leq 2e^{-k} + \sum_{i,j} \frac{e^{(-2k)}k^i k^j}{i!j!} e^{\frac{-ij}{k}} \\ &\leq 2e^{-k} + \sum_i \frac{k^i e^{-2k}}{i!} \sum_j \frac{1}{j!}(ke^{(-i/k)})^j \\ &= 2e^{-k} + e^{-2k} \sum_i \frac{k^i}{i!} e^{ke^{-i/k}}, \end{aligned}$$

and thus,

$$q_0 \leq 2e^{-k} + e^{-2k} \sum_m \frac{k^m}{m!} e^{ke^{-m/k}}. \tag{3.4}$$

**Calculation of an upper bound for the sum in Equation (3.4)**

A non-trivial upper bound for Equation (3.4) involves finding the value $m = m_0$ that maximizes the general term of the sum there.

As soon as this value $m_0$ is obtained, we can bound each term with $m \leq m_0$ (or $m < k$, if $m_0 < k$) by the $m = m_0$ term. The remaining terms are bounded by the sum of the terms of a decreasing geometric sequence with ratio less than one.

We apply Stirling's approximation formula for the factorial and then take the derivative with respect to $m$ of the general term of the sum in Equation (3.4) and equate it to zero

$$-1/2 - \frac{m}{e^{m/k}} - m \log m + m \log k = 0.$$

All logarithms are to base $e$. We then try a solution for $m$ of the form $m = ck$, $c$ a constant to be determined, which gives

$$\frac{1}{2ck} + e^{-c} + \log c = 0,$$

and after ignoring the first term we get

$$e^{-c} + \log c = 0.$$

This equation is solved numerically and a solution is given by $c \approx 0.567$. Thus, a maximum for the general term of the sum of Equation (3.4) occurs at $m_0 = 0.567k$. For $k \geq 20$, it is $m_0 \geq 10$ so that the errors introduced by using Stirling's approximation formula earlier are small.

We can thus bound each of the first $k$ terms of the sum by the $m_0 = 0.567k$ term. All terms with $m > k$ give a decreasing progression with ratio at most $k/(k+1)$. Therefore, the sum of these terms is bounded by $k+1$ times the $m = k$ term. The first $k$ terms thus give an upper bound of

$$\frac{k^{1/2}}{1.887} e^{-0.544k},$$

where $(0.567k)!$ was approximated by Stirling's formula, while the remaining terms give

$$\frac{k+1}{2.506\sqrt{k}} e^{-0.632k}.$$

Both exponents $e^{-0.544k}$ and $e^{-0.632k}$ were derived by finding an approximate solution to an equation, and ignoring there an insignificant term. An exact bound for Equation (3.4) depends on the accuracy of these approximations. If we had not ignored the term $1/(2ck)$ in the equation that was to be solved, we would have gotten a different solution. Even for small values of $k$ (say, $k = 25$) the inclusion of such a term in that equation, would have only changed $c$ giving $c \approx 0.53$ but not the exponent $-0.544k$ in the upper bound. This implies that the bounds we got are as accurate as if we had solved $1/(2ck) + e^{-c} + \log c = 0$ for $k \geq 25$. Therefore, at iteration two, an approximation to the upper bound on the probability of having a set of size zero, asymptotically as $n \to \infty$, is

$$\frac{k^{1/2}}{1.887}e^{-0.544k} + \frac{k^{1/2}}{2.506}e^{-0.632k} + 2e^{-k}.$$

For large $k$ (say, $k \geq 25$) the most significant term is the first one. ■

### A Combinatorial Lemma

We prove here a simple combinatorial lemma, which we shall use later in estimating upper bounds for various probabilities.

**Lemma 2** *The number of pairs $(i, j)$ such that $ij \leq a$ is at most $aH_a$, where $H_a = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{a}$.*

**Proof:** The number of such pairs is at most

$$\sum_{i \leq a} \frac{a}{i} = a \sum_{i \leq a} \frac{1}{i} = aH_a \leq a(\log a + \gamma + O(\frac{1}{a}))$$

where $\gamma = 0.577...$, is Euler's constant. ■

One can show inductively, as a corollary to this lemma, that the number of tuples $(a_1, \ldots, a_{n+1})$ such that

$$a_1 \ldots a_{n+1} \leq a,$$

is at most

$$aH_a^n \leq a\left(\log a + \gamma + O(\frac{1}{a})\right)^n.$$

**Right Tail**

The probability of having a set of size $l$ formed at iteration two is given by Equation (3.3).

$$q_l = \sum_{i,j} p_i p_j \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l}.$$

We are interested in finding an upper bound for

$$q = \sum_{l>d} \sum_{i,j} p_i p_j \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l},$$

with $d = ak$, where $a = e^2$.

**Proof of Theorem 5:** We shall choose $a = e^2$ so that $d = e^2 k$ in the expression for $q$. Considerations similar to those for the left tail of the distribution of $Y_2$ show that indices $i, j$ may be restricted up to a quantity which is $o(\sqrt{nk})$ say, $\Theta(n^{1/3})$. The contributions to the sum for other ranges of $i$ and $j$ are $1/\Theta(n^{2/3})$ and tend to zero as $n \to \infty$.

In the expression for $q$ we expand $p_i,\ p_j$ to get

$$\begin{aligned} q &\approx \sum_{l>d} \sum_{i,j=o(\sqrt{kn})} p_i p_j \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l} \\ &\approx \sum_{l>d} \sum_{i,j} \frac{e^{(-2k)} k^i k^j}{i! j!} \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l} \\ &= \sum_{i,j} \sum_{l>d} \frac{e^{(-2k)} k^i k^j}{i! j!} \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l} \\ &= \sum_{ij \geq kd} \sum_{l>d} \frac{e^{(-2k)} k^i k^j}{i! j!} \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l} + \\ &\quad \sum_{ij < kd} \sum_{l>d} \frac{e^{(-2k)} k^i k^j}{i! j!} \binom{n_0}{l} {r_{ij}}^l (1 - r_{ij})^{n_0 - l}. \end{aligned}$$

**Case 1 ($ij < kd$):** We first examine the second term of the sum above. Since $i, j$ are $o(\sqrt{nk})$ we get that $r_{ij} \approx ij/kn$ and therefore $r_{ij} n_0 \approx ij/k$. We use the bounds due to Angluin and Valiant [4] for the right tail of the binomial distribution. These bounds hold for $ak/2 < ij/k$. If $ij/k \leq ak/2$, we approximate the binomial distribution by a Poisson distribution of constant bounded mean $ij/k$ and use this approximation to find an upper bound for the sum involved.

For the second term of the sum into which the expression for $q$ was broken we have that $ij/k < d$ and all terms with $l > d$ are summed up. By the Angluin-Valiant bounds on the right tail of the binomial distribution we get the following.

**Case 1a:**

$$\begin{aligned}\sum_{\frac{ak^2}{2}<ij<kd} \sum_{l>d} \frac{e^{(-2k)}k^ik^j}{i!j!}\binom{n_0}{l}r_{ij}{}^l(1-r_{ij})^{n_0-l} &\leq \sum_{\frac{ak^2}{2}<ij<kd} \frac{e^{(-2k)}k^ik^j}{i!j!}e^{-\frac{1}{3}\frac{ij}{k}(\frac{a^2k^4}{i^2j^2}+1-\frac{2ak^2}{ij})} \\ &= e^{\frac{2}{3}ak} \sum_{\frac{ak^2}{2}<ij<kd} \frac{e^{(-2k)}k^ik^j}{i!j!}e^{-\frac{a^2k^3}{3ij}-\frac{ij}{3k}}. \qquad (3.5)\end{aligned}$$

**Case 1b:** For the case $ij/k \leq ak/2$ we get by approximating a Binomial distribution with a Poisson distribution:

$$\begin{aligned}\sum_{ij\leq\frac{ak^2}{2}} \sum_{l>d} \frac{e^{(-2k)}k^ik^j}{i!j!}\binom{n_0}{l}r_{ij}{}^l(1-r_{ij})^{n_0-l} &\leq \sum_{ij\leq\frac{ak^2}{2}} \frac{e^{(-2k)}k^ik^j}{i!j!} \sum_{l>ak} \frac{e^{(-\frac{ij}{k})}(\frac{ij}{k})^l}{l!} \\ &\leq 2\sum_{ij\leq\frac{ak^2}{2}} \frac{e^{(-2k)}k^ik^j}{i!j!} \frac{e^{(-\frac{ij}{k})}(\frac{ij}{k})^{ak}}{ak!} \\ &\approx \frac{2e^{-2k}}{\sqrt{2\pi ak}} \sum_{ij\leq\frac{ak^2}{2}} \frac{k^ik^j}{i!j!}e^{-\frac{ij}{k}}(\frac{ije}{ak^2})^{ak}.\end{aligned}$$

The second inequality holds because the sum over $l$ is over terms of a geometrically decreasing sequence with ratio at most $1/2$, since $a \geq 2$. We approximate the factorial using Stirling's formula to get the last term.

**Case 2:** The first term in the derived expression for $q$ gives,

$$\sum_{ij>kd} \sum_{l>d} \frac{e^{(-2k)}k^ik^j}{i!j!}\binom{n_0}{l}r_{ij}{}^l(1-r_{ij})^{n_0-l} \leq \sum_{ij>kd} \frac{e^{(-2k)}k^ik^j}{i!j!}. \qquad (3.6)$$

An upper bound for $q$ is thus the sum of the upper bounds derived in the three previous cases.

**Calculation of upper bounds for the three sums**

We get here further bounds for the ones derived in each of the three previous cases. We use Lemma 2 to bound Equation (3.5) since the product of $i,\ j$ there, is bounded from above by $kd$. Following the method exhibited earlier for the left tail of $Y_2$ we shall find the values of $i, j$ for which the general term of the sum in Equation (3.5) attains its maximum value. We take the derivative with respect to $i$ of this general term, assume that both $i!,\ j!$ can be approximated using Stirling's formula, and equate the result to zero to get

$$-j/2 - ij^2/(3k) + ak^3/(3i) - ij\log i + ij\log k = 0.$$

We do the same for the derivative with respect to $j$. For $a = e^2$, the general term of the sum in Equation 3.5 is maximized at $i = j \approx 2.2625k$. We now use Lemma 2 and Equation (3.5) becomes

$$\begin{aligned} e^{\frac{2}{3}ak} \sum_{ij<kd} \tfrac{e^{(-2k)}k^ik^j}{i!j!} e^{-\frac{a^2k^3}{3ij} - \frac{ij}{3k}} &\leq e^{-2k} e^{\frac{2}{3}ak} dk(\log(dk) + \gamma) \\ &\cdot \frac{k^{2\ 2.2625k}}{((2.2625\ k)!)^2} e^{-\frac{a^2k^3}{3\ 2.2625^2\ k^2}} e^{-\frac{2.2625^2\ k^2}{3k}} \\ &\approx 0.52(2\log k + 2 + \gamma)ke^{-1.50512k}. \end{aligned}$$

We proceed to examining the sum of Case (1b).

$$\frac{2e^{-2k}}{\sqrt{2\pi ak}} \sum_{ij<\frac{ak^2}{2}} \tfrac{k^ik^j}{i!j!} e^{-\frac{ij}{k}} \left(\frac{ije}{ak^2}\right)^{ak}.$$

By the symmetry of the general term of the sum we look for values of $i, j$ such that $i = j = ck$. The first derivative with respect to $c$ of the logarithm of the general term of the sum when equated to zero gives

$$-c\log c - c^2 + a = 0.$$

For $a = e^2$ and given that $c^2 < a/2$ so that $ij < ak^2/2$ we get that the general term of the sum is maximized at $c = 1.92$. We then get the following approximation to an upper bound for Case (1b).

$$0.18k^{1/2}(1.31 + 2\log k + \gamma)e^{-2.10k}.$$

This contribution is negligible compared to the one derived from Equation (3.5).

The quality of the bounds we propose depends on the quality of the approximations we use at various stages of their derivations. In some cases it seems that these approximations can be considered fairly accurate for values of $k$ that are considered interesting, that is $k \geq 25$. An upper bound for Equation (3.6) is derived below.

$$\sum_{ij>kd} \frac{e^{(-2k)}k^i k^j}{i!j!} = \sum_{ij>ak^2} \frac{e^{(-2k)}k^i k^j}{i!j!}.$$

In case both indices $i, j$ are greater than $\sqrt{a}k$ we get that

$$\begin{aligned}
\sum_{i,j>\sqrt{a}k} \frac{e^{(-2k)}k^i k^j}{i!j!} &= \sum_{i>\sqrt{a}k} \frac{e^{(-2k)}k^i}{i!} \sum_{j>\sqrt{a}k} \frac{k^j}{j!} \\
&\leq 4e^{-2k} \frac{k^{\sqrt{a}k}}{(\sqrt{a}k)!} \frac{k^{\sqrt{a}k}}{(\sqrt{a}k)!} \\
&\approx \frac{2e^{-2k}}{\pi\sqrt{a}k} \left(\frac{e}{\sqrt{a}}\right)^{2\sqrt{a}k}.
\end{aligned}$$

The inner sum of $k^j/j!$ for $j > \sqrt{a}k$ when $a = e^2$ gives a decreasing progression with ratio $1/\sqrt{a}$ and therefore the sum is bounded above by twice its first term. The same holds for index $j$ as well. Use of Stirling's approximation formula gives the last expression.

In case $i$ and $j$ are not both greater than $\sqrt{a}k$ let us assume that $i < \sqrt{a}k$ and $j > ak^2/i$. The symmetric case is included by multiplying all expression to follow by two. Since $1/i > 1/(\sqrt{a}k)$ and $j > ak^2/i$ we get that $j > \sqrt{a}k$. We then have

$$2\sum_{i\leq\sqrt{a}k} \sum_{j>\sqrt{a}k} \frac{e^{(-2k)}k^i k^j}{i!j!} \leq 2\sum_{i} \frac{e^{(-2k)}k^i}{i!} 2\frac{k^{\sqrt{a}k}}{(\sqrt{a}k)!}$$

$$\begin{aligned} &\approx \quad 2 \sum_{i \geq \sqrt{a}k} \tfrac{e^{(-2k)}k^i}{i!} 2 \frac{1}{\sqrt{2\pi\sqrt{a}k}} \left(\frac{e}{\sqrt{a}}\right)^{\sqrt{a}k} \\ &\leq \quad \frac{4e^{-k}}{\sqrt{2\pi\sqrt{a}k}} \left(\frac{e}{\sqrt{a}}\right)^{\sqrt{a}k}, \end{aligned}$$

for $a > 2$.

Therefore, a numerical approximation to the upper bound for the probability $q$ that holds for $n \to \infty$, is given by

$$\begin{aligned} q = Pr(Y > e^2 k) \quad &\preceq \quad 1.04(\log k + 1.29)ke^{-1.50512k} \\ &+ \quad 0.18k^{1/2}(1.31 + 2\log k + \gamma)e^{-2.10k} \\ &+ \quad \frac{2e^{-2k}}{\pi e k} + \frac{4e^{-k}}{\sqrt{2\pi e k}}, \end{aligned}$$

after substituting for $a = e^2$ in the formulae above. ■

For iterations greater than two, similar bounds can be derived if we use the convergence of Bernstein polynomials (see [17] for a definition of these polynomials). Since the derived bounds are in the form of complicated sums we have not included this discussion in this chapter, but is examined elsewhere in detail.

## 3.11 Upper bounds using derivations of Hoeffding's inequalities

In this section, let $\lg x$ be the logarithm of $x$ to base two.

### 3.11.1 Left tail

We examine below the left tail of the distribution of the random variable $Y_r$ introduced in section 3.2.

The following definition will be needed.

**Definition 8** *In the class $G_{n,1/(k\cdot n)^{1/2}}$ of random graphs, where $k$ is a constant, a set formed* *at iteration $r$ is* declared *to be of size at most $(1-a)^{2^r-1}k$, where $0<a<1$, if either this set is of size at most $(1-a)^{2^r-1}k$, or any of the sets formed at some intermediate iteration $m<r$ is of size at most $(1-a)^{2^m-1}k$.*

The probability that a set is of size at most $(1-a)^{2^r-1}k$ is bounded above by the probability that this set is declared to be of size at most $(1-a)^{2^r-1}k$. The latter probability will be easier to find.

In the tree of iteration $r$, at iteration zero $2^r$ sets each of size $k$ are paired in $2^{r-1}$ pairs to form $2^{r-1}$ sets of vertices at iteration one. We form sets according to Select and Definition 6. Then, a vertex of set $P$ is common to two such sets of size exactly $k$ with probability $r_{kk}$ given by Equation (3.1). The size of a set formed at iteration one follows a binomial distribution with parameters $n_0$ and $r_{kk}$. We apply the Angluin-Valiant bounds for the left tail of the binomial distribution, since $r_{kk}n_0 \approx k$, to get that the probability of having a set of size less than $(1-a)k$ formed at iteration one is at most $e^{-\frac{1}{2}a^2k}$, for large $n$. Hence, the probability that any of the $2^{r-1}$ sets of iteration one is of size at most $(1-a)k$ is for large $n$ at most $2^{r-1}e^{-\frac{1}{2}a^2k}$. We shall call this term $p_1$.

Let two sets $A$, $B$ so formed at iteration one be of sizes at least $(1-a)k$ each. The probability that a vertex in $P$ is common to both $A$ and $B$ is at least, by Equation (3.1), $r_{(1-a)k\ (1-a)k}$. Thus at iteration two the mean $r_{(1-a)k\ (1-a)k}n_0$ of the size of set $C$ is by Proposition 2 and in the limit $n\to\infty$ at least $(1-a)^2k$. Thus set $C$ by the Angluin-Valiant bounds is of size at most $(1-a)^3k$, which is at most $(1-a)$ times the lower bound of its mean, with probability $e^{-\frac{1}{2}a^2(1-a)^2k}$ in the limit $n\to\infty$.

In the tree of iteration $r$ the $2^{r-1}$ sets formed at iteration one combine in pairs in some order to give $2^{r-2}$ sets of vertices thus formed at iteration two. For $n\to\infty$ with probability at least $1-p_1$ each of the $2^{r-1}$ sets formed at iteration one is of size at least $(1-a)k$. Taking into consideration the discussion of the previous paragraph we get that the probability that any of the $2^{r-2}$ sets formed at iteration two is of size at most $(1-a)^3k$, is at most $2^{r-2}e^{-\frac{1}{2}a^2(1-a)^2k}$. We call this last term $p_2$.

Therefore, with probability at least $1-p_1-p_2$ for $n\to\infty$ all sets formed at iteration one are of size at least $(1-a)k$ and all sets formed at iteration two are of size at least $(1-a)^2k$. Otherwise we declare the set that is to be formed at iteration $r$ to be of size at most $(1-a)^{2^r-1}k$.

Inductively at iteration $l \leq r$, given that all $2^{r-l}$ sets formed at iteration $l-1$ are of size at least $(1-a)^{2^{l-1}-1} \cdot k$, the size of a set common to two such sets and thus formed at iteration $l$ has mean at least $(1-a)^{2^l-2}k$. Hence, the probability that this set is of size at most $(1-a)^{2^l-1} \cdot k$ is, by the Angluin-Valiant bounds, at most $e^{-\frac{1}{2}a^2(1-a)^{2^l-2}k}$. We then conclude that in the tree of iteration $r$, the probability that any of the $2^{r-l}$ sets formed at iteration $l$ is of size at most $(1-a)^{2^l-1} \cdot k$ is at most $2^{r-l}e^{-\frac{1}{2}a^2(1-a)^{2^l-2}k}$. We call this term $p_l$. Consequently, with probability at least $1-p_1-\ldots-p_l$ all $2^{r-i}$ sets at iteration $i \leq l$ are of size at least $(1-a)^{2^i-1} \cdot k$, otherwise the unique set at iteration $r$ is declared to have size at most $(1-a)^{2^r-1}k$.

For $l = r$ we we get the bound given in Theorem 6 for large $n$. Furthermore, it is an upper bound of the probability that a set at iteration $r$ is of size at most $(1-a)^{2^r-1} \cdot k$ and hence, Theorem 6 is derived.

For a constant $a$, the term $\lambda = r-1$ in the Theorem becomes equal to $1/e$ , for

$$r = \lg\left(\frac{\log(a^2\ k/2)}{-\log(1-a)} + 2\right),$$

and $r$ can thus grow as much as $O(\text{lglg}k)$ before this expression gives trivial bounds.

### 3.11.2 Right tail

The discussion of the right tail of $Y_r$ follows similarly that of the left tail but we now use the Angluin-Valiant bounds for the right tail of the binomial distribution.

The following definition will be used.

**Definition 9** *In the class $G_{n,1/(k\cdot n)^{1/2}}$ of random graphs, where $k$ is a constant, a set formed at iteration $r$ is* declared *to be of size at least $(1+a)^{2^r-1}k$, where $0 < a < 1$, if either this set is of size at least $(1+a)^{2^r-1}k$, or any of the sets formed at some intermediate iteration*

*$m < r$ is of size at least $(1+a)^{2^m-1}k$.*

Then the probability that a set is of size at least $(1+a)^{2^r-1}k$ is bounded above by the probability that this set is declared to be of size at least so.

Similarly to Theorem 6 we get Theorem 7 stated in section 3.2.

Since $a$ is a positive number, for a bound $(1+a)^{2^r-1} \cdot k$ that is polynomial in $k$ we need to have $r = \Theta(\lg \lg k)$.

For $a > 1$, constant $r$, and $(1+a)^{2^r-1} \cdot k$ that depends on $n$, one will have to rely not on the Angluin-Valiant bounds but say, on the DeMoivre-Laplace Theorem or its extensions (see [10],pages 13-14) presented as Theorem 2.

By repeating our claims that resulted in the proof of Theorem 7, using $u$ in place of $(1+a)$, and the DeMoivre-Laplace bound instead of the Angluin-Valiant bounds, we get the variant of Theorem 7, expressed by Theorem 8.

In Theorem 8 we take $u^{2^r-1} \cdot k = \sqrt{kn}$. This requires $u = (\sqrt{n/k})^{1/2^{r-1}}$. For $r$ a constant it is straightforward that the upper bound given by Theorem 8 decreases exponentially fast with $n$. This in turn shows that a sum of the form $\sum_{i \geq \sqrt{kn}} p_i$, given at the end of the proof of Proposition 3 is at most $1/n^4$ for sufficiently large $n$. The tight bound claimed in Proposition 4 follows easily, by using the previous bound on the sum $\sum_{i \geq \sqrt{kn}} p_i$ and the techniques in the proof of Proposition 2 to give the stronger Proposition 4.

## 3.12 Preample for the asymptotic results for large $k$

In all previous sections we examined the distribution of the random variable $Y_r$ in $G_{n,1/(k \cdot n)^{1/2}}$ random graphs for a constant $k$. In this section we examine the distribution of $Y_r$ for large values of $k$ in addition to having $n \to \infty$. Thus the $O$ and $\Theta$ notations to be used in the following sections and in the Theorems related to these sections already stated are for $n \to \infty$ and $k \to \infty$. The replication factor $k$ remains in this discussion independent of $n$.

### 3.12.1 Some results from Probability Theory

We first present some results from Probability Theory. For more information we refer to either an introductory book in probability theory or to say, [7].

Let $\{U_1, U_2, \ldots, U_m, \ldots\}$, $\{V_1, V_2, \ldots, V_m, \ldots\}$ be sequences of random variables.

**Definition 10 (Convergence in Distribution)** *The sequence $\{U_1, U_2, \ldots, U_m, \ldots\}$ converges in distribution to the cumulative distribution function $F(u)$ if for all $u$ that are continuity points of $F(u)$*

$$\lim_{m\to\infty} Prob(U_m \leq u) = F(u)$$

Equivalently one may require that $\mathcal{E}[h(U_m)]$ tends to $\mathcal{E}[h(U)]$, where $U$ is a random variable having probability distribution function $F(u)$ and $h(u)$ is any bounded continuous function.

As an example, consider the Central Limit Theorem. For a positive integer $n$, $X_1, \ldots, X_n$ independent and identically distributed random variables with mean $\mu$ and variance $\sigma^2$, it states that the cumulative distribution function of $U_n = (\overline{X}_n - \mathcal{E}[\overline{X}_n])/(\sigma/\sqrt{n})$ converges in distribution to $\Phi(u)$ as $n$ approaches infinity. $\Phi()$ denotes the cumulative distribution function of the standardized normal distribution $N(0,1)$. Therefore $\overline{X}_n$ is approximately distributed $N(\mu, \sigma^2/n)$, and thus asymptotically normal.

The notation $U \simeq F$ will be used in such convergence cases. We shall thus write $X_m \simeq N(\mu, \sigma^2)$ to indicate that the sequence $\{X_m\}$ is asymptotically normal, that is it converges in distribution to the normal distribution with mean $\mu$ and variance $\sigma^2$. We shall also write $U \sim F$, $U \sim V$ to indicate that $U$ has distribution function $F$ and $U$ and $V$ have the same distribution.

For sequence $U_m$ we define the relation "$\sim as$" and thus write $U_m \sim asN(\mu_m, \sigma_m^2)$ to indicate that $U_m$ is asymptotically normal, i.e. $(U_m - \mu_m)/\sigma_m$ converges in distribution to the standard normal distribution.

In many cases we shall use the "=" sign to indicate a limit but such a usage will be evident from the context of the relevant arguments.

### 3.12.2 Poisson Distribution

Let $X$ be a random variable that follows, possibly in the limit, a Poisson distribution with mean $k$. We examine below the limiting distribution of the product $aX$, where $a$ is a positive real number, for large $k$. We shall show that for large $k$, $aX$ has a normal approximation.

The exponential generating function (e.g.f. in short) of a Poisson random variable with expectation $k$ (see, say [17]) is given by the following equation.

$$\mathcal{E}[e^{t\,X}] = e^{-k(1-e^t)}.$$

In order to examine the limiting distribution of $aX$ we standardize it and thus examine the e.g.f. of $(aX - \mathcal{E}[aX])/\sqrt{var(aX)}$.

$$\begin{aligned}
\mathcal{E}[e^{t\,a\,\frac{X-k}{a\,\sqrt{k}}}] &= e^{-t\,\sqrt{k}}\;\mathcal{E}[e^{t\,\frac{X}{\sqrt{k}}}] \\
&= e^{-t\,\sqrt{k}}\;e^{-k(1-e^{t/\sqrt{k}})} \\
&= e^{-t\,\sqrt{k}}\;e^{-k(1-(1+\frac{t/\sqrt{k}}{1!}+\frac{(t/\sqrt{k})^2}{2!}+\ldots+\frac{(t/\sqrt{k})^j}{j!}+\ldots))} \\
&= e^{(1/2)t^2+t^3/(3\sqrt{k})+O(1/k)}.
\end{aligned}$$

For large $k$ all terms of the exponent of the e.g.f. of $(aX - \mathcal{E}[aX])/\sqrt{var(aX)}$ other than the first one tend to zero. Hence, the standardized random variable has an exponential generating function approaching in the limit for large $k$ that of the standard normal distribution. Therefore, for large $k$,

$$\frac{aX - ak}{a\sqrt{k}} \simeq N(0,1),$$

and consequently,

$$aX \sim asN(a\,k, a^2\,k).$$

## 3.13 Results for large $k$

### 3.13.1 Distribution of the Product of two normal random variables

Let $X_1$, $X_2$ be two independent random variables which follow, possibly in the limit, the same distribution $N(\mu, \sigma^2)$. We are interested in the behavior of the product $X_1 X_2$.

**Fact 8** *Let $X_1$, $X_2$ be two independent random variables that follow, possibly in the limit, a normal $N(\mu, \sigma^2)$ distribution. Then,*

$$\mathcal{E}[X_1 X_2] = \mu^2,$$

*and*

$$\mathcal{E}[X_1^2 X_2^2] = (\mu^2 + \sigma^2)^2.$$

**Proof:** The central moments $\mathcal{E}[(X_1 X_2)^r]$ can be derived easily by considering the exponential generating function of a bivariate $X = (X_1, X_2)$ normal distribution, where the correlation between $X_1$ and $X_2$ is zero since they are independent, and using the fact (see say, [7], page 131) that $\mathcal{E}[(X_1 X_2)^r]$ is the coefficient of $(t_1^r \ t_2^r)/(r! \ r!)$ in the expansion of $m_X(t_1, t_2) = \mathcal{E}[e^{tX}] = \mathcal{E}[e^{t_1 X_1 + t_2 X_2}]$. An expansion of the exponential generating function of a bivariate normal distribution, remembering that

$$\mathcal{E}[e^{t_1 X_1 + t_2 X_2}] = e^{\mu\ t_1 + 1/2 \sigma^2 t_1^2 + \mu t_2 + 1/2 \sigma^2 t_2^2},$$

gives the following results after some calculations.

$$\mathcal{E}[X_1 X_2] = \mu^2,$$

$$\mathcal{E}[X_1^2 X_2^2] = (\mu^2 + \sigma^2)^2,$$

and also,

$$\mathcal{E}[X_1^3 X_2^3] = \mu^2 (\mu^2 + 3\ \sigma^2)^2.$$

[In case $X_1, X_2$ are in the limit $N(\mu, \sigma^2)$, the equality signs in the three formulae above should be interpreted as limits.] ■

Although this way we can find the moments of the product $X_1X_2$, we cannot get its exponential generating function easily.

We now give a direct method for finding the e.g.f. of $X = X_1X_2$. This is expressed by Theorem 9 of section 3.2 proven below.

**Proof of Theorem 9:** For either of the two independent random variables $X_1$, $X_2$ we have possibly in the limit for large $k$ that $\mathcal{E}[e^{tX_i}] = e^{\mu t + 1/2\sigma^2 t^2}$, with $\mu = \Theta(k)$, $\sigma^2 = \Theta(k)$.

Since $X_1, X_2$ are also independent, we get the following.

$$
\begin{aligned}
m_X(t) &= \mathcal{E}[e^{tX}] \\
&= \mathcal{E}[e^{tX_1X_2}] \\
&= \mathcal{E}[e^{(tX_1)\,X_2}] \\
&= \mathcal{E}[e^{\mu t X_1 + \frac{1}{2}\sigma^2 t^2 X_1^2}] \\
&= \frac{1}{\sqrt{2\pi}\,\sigma} \int_{-\infty}^{\infty} e^{\mu\, t\, x + \frac{1}{2}\,\sigma^2\, t^2\, x^2} e^{-\frac{(x-\mu)^2}{2\,\sigma^2}}\, dx \\
&= \frac{1}{\sqrt{1-\sigma^4\, t^2}}\, \frac{1}{\sqrt{2\,\pi}\, \frac{\sigma}{\sqrt{1-\sigma^4\, t^2}}} \int_{-\infty}^{\infty} e^{-\frac{\left(x - \frac{\mu+\sigma^2\,\mu\, t}{1-\sigma^4\, t^2}\right)^2}{2\,\frac{\sigma^2}{1-\sigma^4\, t^2}}}\, e^{\frac{\mu^2\, t\,\left(\sigma^2\, t+1\right)}{(1-\sigma^4\, t^2)}}\, dx \\
&= \frac{1}{\sqrt{1-\sigma^4\, t^2}}\, e^{\frac{\mu^2\, t\,\left(\sigma^2\, t+1\right)}{(1-\sigma^4\, t^2)}}\, \frac{1}{\sqrt{2\,\pi}\, \frac{\sigma}{\sqrt{1-\sigma^4\, t^2}}} \int_{-\infty}^{\infty} e^{-\frac{\left(x - \frac{\mu+\sigma^2\,\mu\, t}{1-\sigma^4\, t^2}\right)^2}{2\,\frac{\sigma^2}{1-\sigma^4\, t^2}}}\, dx \\
&= \frac{1}{\sqrt{1-\sigma^4\, t^2}}\, e^{\frac{\mu^2\, t\,\left(\sigma^2\, t+1\right)}{(1-\sigma^4\, t^2)}}.
\end{aligned}
$$

Since $\mathcal{E}[X^i] = m_X^{(i)}(0)$, the $i$th derivative of $m_X(t)$ at $t = 0$, we get after a number of calculations that

$$\mathcal{E}[X] = \mu^2,$$

$$\mathcal{E}[X^2] = \mu^4 + 2\,\mu^2\,\sigma^2 + \sigma^4,$$

$$\mathcal{E}[X^3] = \mu^6 + 6\,\mu^4\,\sigma^2 + 9\,\mu_2\,\sigma^4.$$

Therefore, the variance of $X$ is $var(X) = \mathcal{E}[X^2] - \mathcal{E}^2[X] = \sigma^4 + 2\,\mu^2\,\sigma^2$.

We are interested in examining the limiting behavior of $X$ for large $k$ for the case where both $\mu$ and $\sigma^2$ are such that $\mu = \Theta(k)$, $\sigma^2 = \Theta(k)$, where $k$ is some parameter of interest. We standardize $X$ and examine the e.g.f. of $(X - \mathcal{E}[X])/\sqrt{var(X)}$ by taking into consideration the e.g.f. of $X$ derived earlier.

$$\begin{aligned}
\mathcal{E}[e^{t\,\frac{X-\mu^2}{\sqrt{\sigma^4+2\,\mu^2\,\sigma^2}}}] &= e^{-t\frac{\mu^2}{\sqrt{\sigma^4+2\,\mu^2\,\sigma^2}}}\,\mathcal{E}[e^{\frac{t\,X_1\,X_2}{\sqrt{\sigma^4+2\,\mu^2\,\sigma^2}}}] \\
&= e^{-t\frac{\mu^2}{\sqrt{\sigma^4+2\,\mu^2\,\sigma^2}}}\,\frac{1}{\left(1-\sigma^4\,\frac{t^2}{\sigma^4+2\,\mu^2\,\sigma^2}\right)^{1/2}}\,e^{\frac{1}{\left(1-\sigma^4\,\frac{t^2}{\sigma^4+2\,\mu^2\,\sigma^2}\right)}\,\frac{\mu^2\,\sigma^2\,t^2}{\sigma^4+2\,\mu^2\,\sigma^2}} \\
&\cdot\; e^{\frac{1}{\left(1-\sigma^4\,\frac{t^2}{\sigma^4+2\,\mu^2\,\sigma^2}\right)}\,\frac{\mu^2\,t}{\sqrt{\sigma^4+2\,\mu^2\,\sigma^2}}}.
\end{aligned}$$

For $\mu = \Theta(k)$ and $\sigma^2 = \Theta(k)$ and for large $k$ we have the convergences $(\sigma^4)/(2\mu^2\,\sigma^2+\sigma^4) \to 0$, $(\mu^2\,\sigma^2)/(2\mu^2\,\sigma^2 + \sigma^4) \to 1/2$ thus getting

$$\mathcal{E}[e^{t\,\frac{X-\mu^2}{\sqrt{\sigma^4+2\,\mu^2\,\sigma^2}}}] \to e^{1/2t^2}.$$

Hence, in the limit for large $k$ the standardized random variable $X$ converges to the standard Normal distribution $N(0,1)$, which implies $X \sim asN(\mu^2, \sigma^4 + 2\,\mu^2\,\sigma^2)$. ■

Theorem 9 also holds if $X_1, X_2 \sim asN(\mu, \sigma^2)$ by interpreting the equalities in the proof of the theorem in the limit for large $k$.

Suppose we have two sets $A$, $B$ of sizes $x_1$, $x_2$ respectively. The sizes of the two sets can be treated as random variables and let $X_1$ (respectively $X_2$) be the random variable that takes with some probability the value $x_1$ (respectively $x_2$) when set $A$ (respectively $B$) is of size $x_1$ (respectively $x_2$).

We start examining again our graph theoretic problem.

**Example 1.** When sets $A$, $B$ of the previous paragraph are formed at iteration one of the tree of iteration $r$, with constant $r$, then the discussion prior to this section indicates that $X_i$, $i = 1, 2$, are binomial random variables with density $B(k; n_0, r_{kk})$ and identically distributed with random variable $Y_1$ of sections 3.2 and 3.9. Such a binomial random variable has a normal approximation of mean and variance $k$ for large $k$ and $n \to \infty$. Theorem 9 then implies that $X_1 X_2 \sim asN(k^2, 2k^3 + k^2)$. We have by Equation (3.1) that $r_{x_1,x_2} \approx (x_1 x_2)/(kn)$, if $x_1, x_2 = o(\sqrt{n})$. Then it is $x_1 x_2 = o(n)$. Let $X = X_1 X_2$. The probability that set $C$ formed at iteration two and common to $A$, $B$ has size $j$ is given by the following expression.

$$Prob(Y = j) \approx \sum_i Prob(X = i) \ \binom{n_0}{j} (\tfrac{i}{k\,n})^j (1 - \tfrac{i}{k\,n})^{n_0 - j},$$

where $Y$ is the random variable that represents the size of $C$. We examine the exponential generating function of $Y$ by showing Proposition 8 below.

**Proposition 8** *Let $X$ be the random variable defined in Theorem 9 that in the limit for large $k$ has a normal approximation $X \sim asN(k^2, \sigma^2)$, where $\sigma^2$ depends on $k$ only. Then the random variable $Y$ such that,*

$$Prob(Y = j) \approx \sum_i Prob(X = i) \ \binom{n_0}{j} (\tfrac{i}{k\,n})^j (1 - \tfrac{i}{k\,n})^{n_0 - j}$$

*has an exponential generating function that is given, for large $k$ and $n \to \infty$, by the following expression.*

$$\mathcal{E}[e^{t\,Y}] \approx e^{(\frac{\frac{\sigma^2}{2} - k^3}{k^2}) + \frac{1}{k^2}(k^3 - \sigma^2)\, e^t + \frac{\sigma^2}{2\,k^2}\, e^{2t}}.$$

*Hence, $\mathcal{E}[Y] \approx k$ and $var(Y) \approx k + \sigma^2/k^2$.*

**Proof of Proposition 8:** The approximation $r_{x_1,x_2} \approx \frac{x_1\,x_2}{k\,n}$ holds when both $x_1, x_2$ are $o(\sqrt{n})$. For $i = \Omega(\sqrt{n})$, the sum over such $i$ of $Prob(X = i)$ is $O(e^{-n})$. This can be proved as follows. By the limiting behavior of $X$ we have

$$\sum_{i=\Omega(\sqrt{n})} Prob(X=i) \approx \frac{1}{\sqrt{2\pi}} \int_{\frac{\sqrt{n}-k^2}{\sigma}}^{\infty} e^{-x^2/2}\, dx = 1 - \Phi(\frac{\sqrt{n}-k^2}{\sigma}).$$

We now claim (see [10], page 9) that for $x \to \infty$,

$$1 - \Phi(x) \approx \frac{1}{\sqrt{2\pi}} \frac{1}{x} e^{-x^2/2},$$

A substitution for $x = (\sqrt{n} - k)/\sigma$ yields that in the limit $n \to \infty$ and for large $k$ the sum $\sum_{i=\Omega(\sqrt{n})} Prob(X = i)$ tends to zero exponentially fast.

In the following discussion we shall examine the e.g.f. of $Y$ in the limit for large $k$ and $n \to \infty$. The expectation used below are the limiting ones for the random variables under consideration.

We now calculate the exponential generating function $m_Y(t) = \mathcal{E}[e^{t\,Y}]$ of $Y$. We have that $X \sim asN(k^2, \sigma^2)$. We also approximate the Binomial term in the expression for $Prob(Y = j)$, by a Poisson distribution of mean $i/k$. After some calculations similar to the ones that led to the calculation of $\mathcal{E}[e^{t\,X}]$ in the Proof of Theorem 9, we get the following.

$$\mathcal{E}[e^{t\,Y}] \approx e^{(\frac{\frac{\sigma^2}{2}-k^3}{k^2}) + \frac{1}{k^2}(k^3-\sigma^2)\, e^t + \frac{\sigma^2}{2\,k^2}\, e^{2t}}.$$

From this expression we easily get that in the limit for large $k$

$$\mathcal{E}(Y) = m'_Y(0) \approx k,$$

and

$$\begin{aligned} var(Y) &= -m'^2_Y(0) + m''_Y(0) \\ &\approx -k^2 + k^2 + k + \frac{\sigma^2}{k^2} \\ &= k + \frac{\sigma^2}{k^2}. \end{aligned}$$

■

**Example 2.** From the discussion of Example 1 we get that at iteration two we have $var(X) = \sigma^2 \approx 2k^3 + k^2$ thus getting by Proposition 8 that $\mathcal{E}[Y] \approx k$ and $var(Y) \approx 3\ k + 1$.

We then standardize $Y$ and examine the generating function for the standardized random variable $Y' = (Y - k)/(\sqrt{k + \frac{\sigma^2}{k^2}})$. Using the expansion $e^t = \sum_i t^i/i!$, we get the following

$$\mathcal{E}[e^{t\,Y'}] \approx e^{\frac{1}{2}t^2 + O(\frac{1}{\sqrt{k}}t^3)}.$$

For large enough $k$ the exponent of $e$ in $\mathcal{E}[e^{t\,Y'}]$ is asymptotically $1/2t^2$, that of the standard normal distribution. Thus for large $k$, $Y'$ is approximately $N(0, 1)$ and consequently, $Y$ is approximately $N(k, 3k + 1)$ at iteration two.

This allows us to proceed iteratively to the third iteration. Random variable $Y$ can similarly be shown to be approximately $N(k, 7k + 11)$.

We shall show that random variable $Y$ previously defined at a constant iteration $r$ approximates, in the limit for large $k$, $N(k, (2^r - 1)k + O(1))$. The term $O(1)$ describes contributions which are due to the propagation in higher iterations of the additive one in the variance $3k+1$ of iteration two. For large iterations but small $k$ it may be the case that this constant is the overall dominant term in the variance. We must thus bear in mind that the results we claim in this section hold for large $k$ only. These results are summarized in Theorem 10 whose proof follows. We shall call random variable $Y$, just as we did for the case of constant $k$, $Y_r$.

**Proof of Theorem 10:** The proof is by induction on the iteration. The basis of the induction has been proved in Example 1, and can also be derived from section 3.10.

In the inductive step we take two sets $A$ and $B$ formed at iteration $r$, with sizes represented by random variables, which are identically distributed and approximately $N(k, (2^r - 1)k + O(1))$. Let us call $b_r^2 = (2^r - 1)k + O(1)$. The product of these random variables, by Theorem 9, can be thus approximated by a normal random variable $N(k^2, b_r^2 + 2\ k^2\ b_r) = N(k^2, \sigma_r^2)$.

Random variable $Y_{r+1}$ that represents the size of set $C$ formed at iteration $r + 1$ and common to $A$ and $B$ has, by Proposition 8, expectation $\mathcal{E}[Y_{r+1}] \approx k$ independent of the iteration and variance $var(Y_{r+1}) \approx k + (b_r^4 + 2\ k^2\ b_r^2)/k^2$. Substituting for $b_r^2$ we get that in the limit for large $k$ and constant $r$, $var(Y_{r+1}) = (2^{r+1} - 1)k + O(1)$.

The hidden constants grow exponentially fast with the iteration. That is, for $b_r^2 = (2^r - 1)k + A_r$ we get $A_{r+1} = (2^r - 1)^2 + A_r + A_r^2/k^2 + (2^{r+1} - 2)A_r/k$.

We now standardize $Y_{r+1}$ and for the standardized random variable $(Y_{r+1} - k)/(\sqrt{var(Y_{r+1})})$ we find its exponential generating function in the limit for large $k$. Let for clarity in the computations that will follow $k + \sigma_r^2/k^2 = var(Y_{r+1}) = (2^{r+1} - 1)k + O(1)$. The limiting e.g.f. of $Y$ is given by Proposition 8 as well. We thus get

$$
\begin{aligned}
\mathcal{E}[e^{t\ \frac{Y_{r+1}-k}{\sqrt{k+\sigma_r^2/k^2}}}] &= e^{-tk/\sqrt{k+\sigma_r^2/k^2}}\ \mathcal{E}[e^{t\ Y_{r+1}/\sqrt{k+\sigma_r^2/k^2}}] \\
&= e^{-tk/\sqrt{k+\sigma_r^2/k^2}}\ \exp\left((\frac{\sigma_r^2}{2\ k^2} - k) + \frac{1}{k^2}(k^3 - \sigma_r^2)e^{t/\sqrt{k+\sigma_r^2/k^2}} + \frac{\sigma_r^2}{2\ k^2}e^{2\ t/\sqrt{k+\sigma_r^2/k^2}}\right) \\
&= e^{-tk/\sqrt{k+\sigma_r^2/k^2}}\ \exp\left(\frac{\sigma_r^2}{2\ k^2} - k\right) \\
&\cdot\ \exp\left(\frac{k^3 - \sigma_r^2}{k^2}\ \left(1 + \frac{t}{1!\ \sqrt{k + \sigma_r^2/k^2}} + \frac{t^2}{2!\ (\sqrt{k + \sigma_r^2/k^2})^2} + \dots\right)\right) \\
&\cdot\ \exp\left(\frac{\sigma_r^2}{2\ k^2}\ \left(1 + \frac{2\ t}{1!\ \sqrt{k + \sigma_r^2/k^2}} + \frac{4\ t^2}{2!\ (\sqrt{k + \sigma_r^2/k^2})^2} + \dots\right)\right) \\
&= e^{t^2/2 + O\left((k+3\ \sigma_r^2/k^2)/3!\ (\sqrt{k+\sigma_r^2/k^2})^3\right)}.
\end{aligned}
$$

We then substitute the value for $\sigma_r^2$ and for large $k$ the $O$ term in the expression above becomes $O(1/\sqrt{2^r\ k})$ thus approaching zero. The standardized variable $Y_r$ has then e.g.f. that approaches in the limit for large $k$ that of the standard normal distribution and the claim of the Theorem follows. Note that at each iteration in order to establish our result we take limits over $k$. Since our claims above hold for constant number of iterations only this eliminates problems caused by having a constant number of nested limits.■

### 3.13.2 Tails of the distribution

Since $Y_r$, the random variable representing the size of a set formed at some iteration $r$, for large enough $k$, $n \to \infty$ and constant $r$ follows a normal distribution $N(k, (2^r - 1)k + O(1))$, we get that the probability that $Y_r$ is zero is given by

$$Pr(0 \leq Y_r \leq 1) = Pr\left(\frac{-k}{\sqrt{(2^r - 1)k + O(1)}} \leq \frac{Y_r - k}{\sqrt{(2^r - 1)k + O(1)}} \leq \frac{1 - k}{\sqrt{(2^r - 1)k + O(1)}}\right).$$

Then

$$\begin{aligned} Prob(0 \leq Y_r \leq 1) &= \frac{1}{\sqrt{2\pi}} \int_{\frac{-k}{\sqrt{(2^r-1)k+O(1)}}}^{\frac{1-k}{\sqrt{(2^r-1)k+O(1)}}} e^{-x^2/2}\, dx \\ &= \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{2^r\ k}} O(e^{-k/2^r}). \end{aligned}$$

## 3.14 Experimental Results

We have run random graph simulations in order to verify our claims related to the distribution and properties of the random variable $Y_r$ in the iterative procedure described by Definition 6. We examined the cases with $k = 25$ and $k = 50$.

The first measurements we got were related to the mean and the variance of $Y_r$, for $r = 1, \ldots, 5$.

These values are presented in the first column of the two tables to follow under the title “Observed Statistics”. The column under the title “Theoretical Values” gives the theoretical values as these can be derived from Proposition 8 (or Theorem 10) as well as Proposition 4.

In the two remaining tables we give various statistics of interest for the first five iterations of our experiments for both values of $k$. Such statistics are the relative frequency of sets of size zero, of sets of size less than $k/2$, greater than $2k$, greater than $3k$, and the fraction of the generated sets whose size lies within $\pm 2\sqrt{k}$, $\pm 3\sqrt{k}$, and $\pm 3\sqrt{k}$ the mean $k$.

In the experiments, for $k = 25$, at iteration one we did not observe the formation of any

| $k = 25$ | | | | |
|---|---|---|---|---|
| | Observed Statistics | | Theoretical Values | |
| *Iteration* | *Sample Mean* | *Sample Variance* | *Mean* | *Variance* |
| 1 | 25.00247 | 24.99586 | 25 | 25 |
| 2 | 25.00758 | 76.08536 | 25 | 76 |
| 3 | 25.01322 | 186.73018 | 25 | 186.2416 |
| 4 | 25.02155 | 453.74548 | 25 | 452.9806938 |
| 5 | 24.97017 | 1214.73326 | 25 | 1259.267802 |

Figure 3-4: Mean and Variance for $k = 25$.

| $k = 50$ | | | | |
|---|---|---|---|---|
| | Observed Statistics | | Theoretical Values | |
| *Iteration* | *Sample Mean* | *Sample Variance* | *Mean* | *Variance* |
| 1 | 50.00167 | 49.99370 | 50 | 50 |
| 2 | 50.00502 | 151.15020 | 50 | 151 |
| 3 | 50.00502 | 361.84793 | 50 | 361.1204 |
| 4 | 50.00698 | 824.89889 | 50 | 824.4039772 |
| 5 | 49.97294 | 1957.54140 | 50 | 1970.664722 |

Figure 3-5: Mean and Variance for $k = 50$.

| Observed Statistics | | | | | |
|---|---|---|---|---|---|
| k=25 | | | | | |
| *Iteration (r)* | 1 | 2 | 3 | 4 | 5 |
| $Y_r = 0$ | <0.000000152588 | 0.000001221 | 0.000251 | 0.0060315 | 0.045813 |
| $Y_r < k/2$ | 0.003155365 | 0.054816 | 0.166768 | 0.310787 | 0.47074 |
| $Y_r > 2k$ | 0.0000018311 | 0.00715271 | 0.0512732 | 0.104262 | 0.1295 |
| $Y_r > 3k$ | $<$ 0.000000152588 | 0.000018311 | 0.00483 | 0.032035 | 0.06519 |
| $k \pm \sqrt{k}$ | 0.729891 | 0.4734015 | 0.3169 | 0.21156 | 0.13639 |
| $k \pm 2\sqrt{k}$ | 0.96519 | 0.781260 | 0.5797 | 0.4092 | 0.2719 |
| $k \pm 3\sqrt{k}$ | 0.997742 | 0.93228 | 0.7864 | 0.6103 | 0.4327 |

Figure 3-6: Collected Statistics for $k = 25$.

| Observed Statistics | | | | | |
|---|---|---|---|---|---|
| k=50 | | | | | |
| *Iteration (r)* | 1 | 2 | 3 | 4 | 5 |
| $Y_r = 0$ | $< 0.000000153$ | $< 0.00000031$ | $< 0.00000062$ | 0.000019532 | 0.0011036 |
| $Y_r < k/2$ | 0.000074 | 0.0111551 | 0.07023987 | 0.1815918 | 0.325677 |
| $Y_r > 2k$ | $< 0.000000153$ | 0.0003675 | 0.0147132 | 0.05891 | 0.10462 |
| $Y_r > 3k$ | $< 0.000000153$ | $<0.00000031$ | 0.00020081 | 0.007785 | 0.036 |
| $k \pm \sqrt{k}$ | 0.71173 | 0.45919 | 0.3088 | 0.20989 | 0.1416 |
| $k \pm 2\sqrt{k}$ | 0.960215 | 0.7665 | 0.5643 | 0.401579 | 0.2763 |
| $k \pm 3\sqrt{k}$ | 0.99747 | 0.9237 | 0.76203 | 0.58141 | 0.41702 |

Figure 3-7: Collected Statistics for $k = 50$.

set of size zero or of size at least $ke = 25e$. Therefore we can not draw conclusions on how good the theoretical derived bounds are compared to the ones derived from the simulations. The latter suggest that $Prob(Y_1 = 0) < 10^{-6}$ and $Prob(Y_1 \geq ek) < 10^{-6}$. For iteration two the theoretically derived bound for $Prob(Y_2 = 0)$, which is $0.3561194 \times 10^{-5}$, is about three times larger than observed in the simulations. No set of size at least $e^2k$ was observed in the simulations.

For $k = 50$, no set of size equal to zero was formed in the first three iterations. In the first two iterations no sets of size at least $e^2k$ were formed. This may confirm the theoretically derived bounds which indicate that the probability of appearance of such sets is very small (smaller than $0.6 \times 10^{-11}$ in any of the cases of interest).

## 3.15 Conclusions

We described in this chapter a graph construction problem so that graphs so constructed have some property $P^k$, for $k$ some parameter of interest. We presented a negative result related to such constructions. We also gave a graph construction that exhibits property $P^k$ for some, but not all, possible cases. It remains an interesting question whether graphs better than the one constructed here can be explicitly constructed.

We also examined in this chapter the properties of a class of random graphs for some iterative procedure. We got some non-trivial upper bounds for the tails of the distribution of the size of various sets of vertices formed at various iterations of this procedure.

We observed a dependence of these bounds on the iteration and the replication factor $k$. The bounds imply that moderate values of $k$ (say, 25-50) are sufficient for various applications and sets formed at the first few iterations of this procedure have rarely sizes greater than $3-5$ times the parameter $k$. These results hold for constant $k$.

In two of the last sections we presented asymptotic results for large values of $k$. We proved a normal approximation for the random variable that represents the size of a set of vertices formed at some constant iteration of the iterative procedure.

Generalizing the proof techniques of section 3.11 and using Bernstein polynomial convergence [17], we have attained more accurate bounds than the ones given here. The bounds are in the form of complicated sums and seem to be of little interest. This is the reason they were not included in this thesis. It would nevertheless be interesting to get for small constant values of $k$ simple upper bounds for the tails of the random variable $Y_r$ for values of $r$ up to say 8-10. Using Bernstein polynomial convergence such expressions have been obtained for iterations three and four by simplifying complicated sums.

# Chapter 4

# Broadcasting in Random Graphs

## 4.1 Introduction

In a complete graph $K_n$ on $n$ vertices it is possible in $\lceil \lg n \rceil$ time steps to broadcast a message from any vertex of the graph to the remaining vertices by doubling at each step the number of vertices holding the message. In this discussion $\lg n$ will denote the base two and $\log n$ the natural logarithm of $n$.

We are interested in the minimum time needed to broadcast a message from a given vertex to all the vertices of $G_{n,p}$ random graphs. We are also interested in the *broadcast number* of such graphs, i.e. the maximum over all vertices of the minimum time needed to broadcast a message from a given vertex.

We shall use the term *near-optimal* introduced in [46] to refer to graphs whose broadcast number is $\lg n + o(\lg n)$. The term *close-to-optimal* will similarly refer to graphs with broadcast number $\lg n + O(1)$, where $O(1)$ is a small constant, and the term *optimal* to graphs with broadcast number equal to $\lceil \lg n \rceil$.

The problem of broadcasting in random graphs was first studied by E. R. Scheinerman and J. C. Wierman in [46], where they proved that it was possible to perform with high probability, that is with probability tending to one as $n$ approaches infinity, what they called *near-optimal* broadcast in $G_{n,p}$ random graphs for $p = \omega_n \log n/n$ with $\omega_n \to \infty$ as $n \to \infty$. It was also proved in that paper that a certain broadcast algorithm could optimally broadcast, with high probability, from any given vertex to all the vertices of $G_{n,p}$ random graphs with edge probability $p = c\log^2 n/n$, for a constant $c > 8/\log 2$, in exactly $\lceil \lg n \rceil$ time steps and thus derived a broadcast number of $\lceil \lg n \rceil$ for almost all such random graphs.

## 4.2 Results

The first set of results to be presented in this chapter is related to near-optimal and optimal broadcast.

In [46] it was conjectured that it is possible to near-optimally broadcast a message in random graphs with edge probability $p = c\log n/n$, for a constant $c > 1$, in $\lg n + o(\lg n)$ time steps with high probability. We prove this conjecture for $c > 18.4$ and for broadcast time $\lg n + d\,\mathrm{lglg}n$, where $d$ is a positive constant less that one.

It was also conjectured in [46] that it is possible to optimally broadcast a message, with high probability, in $\lceil \lg n \rceil$ time steps in random graphs with edge probability $p = \omega_n \log n/n$, where $\omega_n \to \infty$ as $n \to \infty$. We show here that for $p = c\,\mathrm{lglg}n \log n/n$, for any constant $c > 16$, this is feasible thus improving the previous bound of [46] on $p$.

In the second set of results using the techniques presented in the first one and also introducing new ones we show how *close-to-optimal* broadcast in at most $\lceil \lg n \rceil + 1$ steps can be achieved for $G_{n,p}$ random graphs with $p = c\,\mathrm{lglglg}\, n \log n/n$, for any constant $c > 16$. Hence by adding at most one more step in the minimum required broadcast time we manage to further decrease the value of probability $p$.

We also show that the technique used in the proof can be applied to the near-optimal case and give a broadcast number of $\lg n + f\,\mathrm{lglglg}\, n$, $f < 1$, for almost all $G_{n,p}$ random graphs with $p = c\log n/n$, for any $c > 18.4$, thus improving the previously stated result for this class of graphs.

Throughout this chapter we shall assume that $n$ is a power of two. For the near-optimal broadcast case, if this is not the case, we can replace as fit occurrences of $\lg n$ and $\mathrm{lglg}n$ by $\lfloor \lg n \rfloor$ and $\lceil \mathrm{lglg}n \rceil$ respectively. The same holds for the optimal and close-to-optimal broadcast cases although for these cases we examine in detail the case for $n = 2^{2^k}$, $k$ an integer, the subcase $n = 2^k$ follows by replacing each occurrence of lglg$n$ there by $\lceil \mathrm{lglg} n \rceil$. The case of $n$ not being a power of two is discussed separately there.

## 4.3 Broadcast trees on random graphs

We present below a method to find *broadcast trees* in random graphs, i.e. rooted subtrees of the random graphs on at most half of the vertices of the graph that will be used to disseminate quickly a message held by the root of the tree, within a given number of time steps, to as many vertices of this tree and the graph as possible.

A $G_{n,p}$ random graph on $n$ labeled vertices can be constructed by "revealing" graph edges in some order. An edge will be included in the graph independently of the other ones with probability $p$. When we find a broadcast tree rooted at some vertex of this graph we first generate the edges of the tree thus revealing their dependencies in the graph. Such a tree will use at most half of the $n$ vertices of the graph. At the end of this tree finding process we conclude the random graph construction by completing the generation of all the unaccounted edges and the random graph is thus formed.

We need to show that in $G_{n,p}$ random graphs, for values of $p$ as needed for the near-optimal, optimal, and close-to-optimal cases respectively, for each vertex of the random graph we are able to find such a tree with high probability. Each internal vertex of this tree will have a number of children in the tree that belongs to an interval $[\lg n, O(\lg n)]$ (or $[\mathrm{lglg} n, O(\mathrm{lglg} n)]$) as required by the probabilistic considerations of the tree finding process. The upper bound of each of the two possible intervals depends, among other things, on the choice of $p$.

We find such a broadcast tree in a random graph using a technique based on the one that appeared in [1] and also used in [21].

We shall describe this procedure and also present it in algorithmic form. Besides the labeling of the vertices of the random graph implied by the definition of such a graph, we shall also label its vertices using a mapping $l : V \rightarrow \{0, \ldots, L\} \cup \{*\}$, where $L$ in the first set of results is such that $L = \lg n - \mathrm{lglg} n - 5$ and in the second one is such that $L =$

$\lg n - \lceil \mathrm{lglglg}\, n \rceil - 5$. Such a labeling of the vertices of the tree will be referred hereafter as a *broadcast labeling*. We shall often omit the "broadcast" part when we refer to such a labeling. If vertex $v$ of some broadcast tree $T$ is labeled $l(v) \neq *$, this will imply that vertex $v$, in the course of execution of the following broadcast algorithm, will receive from its father the message to be broadcast in the tree at time step $l(v)$ and in the next time step it will start broadcasting it to some of its children.

**Procedure** Broadcast (d)
// Broadcast a message for $d$ time steps.
// The message is held initially by the root of a broadcast tree.
// This procedure is executed asynchronously by each vertex of the tree.
//
**begin**
**for all** vertices $u$ such that $l(u) \neq *, d$ **do**
// Let $u_j$, $1 \leq j \leq d - l(u)$, be the child of $u$
// in the tree, if there is such, with label $l(u) + j$.
//
    **for** $j = 1$ **to** $d - l(u)$ **do**
    At time step $l(u) + j$ vertex $u$, which received the message at
    time step $l(u)$, transmits it to vertex $u_j$;
**end**

When Broadcast($L$) is executed on the trees that will be formed in this chapter, depending on the value of $L$ used in each set of results, we get that in the first set of such results, by the completion of the $L$-th time step $n/(32 \lg n)$ vertices will have received the broadcast message. Because of weaker probabilistic considerations in the second set of results the message will be broadcast by the completion of the $L$-th time step, using the second value for $L$, to at least $n(2-a)^{\mathrm{lglg} n - \mathrm{lglglg}\, n}/(32 \lg n)$ vertices, where $a$ depends on $n$ and $a \to 0$ as $n \to \infty$.

We now describe this tree finding process for a tree rooted at some particular vertex $u$, by revealing the edges of the random graph in such an order that the edges of the desired tree are revealed first.

In the beginning of the tree finding process we first label the root $u$ of the tree to be constructed. Its label will be 0. At this point no generation of the edges of the random graph has occurred and only the root of the tree to be found has been labeled.

We then "expand" $u$ that is, reveal the edges connecting $u$ with other vertices of the random graph. We would like to reveal edges among at most half of the vertices of the random graph during the tree finding process. Let set $S$ be a set of vertices of the graph of size $|S| = s$. Its exact size will depend on various parameters, including $p$, and will be at most $n/4$. Set $S$ will hold the prospective children of $u$ and in general of a vertex about to be "expanded". For $u$ and in general for any vertex about to be "expanded" we first determine the number $U$ of its children in the tree according to the following binomial distribution:

$$Prob(U = k) = \binom{s}{k}\, p^k\, (1-p)^{s-k}.$$

That way we decide *some* but not necessarily all the vertices adjacent to a given vertex in $G_{n,p}$. As soon as we decide the number of children of the vertex in question ($u$ here) and let this number be $k$, we choose $k$ among the $s$ vertices of $S$ uniformly at random. Each subset of $S$ of size $k$ is chosen with the same probability among the $\binom{s}{k}$ possible ones. These vertices will become the children of the expanded vertex (that is, $u$) in the tree being formed. We decide arbitrarily an ordering of the vertices say, according to the labeling of the vertices in the random graph. This ordering will be used for broadcast labeling purposes. We then remove these vertices from $S$. We add vertices to $S$ so that it recaptures its original size $s$ by choosing $k$ arbitrary vertices, among the remaining vertices of the graph or alternatively, from a subset of those vertices. In this treatment we shall add vertices to $S$ so that a total of at most $n/2$ of the $n$ vertices of the random graph will ever participate in this tree finding process either by including them in the tree or in set $S$. After we have revealed the children of $u$ in the tree and let them be $u_1, \ldots, u_k$, we label them in such a way that vertex $u_i$, $1 \leq i \leq L - l(u)$, is labeled $l(u) + i$ and the remaining vertices are labeled $*$. We repeat this process by similarly expanding the children of $u$ that have labels belonging to the set $\{1, \ldots, L-1\}$. The order of expansion of these vertices is given in procedure Expand that follows this discussion.

If we assume that in the beginning of the tree finding process $|S| = s \leq n/4$, then we need to show that this process will consider with high probability at most $n/2 - s$ vertices altogether so that we will be able at any step of this process to add vertices to $S$ so that it is always of size $s$ and use overall in this process at most $n/2$ vertices. In case more than $n/2$ vertices are considered, we declare FAILURE of the process and the broadcast.

This tree finding process can be described in terms of procedure Expand given below.

**Procedure** Expand(T,P,s,d,D,L);
// Procedure Expand expands vertices, by revealing some of the edges of the
// random graph, to form a tree that is a subgraph of the random graph.
// $T$ is a set variable that contains the set of edges revealed so far.
// $P$ is a set variable containing a set of vertices. All vertices in $P$
// will be considered for expansion.
// $s$ is the cardinality of set $S$ and is fixed in any single call of Expand.
// The distance of any leaf of the tree at the completion of Expand, from any
// vertex of $P$ will be at most $d$.
// Only vertices that will be assigned labels less than $D$,
// during that call of Expand, will be expanded.
// $L$ is the maximum value of a label to a vertex.
// $p$ is the edge probability of the class of random graphs considered.
**begin**
0. **for every vertex** $u$ **in** $P$ **do**
    **begin**
//
// One must interpret the set notation used for $P$ in terms of a priority queue.
// The smaller the label $l(v)$ of a vertex the higher its priority.

// When we remove an element from $P$,we remove the
// one with the highest priority (and an arbitrary one in case of a tie).
// That is the meaning of $P = P - \{u\}$ in step 1 below.
// In the same manner, $P = P \cup \{w\}$ inserts element $w$ into
// the priority queue.
//

1. $P = P - \{u\}$;
   **if** $l(u) < D$ **and** $l(u) \neq *$
   **begin**
2. Determine the value of the random variable $U$ that gives the number of prospective children of $u$ in the tree being constructed. Hence choose a $k$ such that the random variable $U$ equals $k$ with probability
   $Prob(U = k) = \binom{s}{k}\, p^k\, (1-p)^{s-k}$.
3. Choose uniformly at random $k$ among the $s$ vertices of set $S$.
4. Let $u_1, \ldots, u_k$ be the vertices of $S$ chosen in the previous step. Fill $S$ with $k$ vertices, drawn arbitrarily from POOL, so that it becomes again of size $s$.

$4'$. Determine whether we need to declare FAILURE of Expand and the broadcast, according to the comments in the proof of any one of Propositions 9, 10, 13 that follow.

5. **for** $i = 1$ **to** $k$ **do**
   **if** $l(u) + i \leq L$
6. $l(u_i) = l(u) + i$;
   **else**
7. $l(u_i) = *$;

// Only vertices that are scheduled to be expanded are added to $P$.
// The index $d - 1 - (l(u) - D - d)$ below is dependent on

// the values of $D$, $d$ used in this call of Expand.

8. $P = P \cup \{u_1, \ldots, u_{d-1-(l(u)-D-d)}\}$;

9. Store the edges between $u$ and $u_1, \ldots, u_k$ in $T$.

**end**

**end**

Return(T);

**end**

We present below three different procedures which share the name Phase. The first two, Phase 1 and Phase 2, are used in the first set of results related to near-optimal and optimal broadcast. Their only difference is that they apply to different classes of random graphs and therefore have different values for parameter $s$. Procedure Phase3 is used in the second set of results related to close-to-optimal broadcast, and to an improved bound for near-optimal broadcast, which would require some changes on the values of $s$ used in Phase3. There are null (no operation) steps in procedures Phase1 and Phase2. They exist only for Phase1, Phase2 and Phase3 to have the same number of steps.

Given Expand, each Phase Procedure forms a random $G_{n,p}$ graph. The first three (respectively six) steps of Phase1 or Phase2 (respectively Phase3) generate edges randomly and thereby construct the claimed broadcast tree. Step 6 completes the generation of all the remaining edges of the random graph.

**Procedure** Phase1 (u)

// Vertex $u$ is going to be the root of the broadcast tree to be constructed.

// The size of the graph $n$ is of the form $n = 2^{2^k}$, $k$ a positive integer.

**begin**

0. Let POOL contain an arbitrary set of $n/2$ vertices of the graph.

Set $S$ in step 1 will be formed from vertices of this set;

1. $P = \{u\}; l(u) = 0; s = 23n/64; d = \lg n - \mathrm{lglg} n - 5; D = d;$

$L = \lg n - \mathrm{lglg} n - 5; T = \emptyset;$

2. $Expand(T, P, s, d, D, L);$

3.

4.

5.

// All vertices with labels $< L$ have been expanded.

6. Generate the remaining edges of the random graph.

// The random graph has been constructed and $T$ shows a subtree of it rooted at $u$.

**end**

**Procedure** Phase2 (u)

// Vertex $u$ is going to be the root of the broadcast tree to be constructed.

// The size of the graph $n$ is of the form $n = 2^{2^k}$, $k$ a positive integer.

**begin**

0. Let POOL contain an arbitrary set of $n/2$ vertices of the graph.

   Set $S$ in step 1 will be formed from vertices of this set;

1. $P = \{u\}; l(u) = 0; s = (6.75n)/(16\,\mathrm{lglg} n); d = \lg n - \mathrm{lglg} n - 5; D = d;$

   $L = \lg n - \mathrm{lglg} n - 5; T = \emptyset;$

2. $Expand(T, P, s, d, D, L);$

3.

4.

5.

// All vertices with labels $< L$ have been expanded.

6. Generate the remaining edges of the random graph.

// The random graph has been constructed and $T$ shows a subtree of it rooted at $u$.

**end**

Procedure Phase3 is given below.

**Procedure** Phase3 (u)

// Vertex $u$ is going to be the root of the broadcast tree to be constructed.

// The size of the graph $n$ is of the form $n = 2^{2^k}$, $k$ a positive integer.

**begin**

0. Let POOL contain an arbitrary set of $n/2$ vertices of the graph.
   Set $S$ in steps 1 and 4 will be formed from vertices of this set;
1. $P = \{u\}; l(u) = 0; s = (6.7n)/(16 \,\mathrm{lglglg}\, n); d = \lg n - \mathrm{lglg} n - 5; D = d;$
   $L = \lg n - \lceil \mathrm{lglglg}\, n \rceil - 5; T = \emptyset;$
2. $Expand(T, P, s, d, D, L);$
3. Let $W$ be the set of vertices $v$ of $T$ such that $l(v) \geq \lg n - \mathrm{lglg} n - 5$.
   These vertices are leaves in tree $T$.
4. $P = W; s = (5n \,\mathrm{loglog} n)/(16 \,\mathrm{lglglg}\, n \log n); d = \mathrm{lglg} n - \lceil \mathrm{lglglg}\, n \rceil;$
   $L = \lg n - \lceil \mathrm{lglglg}\, n \rceil - 5; D = L;$
5. $Expand(T, P, s, d, D, L);$

// All vertices with labels $< L$ have been expanded.

6. Generate the remaining edges of the random graph.

// The random graph has been constructed and $T$ shows a subtree of it rooted at $u$.

**end**

The broadcast tree represented by the variable $T$ in any of the three procedures called Phase will be referred hereafter simply as $T$. The contents of $T$ after the completion of steps 2 and 5 in Phase will be of interest.

We note that all the edges revealed in steps 1-5 of Phase are between pairs of the $n/2$ vertices in POOL, and do not involve the remaining $n/2$ vertices at all.

## 4.4 Properties of broadcast trees

We present below some properties of broadcast tree $T$. We first present some preliminary Propositions that are specific to each result related to near-optimal, optimal and close-to-optimal broadcast. We then show some properties of $T$ implied by any of these Propositions.

### 4.4.1 Preliminary results for Near-optimal and Optimal broadcast

We claim the following Proposition which holds for $G_{n,p}$ random graphs for both cases of $p$ needed that is, $p = p_1 = c \log n/n$, for $c > 18.4$, for the near-optimal case and $p = p_2 = c\,\mathrm{lglg}n \log n/n$ for $c > 16$ for the optimal broadcast case.

We first show the following Proposition.

**Proposition 9** *In the class $G_{n,p_1}$ (respectively $G_{n,p_2}$) of random graphs when Phase1 (respectively Phase2) is executed, for some positive constant $\delta$ and $n > 10$, with probability at least $1 - 1/n^{2+\delta}$, any one fixed vertex of $T$ obtains at least $\lg n$ and at most $8.97 \lg n$ (respectively $9.3 \lg n$) children.*

**Proof:** Let for the $G_{n,p_1}$ case $p = p_1 = 6.6\frac{64}{23}\frac{\log n}{n} < 18.4\ \frac{\log n}{n}$. Let the size of set $S$ be taken to be $|S| = s = 23n/64$. Since every vertex in the random graph and in $T$ is connected to a particular vertex with probability $p$, the mean of the binomial distribution that gives the number of prospective children, indicated in line 2 of Expand, of a vertex in $T$ is $p|S| = 6.6 \log n = 6.6 \log 2 \lg n$. We examine for a particular vertex the probability that this vertex gets less than $\lg n$ or more than $8.97 \lg n$ children.

Let $U$ be the random variable that gives the number of children of a given vertex in the tree finding process. The Angluin-Valiant bounds of Fact 7 give the following inequalities.

$$Prob(U \geq 8.97 \lg n) \leq n^{-2.03},$$

and

$$Prob(U \leq \lg n) \leq n^{-2.0149}.$$

These bounds prove our claim for the near-optimal broadcast case.

For $G_{n,p_2}$ graphs, where $p = p_2 = 16\,\mathrm{lglg}n \log n/n$, we only need to change the size of $S$. Let $S$ be of size $|S| = (6.75n)/(16\,\mathrm{lglg}n) < n/4$ (for $n > 10$). Then $p|S| = 6.75 \log n$, and we get the following bounds.

$$Prob(U \geq 9.3 \lg n) \leq n^{-2.195},$$

and

$$Prob(U \leq \lg n) \leq n^{-2.08648}.$$

In case step 2 of Expand gives a value for $U$ that is outside the desired interval, we declare failure of Expand, of Phase1 or Phase2, and the broadcast. ■

### 4.4.2 Preliminary results for Close-to-optimal broadcast

The following Proposition applies to Procedure Phase3 with regard to $G_{n,p}$ random graphs with $p = c\,\mathrm{lglglg}\, n \log n/n$, $c > 16$.

**Proposition 10** *In the class $G_{n,p}$ of random graphs, $p = c\,\mathrm{lglglg}\, n \log n/n$, where $c > 16$, when Phase3 is executed, for $n > 600$ and positive $\delta$ (respectively $\zeta$), with probability at least $1 - 1/n^{2+\delta}$ (respectively $1 - 1/\log^{1+\zeta} n$), any one fixed internal vertex of $T$ between step 1 (respectively step 3) and step 3 (respectively step 6) of Phase3 obtains at least $\lg n$ (respectively $\mathrm{lglg}n$) and at most $9.13 \lg n$ (respectively $6.25566\,\mathrm{lglg}n$) children.*

**Proof:** Let $p = 16\,\mathrm{lglglg}\, n \log n/n$. During the first call to Expand in Phase3 the size of set $S$ is $|S| = s = 6.7n/(16\,\mathrm{lglglg}\, n) < n/4$ (for $n > 600$). Since every vertex in the random graph (and thus in the tree) is connected to a particular vertex with probability $p$, the mean of the binomial distribution that gives, in step 2 of Expand, the number of prospective children of a vertex in the broadcast tree is $p|S| = 6.7 \log n = 6.7 \cdot \log 2 \cdot \lg n$. We examine for a particular vertex the probability that this vertex gets less than $\lg n$ or more than $9.13 \lg n$ children.

Let $U$ be the random variable that gives the number of children of some vertex in the tree finding process. The Angluin-Valiant bounds of Fact 7 give the following inequalities.

$$Prob(U \geq 9.13 \lg n) \leq n^{-2.079},$$

and

$$Prob(U \leq \lg n) \leq n^{-2.062},$$

that is with probability at least $1 - n^{-2.079} - n^{-2.062} \geq 1 - n^{-2-\delta}$, for an appropriate $\delta > 0$, we get that $U$ is at least $\lg n$ and at most $9.13 \lg n$. Thus our claims for the internal vertices of $T$ just after the completion of step 2 are proved.

For the vertices included in $T$ after step 2 and by the time step 5 is completed we only need to apply the previous bounds to the new size of $S$. Since we choose to have $|S| = (5n\,\mathrm{loglog}n)/(16\,\mathrm{lglglg}\,n \log n) < n/4$ (for $n > 600$), $p|S| = 5\,\mathrm{loglog}n$, we similarly get the following bounds as before.

$$Prob(U \geq 6.25566\,\mathrm{lglg}n) \leq \log^{-1.08} n,$$

and

$$Prob(U \leq \mathrm{lglg}n) \leq \log^{-1.0708} n,$$

that is with probability at least $1 - \log^{-1.08} n - \log^{-1.0708} n \geq 1 - \log^{-1-\zeta} n$, for an appropriate $\zeta > 0$, we get that $U$ is at least $\mathrm{lglg}n$ and at most $6.25566\,\mathrm{lglg}n$.

In case step 2 of any execution of Expand gives a value for $U$ that is outside the desired interval, we declare failure of Expand in step 2 or 4 of Phase3, of Phase3 and the Broadcast.

■

### 4.4.3 Two properties of broadcast tree $T$

The following two claims will be needed.

**Claim 5** *In any successful call of Expand from Phase1, Phase2 or Phase3, the number of vertices $v$ after the completion of step 2 of Phase1, Phase2 and Phase3 such that $l(v) \leq i$, $i \leq \lg n - \mathrm{lglg} n - 5$, is equal to $2^i$.*

**Proof:** An inductive argument on $i$ suffices for the proof. There exists one vertex labeled 0. For $i \leq \lg n - \mathrm{lglg} n - 5$, the number of vertices labeled $i$ is equal to the number labeled at most $i-1$ since by Proposition 10 or Proposition 9, each internal vertex of $T$ has at least $\lg n > \lg n - \mathrm{lglg} n - 5$ children (unless we declare failure of Expand) and thus each vertex labeled at most $i-1$ gets exactly one child labeled $i$ in step 6 of Expand. The claim then follows. ■

**Claim 6** *In any successful call of Expand from Phase1, Phase2 or Phase3, the leaves of $T$ labeled other than $*$ just after step 2 of Phase1, Phase2 and Phase3 are the vertices labeled at least $\lg n - \mathrm{lglg} n - 5$.*

**Proof:** The vertices labeled $\geq \lg n - \mathrm{lglg} n - 5$ are never included in set $P$ because of the $if$ statement between steps 1 and 2 in Expand and are thus never "expanded", i.e. they never acquire children in $T$ during the execution of step 2 of Phase. ■

In addition to the previous claim, the expanded vertices, which are the ones labeled at most $\lg n - \mathrm{lglg} n - 6$ are, by Claim 5, equal in number to the number of leaves of $T$ labeled $\lg n - \mathrm{lglg} n - 5$.

## 4.5 Some results on Complete Matchings in Bipartite Graphs

Below we state some results on matchings in bipartite graphs, which will be used later in this work. Some of them can also be found in [8].

**Definition 11** *Let $G = (X \cup Y, E)$, $|Y| \geq |X|$, be a bipartite graph. The deficiency $\delta(G)$ of $G$ is defined as:*

$$\delta(G) = \max_{A \subseteq X}(|A| - |R(A)|),$$

*where $R(A)$ is the range of set $A$ in $Y$, i.e. the vertices of $Y$ that are adjacent to at least one vertex of $A$.*

**Fact 9** *Let $G = (X \cup Y, E)$, $|Y| \geq |X|$, be a bipartite graph. The number of vertices of $X$ that are left unmatched in a maximum matching of $G$ is equal to $\delta(G)$.*

The model of random bipartite graphs with vertex bipartition $V = X \cup Y$ is the following: For each vertex of $X$ and each vertex of $Y$, there is an undirected edge between them with probability $p$.

We give the proof here of a result presented in [10] modified to fit our case.

**Theorem 14** *Let $G = (X \cup Y, E)$ be a random bipartite graph, where $|X| = |Y| = r$, $(n/8 - n/(16 \ \lg n)) \leq r \leq n$, and the edges between $X$ and $Y$ appear independently with probability $p$, where $p = c \log n/n$, $c > 16$, $c = O(\log n)$. Then $G$ has a complete matching with probability at least $1 - 1/n^{1.5}$, for large $n$.*

The restriction $c = O(\log n)$ was added to facilitate the discussion related to the various cases of $p$ used in this work. The proof requires the following lemma from [10].

**Lemma 3** *Let $G = (X \cup Y, E)$ be a bipartite graph and let $|X| = |Y| = n$. Suppose $G$ has no isolated vertices and it does not have a complete matching. Then there is a set $A \subset X$ or $A \subset Y$ such that:*

1. *$R(A) = \{y : \ \{x, y\} \ \in E$ for some $\ x \in A\}$ has $|A| - 1$ elements.*

2. *the subgraph of $G$ spanned by $A \cup R(A)$ is connected and*

3. *$2 \leq |A| \leq (n + 1)/2$.*

The proof of Theorem 14 follows.

**Proof of Theorem 14:** We would like to show that for a random bipartite graph with the stated conditions the probability that the graph has no perfect matching is at most $1/n^{1.5}$. Let $F_a$, as in [10], denote the event that there exists a set $A$ of size $a$ satisfying the three conditions of Lemma 3. We need to show that:

$$Prob(\cup_{a=2}^{n_1} F_a) \leq 1/n^{1.5},$$

where $n_1 = \lfloor (r+1)/2 \rfloor$. Let $A_1 \subset X$, $A_2 \subset Y$ and $|A_1| = |A_2| + 1 = a$. We have $2\binom{r}{a}$ choices for a set $A$ of the Lemma (a subset of either $X$ or $Y$) and $\binom{r}{a-1}$ choices for $R(A)$. The probability that the subgraph spanned by $A_1 \cup A_2$ has at least $2a-2$ edges and no vertex of $A_1$ is joined to a vertex of $Y - A_2$ can be proven to be at most $\binom{a(a-1)}{2a-2} p^{2a-2}(1-p)^{a(r-a+1)}$ (see also [10]). If we combine these terms we get the following inequalities.

$$\begin{aligned}
Prob(\cup_{a=2}^{n_1} F_a) &\leq \sum_{a=2}^{n_1} Prob(F_a) \\
&\leq 2 \sum_{a=2}^{n_1} \binom{r}{a}\binom{r}{a-1}\binom{a(a-1)}{2a-2} p^{2a-2}(1-p)^{a(r-a+1)} \\
&\leq 2 \sum_{a=2}^{n_1} \frac{r^a\ e^a}{a^a}\ \frac{r^{a-1}\ e^{a-1}}{(a-1)^{a-1}}\ \frac{a^{2a-2}\ e^{2a-2}}{2^{2a-2}} p^{2a-2} e^{-pa(r-a+1)} \\
&\leq 2 \sum_{a=2}^{n_1} \left( \frac{r^2\ e^4\ a\ p^2\ e^{-p(r-a+1)}}{4(a-1)} \right)^{a-1} \frac{re}{a}\ e^{-p(r-a+1)} \\
&\leq 2 \sum_{a=3}^{n_1} \left( \frac{r^2\ e^4\ a\ p^2\ e^{-p(r-a+1)}}{4(a-1)} \right)^{a-1} \frac{re}{a}\ e^{-p(r-a+1)} \\
&+ \frac{r^3\ e^5\ p^2\ e^{-2p(r-1)}}{2}.
\end{aligned}$$

We examine first the term $r^2\ e^4\ a\ p^2\ e^{-p(r-a+1)}/(4(a-1))$. Since $a/(a-1) \leq 2$, $e^4\ a/(4(a-1)) \leq e^4/2$. Since $r \leq n$ we get $r^2\ p^2 \leq c^2 \log^2 n$. For $a \leq n_1$ we have $e^{-p(r-a+1)} \leq e^{-p(r-n_1+1)}$.

Then $r^2\ e^4\ a\ p^2\ e^{-p(r-a+1)}/(4(a-1)) \leq (e^4/2)\ c^2 \log^2 n\ e^{-p\,(r+1)/2}$, for $r \geq n/8 - n/(16\ \lg n)$, is less than $1/2$ provided that $c > 16$.

If we combine the various inequalities we get the following.

$$\begin{aligned}
Prob(\cup_{a=2}^{n_1} F_a) &\leq 2 \sum_{a=3}^{n_1} \left( \frac{r^2\ e^4\ a\ p^2\ e^{-p(r-a+1)}}{4(a-1)} \right)^{a-1} \frac{re}{a}\ e^{-p(r-a+1)} \\
&+ \frac{r^3\ e^5\ p^2\ e^{-2p(r-1)}}{2}
\end{aligned}$$

$$
\begin{aligned}
&\leq \; 2 \sum_{a=3}^{n_1} \left( \frac{c^2 \; e^4 \; \log^2 n \; e^{-p(r-n_1+1)}}{2} \right)^{a-1} \frac{ne}{3} \; e^{-p(r-n_1+1)} \\
&+ \; \frac{r^3 \; e^5 \; p^2 \; e^{-2p(r-1)}}{2} \\
&\leq \; 4 \left( \frac{e^4 \; c^2 \; \log^2 n \; e^{-p(r+1)/2}}{2} \right)^2 \frac{ne}{3} \; e^{-p(r+1)/2} \\
&+ \; \frac{n \; e^5 \; c^2 \; \log^2 n \; e^{-2p(r-1)}}{2}.
\end{aligned}
$$

For $c > 16$, after some calculations, we find that

$$Prob(\cup_{a=2}^{n_1} F_a) \;\; < \;\; 1/n^{1.5}.$$

The $1/n^{1.5}$ upper bound also absorbs the probability that there exist some isolated vertices which is at most $2n(1-p)^n \leq 2/n^{15}$. ∎

For the classes of random graphs considered in this chapter we have that the $c$ of Theorem 14 is such that $c > 16$ and $c = O(\log n)$.

We now give the following variation of Theorem 14 when sets $X$ and $Y$ are not of equal size.

**Theorem 15** *Let $G = (X \cup Y, E)$ be a random bipartite graph and let $|X| = r \geq 4, |Y| = x + r \leq n$, $x \geq \lfloor \frac{n}{8\,(\mathrm{lglg}n+1)} \rfloor$ and the edges between $X$ and $Y$ appear independently with probability $p$. Let $p = c \log n/n$ be such that $px > 1.9 \;\log n$ and $c = O(\log n)$. Then $G$ has a complete matching that saturates $X$ with probability at least $1 - \max\{1/n^2, 1/n^{p(x+r)/\log n - 1}\}$, for large $n$.*

The proof requires the following variation of Lemma 3.

**Lemma 4** *Let $G = (X \cup Y, E)$ be a bipartite graph and let $|X| = r, |Y| = r + x$. Suppose $G$ has no isolated vertices belonging to set $X$ and it does not have a complete matching that saturates $X$. Then there is a set $A \subset X$ such that:*

1. *$R(A) = \{b : \; \{a, b\} \; \in E$ for some $\; a \in A\}$ has $|A| - 1$ elements.*

*2. the subgraph of $G$ spanned by $A \cup R(A)$ is connected and*

*3.* $2 \leq |A| \leq r$.

The proof of Theorem 15 follows the steps of that of Theorem 14 . We shall distinguish various cases of the index $a$ introduced in the proof of Theorem 14.

**Proof of Theorem 15:** We would like to show that for a random bipartite graph with the stated conditions, the probability that the graph does not have a complete matching that saturates $X$ is at most $1/n^2$. Let $F_a$, denote the event that there exists a set $A$ of size $a$ satisfying the three conditions of Lemma 4. We need only show that for large $n$:

$$Prob(\cup_{a=2}^{r} F_a) \leq 1/n^2.$$

Let $A_1 \subset X$, $A_2 \subset Y$ and $|A_1| = |A_2| + 1 = a$. We have $2\binom{r}{a}$ choices for a set $A$ of the Lemma (a subset of either $X$ or $Y$), and $\binom{r+x}{a-1}$ choices for $R(A)$. The probability that the subgraph spanned by $A_1 \cup A_2$ has at least $2a - 2$ edges and no vertex of $A_1$ is joined to a vertex of $Y - A_2$ can be proven to be at most $\binom{a(a-1)}{2a-2} p^{2a-2}(1-p)^{a(r+x-a+1)}$. If we combine these terms we get the following inequalities.

$$\begin{aligned} Prob(\cup_{a=2}^{r} F_a) &\leq \sum_{a=2}^{r} Prob(F_a) \\ &\leq \sum_{a=2}^{r} \binom{r}{a}\binom{r+x}{a-1}\binom{a(a-1)}{2a-2} p^{2a-2}(1-p)^{a(r+x-a+1)}. \end{aligned}$$

We distinguish four cases depending on the range of $a$. These are:

- Case 1: $2 \leq a \leq \lfloor r/2 \rfloor$,
- Case 2: $\lfloor r/2 \rfloor + 1 \leq a \leq \lfloor (r+x)/2 \rfloor$. We distinguish this case for $x < r$.
- Case 3: $\lfloor (r+x)/2 \rfloor + 1 \leq a \leq r$. We distinguish this case for $x < r$.
- Case 4: If $x \geq r$ then cases 2 and 3 combine into a single one $\lfloor r/2 \rfloor + 1 \leq a \leq r$. The analysis of this case will be identical to that of case 2.

**Case 1:** We first examine the contribution to the sum of the first case. Let $R = \lfloor r/2 \rfloor$ (not to be confused with $R(X)$). We then get, similarly to the proof of Theorem 14:

$$\begin{aligned}\sum_{a=2}^{R} \tbinom{r}{a}\tbinom{r+x}{a-1}\tbinom{a(a-1)}{2a-2} p^{2a-2}(1-p)^{a(r+x-a+1)} &\leq \sum_{a=3}^{R}\left(\frac{r(r+x)\ e^4\ a\ p^2\ e^{-p(r+x-a+1)}}{4(a-1)}\right)^{a-1} \\ &\cdot \frac{re}{a}\ e^{-p(r+x-a+1)} \\ &+ \frac{r^2\ (r+x)\ e^5\ p^2\ e^{-2p(r+x-1)}}{4}.\end{aligned}$$

Considerations similar to the ones of the corresponding part of the proof of Theorem 14 give that $r(r+x)\ e^4\ a\ p^2\ e^{-p(r+x-a+1)}/(4(a-1)) \leq O(\log^4 n)\ e^{-px}$. This is at most $1/n^{1.5}$ (asymptotically) since $px > 1.9\ \log n$.

If we continue the analysis we get the following inequalities (for large $n$).

$$\begin{aligned}\sum_{a=2}^{R} \tbinom{r}{a}\tbinom{r+x}{a-1}\tbinom{a(a-1)}{2a-2} p^{2a-2}(1-p)^{a(r+x-a+1)} &\leq \sum_{a=3}^{R}\left(\frac{1}{n^{1.5}}\right)^{a-1} \frac{re}{a}\ e^{-p(r+x-a+1)} \\ &+ \frac{r^2\ (r+x)\ e^5\ p^2\ e^{-2p(r+x-1)}}{4} \\ &\leq 1/n^{2.5}.\end{aligned}$$

**Case 2:** Let $W = \lfloor (r+x)/2 \rfloor$.

$$\begin{aligned}\sum_{a=R+1}^{W} \tbinom{r}{a}\tbinom{r+x}{a-1}\tbinom{a(a-1)}{2a-2} p^{2a-2}(1-p)^{a(r+x-a+1)} &\leq \sum_{a=R+1}^{W} \frac{r^{r-a}\ e^{r-a}}{r-a^{r-a}}\ \frac{(r+x)^{a-1}\ e^{a-1}}{(a-1)^{a-1}} \\ &\cdot \frac{a^{2a-2}\ e^{2a-2}}{2^{2a-2}} p^{2a-2}(1-p)^{a(r+x-a+1)} \\ &\leq \sum_{a=R+1}^{W}\left(\frac{(r+x)\ e^3\ a^2\ p^2\ e^{-p(r+x-a+1)}}{4(a-1)}\right)^{a-1} \\ &\cdot \left(\frac{re}{r-a}\right)^{r-a} e^{-p(r+x-a+1)} \\ &\leq \sum_{a=R+1}^{W}\left(\frac{(r+x)\ e^3\ a^2\ p^2\ e^{-p(r+x-a+1)}}{4(a-1)}\right)^{a-1} \\ &\cdot \left(\frac{re}{r-a}\right)^{a} e^{-p(r+x-a+1)}.\end{aligned}$$

since, for the last inequality, $re/(r-a) > 1$ and $r - a \leq a$. We then get the following inequality.

$$\begin{aligned}\sum_{a=R+1}^{W} \binom{r}{a}\binom{r+x}{a-1}\binom{a(a-1)}{2a-2}p^{2a-2}(1-p)^{a(r+x-a+1)} &\leq \sum_{a=R+1}^{W}\left(\frac{r(r+x)\ e^4\ a^2\ p^2\ e^{-p(r+x-a+1)}}{4(r-a)(a-1)}\right)^{a-1}\\ &\cdot \quad \frac{re}{r-a}\ e^{-p(r+x-a+1)}.\end{aligned}$$

The proof technique used in case 1 shows that the contribution of this term is at most $1/n^{2.5}$ (for large $n$).

**Case 3:** The analysis of case 3 follows that of case 2.

$$\begin{aligned}\sum_{a=W+1}^{r} \binom{r}{a}\binom{r+x}{a-1}\binom{a(a-1)}{2a-2}p^{2a-2}(1-p)^{a(r+x-a+1)} &\leq \sum_{a=W+1}^{r-1}\frac{r^{r-a}\ e^{r-a}}{r-a^{r-a}}\ \frac{(r+x)^{r+x-a+1}\ e^{r+x-a+1}}{(r+x-a+1)^{r+x-a+1}}\\ &\quad \frac{a^{2a-2}\ e^{2a-2}}{2^{2a-2}}p^{2a-2}(1-p)^{a(r+x-a+1)}\\ &+ \quad \frac{(r+x)^{x+1}\ e^{x+1}}{(x+1)^{x+1}}\ \frac{a^{2a-2}\ e^{2a-2}}{2^{2a-2}}p^{2a-2}e^{-pa(x+1)}\\ &\leq \sum_{a=W+1}^{r-1}\left(\frac{r(r+x)\ e^2\ \ e^{-pa}}{(r-a)(r+x-a+1)}\right)^{r-a}\\ &\quad \frac{(r+x)^{x+1}\ e^{x+1}}{(r+x-a+1)^{x+1}}\left(\frac{a^2\ p^2\ e^2}{4}\right)^{a-1}\ e^{-pa(x+1)}\\ &+ \quad \frac{(r+x)^{x+1}\ e^{x+1}}{(x+1)^{x+1}}\ \frac{a^{2r-2}\ e^{2r-2}}{2^{2r-2}}p^{2r-2}e^{-pa(x+1)}.\end{aligned}$$

We have that $r(r+x)\ e^2\ e^{-pa}/((r-a)(r+x-a+1)) \leq (n^2\ e^2\ e^{-pa})/x$ since $r+x \leq n$, $a \leq r$. Because $a \geq W+1 \geq (x+r)/2 - 1 + 1 \geq x$ (since $x \leq r$ for case 3) we get that $e^{-pa} \leq e^{-px}$. Since $x \geq \lfloor n/(8(\mathrm{lglg}n+3))\rfloor$ if we combine these inequalities we get that $(n^2\ e^2\ \ e^{-pa})/x = O(1/n^{0.8})$ since $p \cdot x \geq 1.9\ \log n$.

We also get $x \leq (x+r)/2 \leq W+1 \leq a$ and thus $(r+x)^{x+1} \leq (r+x)^{a+1}$. Then

$$\frac{(r+x)^{x+1}}{(r+x-a+1)^{x+1}}\ e^{x+1}\left(\frac{a^2\ p^2\ e^2}{4}\right)^{a-1}\ e^{-pa(x+1)} \quad \leq \quad \frac{(r+x)^2\ e^2}{(r+x-a+1)^{x+1}}$$

$$
\begin{aligned}
\cdot & \left(\frac{(r+x)\ a^2\ p^2\ e^3\ e^{-p(x+1)}}{4}\right)^{a-1} e^{-p(x+1)} \\
\leq & \frac{(r+x)^2\ e^2}{(x+1)^{x+1}} \left(\frac{n^3\ p^2\ e^3\ e^{-p(x+1)}}{4}\right)^{a-1} \\
\cdot & e^{-p(x+1)} \\
\leq & \frac{n^2\ e^2}{(x+1)^{(x+1)}} \left(\frac{c^2\ \log^2 n\ e^3}{4\ n^{0.9}}\right)^{a-1} \frac{1}{n^{1.9}}.
\end{aligned}
$$

Since $a \geq W+1$ and $x \geq \lfloor n/(8(\text{lglg}n+3))\rfloor$, the last inequality is at most $1/n^3$ thus deriving, if we combine the previous inequalities, a contribution of at most $1/n^{2.5}$ (for large $n$) for case 3 [Note: The upper bounds we claim serve only the purpose of showing at the end of the proof an upper bound of $1/n^2$ for the failure probability. A more careful analysis would probably yield stronger bounds.]

**Case 4:** For case 4 we have that $r \leq x$. Considerations similar to those of case 2 lead to the following set of inequalities.

$$
\begin{aligned}
\sum_{a=R+1}^{r} \tbinom{r}{a}\tbinom{r+x}{a-1}\tbinom{a(a-1)}{2a-2}p^{2a-2}(1-p)^{a(r+x-a+1)} \leq & \sum_{a=R+1}^{r-1} \left(\frac{(r+x)\ e^3\ a^2\ p^2\ e^{-p(r+x-a+1)}}{4(a-1)}\right)^{a-1} \\
\cdot & \left(\frac{re}{r-a}\right)^{a-1} e^{-p(r+x-a+1)} \\
+ & \left(\frac{(r+x)\ e^3\ r^2\ p^2\ e^{-p(x+1)}}{4(r-1)}\right)^{r-1} e^{-p(x+1)} \\
\leq & \sum_{a=R+1}^{r-1} \left(\frac{(r+x)\ e^3\ a\ p^2\ e^{-p(r+x-a+1)}}{4(a-1)}\right)^{a-1} \\
\cdot & \left(\frac{re}{r-a}\right)^{a-1} e^{-p(r+x-a+1)} \\
+ & \left(\frac{(r+x)\ e^3\ r^2\ p^2\ e^{-p(x+1)}}{4(r-1)}\right)^{r-1} e^{-p(x+1)}.
\end{aligned}
$$

The analysis of this case follows the other ones and gives an $1/n^{2.5}$ (for large $n$) upper bound.

The contributions of all four cases, even though cases 2 and 3 are independent of 4, give

$$Prob(\cup_{a=2}^{r} F_a) < 1/n^2.$$

The probability that there exists one or more isolated vertices in $X$ is at most $r(1-p)^{x+r} \leq re^{-p(x+r))}$. This is less than $1/n^{p(x+r)/\log n - 1}$. If we add to this probability the contributions of all four cases, we get a failure probability of at most $\max\{1/n^2, 1/n^{p(x+r)/\log n-1}\}$.

■

## 4.6 Near-optimal and Optimal broadcast

We claim the following Proposition which holds for random graphs $G_{n,p}$ for both cases of $p$ needed, that is $p = p_1 = c\log n/n$, for any $c > 18.4$ for the near-optimal case and $p = p_2 = c\,\mathrm{lglg}n \log n/n$ for any $c > 16$ for the optimal broadcast case.

**Proposition 11** *In $G_{n,p_1}$ (respectively $G_{n,p_2}$), procedure Phase1 (respectively Phase2) for some $\epsilon > 0$, with probability at least $1-1/n^{1+\epsilon}$, reveals edges incident to at most $n/2$ vertices, and the number of vertices labeled with labels taken from the set $\{0, \ldots, \lg n - \mathrm{lglg}n - 5\}$ is equal to $n/(32\lg n)$ vertices. These vertices will receive the message held by the root of $T$ if Broadcast($\lg n - \mathrm{lglg}n - 5$) is executed on $T$.*

Proposition 11 uses Proposition 9.

**Proof of Proposition 11:** In Phase1, for the case of $G_{n,p_1}$ random graphs, we expanded $n/(64\lg n)$ vertices. The probability that a vertex, among them, has a number of children which is at less than $\lg n$ or more than $8.97\lg n$ is, by Proposition 9, at most $(n/(64\lg n))(n^{-2.03}+n^{-2.0149}) = O(n^{-1-\epsilon})$, for some $\epsilon > 0$. Therefore with high probability the tree can be built with the desired number of children per internal vertex. Since with this probability every internal vertex of the tree has at most $8.97\lg n$ children and we have expanded $n/(64\lg n)$ vertices, the total number of vertices in the broadcast tree is with high probability at most $((n/(64\lg n))\cdot 8.97\lg n + 1)$. This number added to $s$, the size of $S$, is at most $n/2$, the maximum number of vertices that will be considered in such a tree finding process.

The case of $G_{n,p_2}$ random graphs is similar to this one.

Hence the probability of failing to find a broadcast tree rooted at some given vertex of a $G_{n,p}$ random graph, as required for the two cases of $p$, is $O(1/n^{1+\epsilon})$, for some $\epsilon > 0$, as claimed. ■

The first stage of a near optimal or optimal broadcast involves the execution of procedure Phase1 or Phase2 leading to the construction of tree $T$, whose root holds the message to be broadcast. The message is subsequently broadcast to $n/(32\lg n)$ vertices of $T$ in $\lg n - \mathrm{lglg}n - 5$ time steps (or to $2^{\lfloor \lg n \rfloor - \lceil \mathrm{lglg} n \rceil - 5}$ vertices in $\lfloor \lg n \rfloor - \lceil \mathrm{lglg} n \rceil - 5$ time steps, in case $\lg n$ and lglg$n$ are not integer numbers) by procedure Broadcast. The edges incident to at most $n/2$ vertices are thus revealed with high probability.

We can distinguish three types of vertices at the end of this stage.

- Type 1 vertices. These are the $n/(32\lg n)$ vertices that possess the message. Let us call this set of vertices $T_1$.

- Type 2 vertices. These are the at most $n/2 - n/(32\lg n)$ vertices that participated in the finding of the broadcast tree but do not possess the message. Hereafter we shall assume that the number of Type 1 and Type 2 vertices add up to $n/2$, even if we had not needed so many vertices when we located the broadcast tree. Let us call this set of vertices $T_2$.

- Type 3 vertices. These are the at least $n/2$ vertices of the graph that were not involved in the finding of the tree and thus have not received the message yet. Let us call this set of vertices $T_3$.

No revelation of edges among Type 3 vertices has occurred. Hence, edges among Type 3 vertices or among Type 3 and Type 1 and 2 vertices are independent of the edges revealed while the broadcast tree was being formed.

### 4.6.1 Near-optimal broadcast in random graphs

In this section we examine near-optimal broadcast in $G_{n,p_1}$ random graphs. The first stage of a near optimal broadcast has been described above.

We now claim a Proposition which is Proposition 2 of [46]. We preserve the terminology and wording of the Proposition of that paper here.

Let $X$ be a subset of the vertices of a $G_{n,p}$ random graph with edge probability $p = 18.4 \log n/n$ and let $F_X$ denote the event that a maximum matching of $X$ into $V - X$ has at least $|X|/2$ unmatched vertices in $X$. Thus when $F_X$ *does not* occur, less than half of the vertices of $X$ are left unmatched.

We give now a stronger version of the statement of Proposition 2 of [46]. We also note that this Proposition will be applied to a subset of the vertices of the random graph.

**Proposition 12 (see also [46])** *Suppose $X$ denotes a subset of the vertices $V$ of the class $G_{n,p}$of random graphs with edge probability $p = 18.4 \log n/n$ and let $F_X$ denote the event that a maximum matching of $X$ into $V - X$ has at least $|X|/2$ unmatched vertices. Fix an integer $x$ with $2 \leq x \leq n/4$. For some positive constant $\gamma$ and for $n$ sufficiently large, the probability that there exists some $X$ with $|X| = x$ for which the event $F_X$ holds is $O(1/n^{1+\gamma})$.*

Proposition 12 will be applied to Type 3 and Type 1 vertices only. The proof follows the one given in [46] with slight modifications only. For $p$ we use the $p = p_1 = 18.4 \log n/n$ of $G_{n,p_1}$ random graphs and $q = 1 - p$. Two cases are also distinguished, one for $2 \leq x \leq n/$ loglog$n$, the other for $x > n/$ loglog$n$ and in that latter case we choose the parameter $a_1$ of [46] to be $\lceil n/\log n \rceil$ here.
We thus prove the first of the two Theorems related to the first set of our results.

**Theorem 16** *In almost all $G_{n,p_1}$ random graphs with edge probability $p_1 = c \log n/n$, for any constant $c > 18.4$, near optimal broadcast can be achieved in $\lg n + o(\lg n)$ time steps.*

**Proof:** The proof consists of three stages, each one establishing claims related to one of three stages of a broadcast of the message to be broadcast. We shall refer to that stage of the proof by referring to the stage of the broadcast this stage of the proof establishes.

The second stage follows the proof in [46]. The other two are novel. We assume $n$ and $\lg n$ are powers of two. We have discussed earlier how we can change the various proofs to cover the remaining cases when $n$ or $\lg n$ is not a power of two.

We first execute Phase1, find $T$ and by executing Broadcast we broadcast the message held by the root of $T$ to $n/(32 \lg n)$ vertices using less than half of the vertices of the graph.

This completes the first stage of the broadcast. The probability that this stage fails is, by Proposition 11, $O(1/n^{1+\epsilon})$, for some positive constant $\epsilon > 0$.

We then use Proposition 12 to broadcast the message to the Type 3 vertices of the graph using successive matchings between vertices that possess the message and vertices that do not.

Let $k_0 = n/(32 \lg n)$ be the number of vertices that hold the message after the first $\lg n - \mathrm{lglg} n - 5$ steps of the first stage of the proof. Let $k_i$ be the number of vertices that will have received the message $i$ time steps after the message had been broadcast to the $k_0$ vertices. Suppose that $k_i \leq n/4$. Let us choose a maximum matching from the set of vertices possessing the message at time step $\lg n - \mathrm{lglg} n - 5 + i$ (there are $k_i$ such vertices) to the ones that have not received it yet. Each vertex of this matching possessing the message transmits it to its matched one that does not. By Proposition 12, with high probability at least half of the $k_i$ vertices will be matched and the following will then hold:

$$k_{i+1} \geq 2(1 - 1/4)k_i,$$

whenever $k_i \leq n/4$. The inequality fails with probability $O(1/n^{1+\gamma})$ by Proposition 12. If we solve the recurrence we get the following inequality:

$$k_i \geq (2(1 - 1/4))^i (n/(32 \lg n)),$$

which fails with probability $O(i/n^{1+\gamma})$ whenever $k_{i-1} \leq n/4$. Therefore, after at most

$$\lceil \frac{3 + \mathrm{lglg} n}{\lg 1.5} \rceil + 1 = \lceil 5.1286 + 1.71 \, \mathrm{lglg} n \rceil + 1$$

time steps in addition to those required for the first stage, more than a quarter of the Type 1 and 3 vertices will have received the message. This completes the second stage of the broadcast.

We now describe the last (third) stage of the near-optimal algorithm. We examine first the Type of vertices considered so far. Among the at least $n/4$ vertices which had received the message during the first two stages are the Type 1 ones and subsequently, at least $n/4 - n/(32 \lg n)$ Type 3 vertices. Let us call this set of Type 3 vertices $T_{31}$ so that $T_3 = T_{31} \cup T_{32}$, $|T_{31}| \geq n/4 - n/(32 \ \lg n)$. There are $n/2 - n/(32 \ \lg n)$ Type 2 vertices in the graph. We attempt to match the Type 3 vertices of $T_{31}$ to an arbitrarily chosen equal number of Type 2 vertices. The number of vertices that will be matched are at least $|T_{31}|$ and at most $|T_2|$. A perfect matching exists with high probability (at least $1 - 1/n^{1.5}$) by Theorem 14. It suffices, since the sets of vertices to be matched are of size at least $n/4 - n/(32 \ \lg n)$, to have $c > 16$ in the expression for the probability $p$ in that theorem and we have $c = 18.4$. We subsequently broadcast the message from each vertex in $T_{31}$ of this complete matching to its matched Type 2 vertex. Let the Type 2 vertices of this matching form set $T_{21}$ and the corresponding Type 3 vertices of $T_{31}$ set $T_{311}$ so that $T_2 = T_{21} \cup T_{22}$, $T_{21} \cap T_{22} = \emptyset$, $T_{31} = T_{311} \cup T_{312}$, $T_{311} \cap T_{312} = \emptyset$. We note that we fix the two sets of vertices $T_{311}$ and $T_{21}$ before we claim Theorem 14 above.

In the following time step we attempt match vertices of $T_{31}$ to vertices of $T_{22}$. We have $n/4 \geq |T_{22}|$ since $|T_{21}| \geq |T_{31}| \geq n/4 - n/(32 \ \lg n)$ and $T_2 = T_{21} \cup T_{22}$. It is possible that after this step up to $n/(32 \ \lg n)$ vertices of $T_{22}$ will be left unmatched. This upper bound is attained when the cardinality of $T_{31}$ equals its lower bound. If $|T_{22}| \geq n/8 - n/(32 \ \lg n)$, we match an equal number of vertices of $T_{31}$ to vertices of $T_{22}$, otherwise we match $T_{22}$ to $T_{31}$. In the former case we claim Theorem 14, in the latter one we claim Theorem 15, with $x \geq n/8$, $p \ x > 2 \ \log n$, $p(x + r) > 3 \log n$. The probability that we do not get a complete matching as required, is thus at most $1/n^{1.5}$ for either of the two cases. Having established such a matching we broadcast the message from each $T_{31}$ vertex of this matching that possesses it to its matched one of $T_{22}$ which does not.

Simultaneously, we match the Type 2 vertices that received the message in the previous step (that is vertices of $T_{21}$) with vertices of $T_{32}$ that have not yet received the message. Since $|T_{21}| \geq n/4 - n/(32 \ \lg n)$ and $|T_{32}| \leq n/4 + n/(32 \ \lg n)$ (because $|T_{31}| \geq n/4 - n/(32 \ \lg n)$ and $|T_3| = n/2$) it is possible that up to $n/(16 \ \lg n)$ vertices of $T_{32}$ will be left unmatched after this step. This upper bound is attained for $|T_{21}| = |T_{31}| = n/4 - n/(32 \ \lg n)$. If

$|T_{32}| \geq n/8 - n/(32 \ \lg n)$ we match an equal number of vertices of $T_{21}$ to vertices of $T_{32}$ otherwise we match $T_{32}$ to $T_{21}$. In the former case we claim Theorem 14 while in the latter one we claim Theorem 15, with $x \geq n/8$, and $px > 2 \ \log n$, $p(x+r) > 3 \log n$. With probability at least $1 - 1/n^{1.5}$ a complete matching, as required for each of the two cases, exists and we can thus broadcast the message from each vertex of $T_{21}$ of the matching to its matched one in $T_{32}$.

At the next time step (the third one of the third stage of the broadcast) we match the Type 2 vertices of set $T_{22}$ that may not have received the message (they are at most $n/(32 \lg n)$) with a number of Type 3 vertices that have received it (say we choose $3n/8$ such Type 3 vertices). At this same third step we also match the at most $n/(16 \lg n)$ Type 3 vertices of set $T_{32}$ that may not have received the message to the Type 2 messages that have it. Theorem 15, with $x \geq n/4$, $p \ x > 2 \ \log n$, proves the existence of a complete matching that saturates $T_{22}$ in the former and $T_{32}$ in the latter case with probability at least $1 - 1/n^2$ for each case.

The probability that some of the three steps of the third stage of the near-optimal broadcast fails is $O(1/n^{1+\epsilon_1})$, for some positive constant $\epsilon_1$. The probability of failure of the second stage is also $O(\text{lglg}n/n^{1+\gamma}) = O(1/n^{1+\epsilon_2})$. The probability of failure of the first stage is given by Proposition 11 and is thus $O(1/n^{1+\epsilon})$, for some $\epsilon > 0$. Therefore, after

$$\lg n - \text{lglg}n + \lceil 5.1286 + 1.71 \, \text{lglg}n \rceil - 1 = \lg n + o(\lg n)$$

time steps the message will have been broadcast to all the vertices of the random graph with high probability (the probability of failure will be $O(1/n^{1+\delta})$, where $\delta = \min\{\epsilon_1, \epsilon_2, \epsilon\}$).

The probability of failure of near-optimal broadcast from some vertex of the random graph is thus $O(1/n^{\delta})$. ■

### 4.6.2 Optimal Broadcast in random graphs

In this section we examine optimal broadcast in $G_{n,p_2}$ random graphs with $p_2 = c \, \text{lglg}n \log n/n$, $c > 16$. For simplicity we shall assume that $n$ is of the form $n = 2^{2^k}$. For the remaining cases we follow the discussion at the end of section 4.2. At the conclusion of the proof we shall examine the case of $n$ not being a power of two. We now prove the following.

**Theorem 17** *In almost all $G_{n,p_2}$ random graphs, with edge probability $p_2 = c\,\mathrm{lglg}n \log n/n$, for any constant $c > 16$, optimal broadcast in $\lceil \lg n \rceil$ time steps is feasible.*

**Proof:** The first stage of the optimal broadcast consists of the execution of Phase2 that constructs tree $T$ whose root holds the message to be broadcast. In $\lg n - \mathrm{lglg}n - 5$ steps the message originally held by the root of $T$ is broadcast to $n/(32\ \lg n)$ (or to $2^{\lfloor \lg n \rfloor - \lceil \mathrm{lglg}n \rceil - 5}$ vertices in $\lfloor \lg n \rfloor - \lceil \mathrm{lglg}n \rceil - 5$ time steps if $\lg n$ and $\mathrm{lglg}n$ are not integer numbers) vertices by executing procedure Broadcast on $T$. The probability of failure of this stage is given by Proposition 11 and is $O(1/n^{1+\epsilon})$, for the $\epsilon$ claimed there.

In the second stage, we split the Type 1 and 3 vertices into $t$ sets ($t$ to be defined later). Set $V_0$ is $T_1$ and thus consists of all Type 1 vertices that got the message in the first $\lg n - \mathrm{lglg}n - 5$ time steps. Then $|V_0| = n/(32 \lg n)$.

Sets $V_1, \ldots, V_{t-1}, V_t$ consist of Type 3 vertices which will be arbitrarily partitioned among these sets. The cardinalities of these sets (similarly to [46]) are chosen to be $|V_i| = 2^{i-1}n/(32 \lg n) + \lfloor n/(8\ (\mathrm{lglg}n + 3)) \rfloor$, $i = 1, \ldots, t-1$. Set $V_t$ holds the remaining Type 3 vertices of the graph. We choose $t-1$ so that $2^{(t-1)}n/(32 \lg n) = n/4$ and thus $t - 1 = \mathrm{lglg}n + 3$. We get that the following holds.

$$\sum_{i=0}^{t-1} |V_i| = 2^{t-1}n/(32 \lg n) + (t-1)\lfloor n/(8(\mathrm{lglg}n + 3)) \rfloor \leq n/4 + n/8$$

Therefore $|V_t| = n/4 + n/(32\ \lg n) - (t-1)\lfloor n/(8(\mathrm{lglg}n + 3)) \rfloor$. We then run the following algorithm, which is identical to step 2 of [46].

**Procedure** Partial-Broadcast

**begin**

$A_0 = V_0$, $B = \emptyset$;

**for** $i = 1$ **to** $(t-1)$ **do**

        **begin**

Find a matching between $A_{i-1}$ and $V_i$ that saturates $A_{i-1}$,

and transmit the message from each vertex of $A_{i-1}$ to its matched

one in $V_i$;

Let $A_i = A_{i-1} \cup \{x : \ x \in V_i \ , x \text{ matched to a vertex in } A_{i-1}\}$;

Let $B = B \cup (V_i - A_i)$;

**end**

**end**

This procedure implements the second stage of the optimal broadcast algorithm.

Theorem 15 with $x = \lfloor n/(8(\text{lglg}n + 3)) \rfloor$ gives that for some $\epsilon_1 > 0$, with probability at least $1 - 1/n^{1+\epsilon_1}$ a matching between $A_{i-1}$ and $V_i$ that saturates $A_{i-1}$ exists. We have lglg$n + 3$ iterations in this procedure and the probability of failure to find a matching that saturates some $A_{i-1}$ is thus $O(\text{lglg}n/n^{1+\epsilon_1}) = O(1/n^{1+\epsilon_2})$ for some positive constant $\epsilon_2 < \epsilon_1$.

The size of set $B$ after the completion of this stage is $(t-1)\lfloor n/(8(\text{lglg}n + 3)) \rfloor \leq n/8$.

The first two stages required $\lg n - 2$ time steps for broadcasting. In the third stage, within the remaining two time steps, we shall broadcast the message to the vertices that have not yet received it.

At this point we examine the types of vertices encountered in the first two stages of the Algorithm. We have:

1. Let $A$ be the set of vertices that have received the message so far. It is $|A| = n/4$ and $A$ consists of Type 1 and Type 3 vertices.

2. Let $B$ be the set of Type 3 vertices used in the construction of sets $V_1, \ldots, V_{t-1}$, but never got the message. We have that $|B| \leq n/8$.

3. $V_t$ is the set of Type 3 vertices that did not participate in the formation of sets $V_1, \ldots, V_{t-1}$ and we have $|V_t| = n/4 + n/(32 \ \lg n) - (t-1)\lfloor n/(8(\text{lglg}n + 3)) \rfloor \geq n/8 + n/(32 \lg n)$.

4. $T_2$ is the set of Type 2 vertices which never received the message in the first stage of the algorithm (when the broadcast tree was found). We have that $|T_2| = n/2 - n/(32 \lg n)$.

In the two time steps of the third stage we disseminate the message to the vertices that have not yet received it. This is accomplished as follows.

**Time step** $\lg$ ***n*-1.** We find a complete matching between $B$ and the vertices in $V_{t-1} \cap A$ that saturate $B$. It is $n/8 - \text{lglg}n - 3 \leq |B| \leq n/8$ and $|V_{t-1} \cap A| = n/8$. No edges between the vertices in the two matched sets have been revealed, although edges between $B$ and $A - V_{t-1}$ have been so. Theorem 14, for two sets of size equal to that of set $B$, proves the existence of a complete matching that saturates $B$ with probability at least $1 - 1/n^{1.5}$.

Let the matched vertices in $V_{t-1} \cap A$ form set $D$. It is $|D| = |B|$. We then match the unmatched vertices of $A - D$ to an equal number of vertices of $V_t$. No edges between the two sets have been revealed. Theorem 14 shows that with probability at least $1 - 1/n^{1.5}$, a matching that saturates $A - D$ exists and given such a matching $n/(32 \lg n)$ vertices of $V_t$ will be left unmatched (since the two sets differ by this number of vertices).

After we have established the two matchings we simultaneously broadcast the message from each vertex in $D$ to its matched one in $B$ and from each vertex in $A - D$ to its matched one in $V_t$. The probability that some of the two matchings fail to exist is at most $2/n^{1.5}$.

At the conclusion of this step $n/(32 \lg n)$ Type 3 vertices of $V_t$ along with the vertices of set $T_2$ will have not received the message.

**Time step** $\lg$ ***n*.** Among the $n/(32 \lg n)$ Type 1 vertices, $n/(64 \lg n)$ of them are vertices that are leaves in the broadcast tree of the first stage and let us refer to these vertices as *Sons*. The remaining $n/(64 \lg n)$ ones are vertices which are connected to the *Sons* in a perfect matching (one-to-one) fashion (that is how the *Sons* got the message from these vertices) and let us refer to them as *Fathers*. By Proposition 9 each of the *Fathers* has at least $\lg n$ descendants (and among them a *Son*) but only at most $\lg n - \text{lglg}n - 5$ of them have received the message. Thus each one of the *Fathers* has a spare Type 2 vertex to broadcast the message it currently holds (and let for simplicity this vertex be the "next" one without a message in the ordering used when Broadcast was executed). This observation provides a matching of the $n/(64 \ \lg n)$

Fathers to an equal number of Type 2 vertices.

No edges connecting each of the *Sons* to Type 2 vertices have been revealed since each one of them is a leaf in the broadcast tree, although an edge has been revealed connecting it to its Father but the Fathers are Type 1, not Type 2 vertices. We can thus claim independence for edges between any vertex among the *Sons* and any Type 2 vertex.

A matching between the Fathers ($n/(64\ \lg n)$ Type 1 vertices) and $n/(64\ \lg n)$ Type 2 vertices has already been established. We then match the following two sets. The first one consists of vertices that possess the message, that is of $n/(64\ \lg n)$ Type 1 vertices (these are the *Sons*) along with $n/2 - n/(32\ \lg n)$ Type 3 vertices. The second set consists of vertices that do not possess the message, that is of the remaining $n/2 - n/(32\ \lg n) - n/(64\ \lg n)$ Type 2 vertices along with the $n/(32\ \lg n)$ Type 3 vertices that did not receive the message in the previous time step. Theorem 14 applied to two equal sets of size $n/2 - n/(64\ \lg n)$ each shows that a matching that saturates all the vertices of the two sets exists with probability at least $1 - 1/n^{1.5}$.

The probability that the third stage fails to execute properly is thus $O(1/n^{1.5})$.

Hence the probability that some of the steps of some of three stages fail is at most the sum of the failure probabilities of the three stages and thus $O(1/n^{1+\theta})$ for some positive constant $\theta$.

Therefore in $\lg n$ (for $n$ a power of two) time steps we have broadcast, with high probability, the original message to all the vertices of the random graph. We have thus proved that from a given vertex we can broadcast a message to all the remaining vertices with probability of failure $O(1/n^{1+\theta})$. Then the probability that some vertex fails to broadcast a message it possesses in $\lg n$ steps (for $n$ a power of two) is $O(1/n^{\theta})$ and the broadcast number of almost all such graphs is thus $\lg n$.

If $n$ is not a power of two, that is $n = 2^k + l$ with $0 < l < 2^k$, then the $\lg n$ in the proof above must be replaced by $\lfloor \lg n \rfloor = k$ and in so many steps the message is broadcast to $2^k$ vertices. There are left $l$ vertices with no received message and their edges to the $2^k$ vertices have not been revealed. We can find a complete matching between the vertices that have received the message and the ones that have not. Then each matched vertex holding the message transmits it to its matched vertex and the algorithm terminates. Theorem 14 if $l > n/4$, or Theorem 15 with $p(x+r) > 3\log n$ otherwise, shows the existence of such a matching with probability at least $1 - 1/n^{1.5}$. This completes the proof of the Theorem. ■

## 4.7 Close-to-optimal and Near-optimal broadcast

We now continue the discussion related to the second set of results obtained in this work. These will be related to close-to-optimal and near-optimal broadcast in various classes of random graphs. We have so far established the validity of Proposition 10. We still need to show that steps 1 through 5 of procedure Phase3 generate edges that involve at most half of the $n$ vertices of the random graph with probability at least $1 - 1/n^{1+\epsilon}$, that is the number of vertices that are taken into consideration in procedure Phase3 is at most $n/2$ (the size of POOL).

**Proposition 13** *In the class $G_{n,p}$ of random graphs, with $p = c\,\mathrm{lglglg}\,n \log n/n$, $c > 16$, procedure Phase3, for an $\epsilon > 0$ that depends on the $\delta, \zeta$ of Proposition 10 and large enough $n$, with probability at least $1 - 1/n^{1+\epsilon}$, generates edges that involve at most $n/2$ vertices. The number of broadcast labeled vertices with labels taken from the set $\{0, \ldots, \lg n - \lceil \mathrm{lglglg}\, n \rceil - 5\}$ is at least $n(2-a)^{\mathrm{lglg} n - \lceil \mathrm{lglglg}\, n \rceil}/(32 \lg n)$, where $a = (1 + 1/\log n)/\log^{1+\zeta} n$.*

The proof of this Proposition for the portion of the tree revealed up to step 3 of Phase3 is proved similarly to Proposition 11 if we use Proposition 10 instead of Proposition 9. The non-obvious part is to establish the result for the internal vertices of the tree expanded in step 5, since the subadditivity argument claimed in the proof related to step 3 is not applicable.

We shall also use the asymptotic result derived from the DeMoivre-Laplace Theorem given in [10] (Theorem 7, part (ii)) and presented as Theorem 2 earlier in this work.

**Proof of Proposition 13:** In Phase3 just after step 2, tree $T$ has $n/(64 \lg n)$ internal vertices by Claims 5 and 6. The probability that an internal vertex obtains a number of children which is at most $\lg n$ or at least $9.13 \lg n$ is by Proposition 10 at most $n^{-2-\delta}$. By subadditivity with probability at least $1-(n/(64\lg n))\, n^{-2-\delta} \geq 1-n^{-1-\delta}$ for the $\delta > 0$ determined in Proposition 10, tree $T$ after step 2 of Phase3, has the desired number of children per internal vertex. Since every internal vertex of the tree has thus at most $9.13\lg n$ children and we have expanded $n/(64\lg n)$ vertices, the total number of vertices in the broadcast tree $T$ just after step 2 of Phase3 is with probability at least $1-1/n^{1+\delta}$ at most $((n/(64\lg n))\cdot 9.13\lg n + 1)$ (the one counts the root of $T$). This number added to $s$, the size of $S$, is definitely at most $n/2$ (the size of POOL), the maximum number of vertices that will be considered in such a tree finding process.

We still need to show that if we add to this number the vertices involved in the generation of edges by steps 3 through 5 of Phase3 we can still get a sum of at most $n/2$ vertices.

This latter part is more complicated to prove.

When procedure Expand in step 5 is called vertices are expanded in increasing order of their labels.

We call such an expansion of an individual vertex "successful" if the number of children this vertex obtains is at least $\mathrm{lglg}n$ and at most $6.25566\,\mathrm{lglg}n$. The probability that such an expansion for an individual vertex is not "successful" (or is "unsuccessful") is, by Proposition 10, at most $1/(\log^{1+\zeta} n)$. If an expansion is "unsuccessful" we throw away from $T$ the vertices of $S$ involved in step 4 of Expand and we also never add these vertices to set $S$ again. The term "successful" (or "unsuccessful" ) expansion will also be used in this proof to denote the vertex that has successfully (or not) been expanded.

Let $a = (1+1/\log n)/(\log^{1+\zeta} n)$. We first study the expansion of vertices labeled $\lg n - \mathrm{lglg}n - 5$ in the call to Expand issued in step 5 of Phase3.

After the successful completion of step 2 of Phase3, by Claim 5, there are $n/(32\lg n)$ vertices labeled at most $\lg n - \mathrm{lglg}n - 5$ and these vertices are the ones that will receive the message if Broadcast($\lg n - \mathrm{lglg}n - 5$) is executed on this instance of $T$. Among these vertices, by Claim 5, there are $n/(64\lg n)$ ones labeled $\lg n - \mathrm{lglg}n - 5$. These vertices are the ones that will be "expanded" first when Expand is called in step 5 of Phase3. The probability of an individual "unsuccessful" expansion is at most $1/(\log^{1+\zeta} n)$ by Proposition 10. Thus the probability that among these $n/(64 \lg n)$ expansions we have at least $(an)/(64 \lg n)$ "unsuccessful" ones is, by the Angluin-Valiant bounds, at most $\exp\left(-O(n/(\log^{4+\zeta} n))\right)$ which is asymptotically for large $n$ at most $1/n^3$. Thus with probability at least $1 - 1/n^3$ the number of "unsuccessful" expansions is at most $(an)/(64 \lg n)$, the number of "successful" ones is at least $n/(64 \lg n) - (an)/(64 \lg n)$ (and at most $n/(64 \lg n)$), and each one of these latter expanded vertices will have a number of edges, involving at most $6.25566 \,\mathrm{lglg} n$ vertices of the random graph, revealed during this process.

We still need to count the number of vertices revealed by the "unsuccessful" expansions. Let $U$ denote, as in Expand, the random variable that gives the number of prospective children of a vertex to be expanded and we focus our attention on "unsuccessful" expansions. To estimate $U$ for such expansions we claim Theorem 2. Let us substitute in Theorem 2 the edge probability $p$ of the class of random graphs examined here, $q = 1 - p$, $n = s$ so that $pn = ps = 5 \,\mathrm{loglog} n$. We get that for a $u = \log n/(5 \,\mathrm{loglog} n)$ the conditions of Theorem 2 are satisfied for sufficiently large $n$ and thus the probability

$$Prob(S_{n,p} \geq upn) = Prob(U \geq upn) = Prob(U \geq \log n) \leq \left(5e \,\mathrm{loglog} n/\log n\right)^{\log n}.$$

For sufficiently large $n$ this probability is at most $1/n^3$. Thus, with probability at least $1 - 1/n^3$ an "unsuccessful" expansion generates edges involving at most $\log n$ vertices of $S$. We have at most $(an)/(64 \lg n) < n/(2 \lg n)$ such expansions and thus with probability at least $1 - 1/(2n^2 \lg n)$, by subadditivity, the number of vertices revealed by such expansions is at most $((an)/(64 \lg n)) \log n$.

The discussion of the two previous paragraphs covers the iterations of the external **for** loop in Expand for all vertices labeled $\lg n - \mathrm{lglg} n - 5$. We need to examine all other remaining iterations of Expand. Our claims below will be based on the arguments presented in the preceding two paragraphs.

Let us define $A_0 = n/(32 \lg n)$ to be the vertices that at the end of step 2 in Phase3 were labeled 0 through $\lg n - \mathrm{lglg} n - 5$. Among these vertices, $A_0/2$ were labeled $\lg n - \mathrm{lglg} n - 5$ and were leaves in $T$ just after the completion of step 2 of Phase3. The remaining $A_0/2$ ones were internal vertices of $T$, each such vertex had at least $\lg n$ children (with probability at least $1 - n^{-2-\delta}$) and each label from the set $\{\lg n - \text{lglg}n - 5, \ldots, L\}$, $L < \lg n$, was assigned to exactly one of these children.

Following the new definitions and the justification of the preceding paragraphs we get that with probability at least $1-(1/n^3+1/(2n^2 \lg n)) \geq 1-1/(n^2 \lg n)$, for $n > 4$, among the $A_0/2$ vertices labeled $\lg n - \text{lglg}n - 5$ we had at least $A_0/2 - \lfloor aA_0/2 \rfloor$ "successful" expansions, (that is, the number of children of each such expanded vertex is given by Proposition 10) while the at most $\lfloor aA_0/2 \rfloor$ "unsuccessful" expansions generated edges involving at most $\lfloor aA_0/2 \rfloor \log n \leq a \log n \; A_0/2$ vertices.

This will constitute the basis of the inductive argument that will follow.

In the subsequent discussion $A_i$, $i \geq 0$, will denote the number of vertices labeled $\lg n - \text{lglg}n - 5 + i$ after the expansion of all vertices labeled $\lg n - \text{lglg}n - 5 + i - 1$ has been completed during the execution of Expand in step 5 of Phase3. $B_i$ will denote the vertices labeled 0 through $\lg n - \text{lglg}n - 5 + i$ after this expansion. This number will correspond to the number of vertices that will receive the message held by the root of $T$ if Broadcast($\lg n - \text{lglg}n - 5 + i$) is executed on it.

When step 9 of Expand is executed for the last vertex of $P$ labeled $\lg n - \text{lglg}n - 5$ the number of vertices $A_1$ labeled $\lg n - \text{lglg}n - 4$ is equal to the sum of $A_0/2 = n/(64 \lg n)$ since each internal vertex of $T$ after step 2 of Phase3 had one vertex adjacent to it labeled $\lg n - \text{lglg}n - 4$, and a quantity which is at least $A_0/2 - \lfloor aA_0/2 \rfloor$ since each one among these vertices is a leaf of this instance of $T$ whose labeled $\lg n - \text{lglg}n - 5$ father had a "successful" expansion. Thus $A_1 \geq A_0 - \lfloor aA_0/2 \rfloor$. We also have trivially that $A_1 \leq A_0$. The number of internal vertices of $T$ at this point is also $A_1$, equal to the $A_0/2$ old ones plus the newly expanded vertices due to "successful" expansions. The number of vertices $B_1$ labeled 0 through $\lg n - \text{lglg}n - 4$ which will receive the message held by the root of $T$ if Broadcast($\lg n - \text{lglg}n - 4$) is executed on it, is such that $B_1 \geq 2A_0 - \lfloor aA_0/2 \rfloor$. It is also $B_1 \leq 2B_0 = 2A_0$. The "unsuccessful" expansions generated edges involving at most $\lfloor aA_0/2 \rfloor \log n$ vertices. Each "successful" expansion generated edges involving at most $6.2556\, \text{lglg}n$ vertices and we had $A_1 \le A_0$ such expansions.

During the following iterations of Expand vertices labeled $\lg n - \mathrm{lglg} n - 4$ are expanded similarly to vertices labeled $\lg n - \mathrm{lglg} n - 5$ in the previous paragraphs. Each internal vertex of $T$ will have one vertex labeled $\lg n - \mathrm{lglg} n - 3$ and after all leaves labeled $\lg n - \mathrm{lglg} n - 4$ are expanded we shall have with probability at least $1 - 1/(n^2 \lg n)$, $A_1 - \lfloor aA_1 \rfloor$ "successful" expansions with the "unsuccessful" expansions generating edges involving at most $\lfloor aA_1 \rfloor \log n \le a \log n A_1$ vertices. The number $A_2$ of vertices labeled $\lg n - \mathrm{lglg} n - 3$, and also equal to the number of internal vertices of $T$ at this stage of the expansion, is such that $A_2 \ge 2A_1 - \lfloor aA_1 \rfloor$, and $A_2 \le 2A_1$. The number of vertices $B_2$ is such that $B_2 \ge B_1 + 2A_1 - \lfloor aA_1 \rfloor$, and $B_2 \le 2B_1$.

This inductive argument yields that at the completion of the $i(\ge 2)$-th expansion cycle, when all vertices labeled $\lg n - \mathrm{lglg} n - 5 + (i-1)$ are expanded, we have that $B_i \ge B_{i-1} + 2A_{i-1} - \lfloor aA_{i-1} \rfloor$ and $A_i \ge 2A_{i-1} - \lfloor aA_{i-1} \rfloor$. It is also that case that $A_i \le 2A_{i-1}$ and $B_i \le 2B_{i-1}$. The number of vertices revealed by "unsuccessful" expansions is at most $\lfloor aA_{i-1} \rfloor \log n$, and the number of "successful" expansions were at least $A_{i-1} - \lfloor aA_{i-1} \rfloor$. Each "successful" expansion generated edges involving at most $6.2556\, \mathrm{lglg} n$ vertices. The number of internal vertices of $T$ at this stage of the expansion is also $A_i$.

For $i \ge 2$ the previous inequalities give

$$\begin{aligned} A_i &\ge 2A_{i-1} - \lfloor aA_{i-1} \rfloor \\ &\ge (2-a)A_{i-1} \\ &\ge (2-a)^{i-1} A_1 \\ &\ge (2-a)^{i-1}(1-a/2)A_0 \end{aligned}$$

Similarly, for $i \ge 2$ and given that $0 < a < 1$ we get

$$\begin{aligned} B_i &\ge B_{i-1} + 2A_{i-1} - \lfloor aA_{i-1} \rfloor \\ &\ge B_{i-1} + (2-a)A_{i-1} \\ &\ge (2-a/2)A_0 + \left[(2-a) + \ldots + (2-a)^{i-1}\right](1-a/2)A_0 \end{aligned}$$

$$
\begin{aligned}
&\geq A_0 \left[ (2-a)^i + \frac{(a/2)(2-a)^i}{1-a} - \frac{a/2}{1-a} \right] \\
&\geq A_0 (2-a)^i \\
&\geq (n/(32 \lg n))(2-a)^i,
\end{aligned}
$$

The largest label a vertex can get is given by the value parameter $d$ is assigned in step 4 of Phase3. Substituting $d = \mathrm{lglg}n - \lceil \mathrm{lglglg}\, n \rceil$ for $i$ into the expression for $B_i$ derived above, the second claim of Proposition 13 is shown. It is also the case that $B_d \geq n/(32\ 2^{\lceil \mathrm{lglglg}\, n \rceil})$. For the base case we have that $B_1 \geq 2A_0 - \lceil aA_0/2 \rceil$.

The number of internal vertices of $T$ at most doubles at the completion of an expansion cycle and thus for $i \geq 2$

$$
\begin{aligned}
A_i &\leq 2A_{i-1} \\
&\leq 2^{i-1} A_1 \\
&\leq 2^{i-1} A_0,
\end{aligned}
$$

and $A_1 \leq A_0$, while for $B_i$, $i \geq 1$, we have that

$$
\begin{aligned}
B_i &\leq 2B_{i-1} \\
&\leq 2^i A_0.
\end{aligned}
$$

When vertices labeled $\lg n - \mathrm{lglg}n - 5 + (i-1)$ are expanded in Expand, if for any reason the expansion of such vertices fails to follow our probabilistic claims we stop Expand and declare failure of Expand, Phase3, and the Broadcast. This check is performed in step $4'$.

The probability that this happens is at most $1/(n^2 \lg n)$ for the expansion of vertices labeled $\lg n - \mathrm{lglg}n - 5 + (i-1)$ and since the largest value $i$ assumes is at most $\mathrm{lglg}n - \lceil \mathrm{lglglg}\, n \rceil$, by subadditivity, the failure probability is at most $1/n^2$, for the whole execution of Expand in step 5 of Phase3.

The number of vertices involved in the generation of edges in the first call to Expand in step 2 of Phase3 is at most $(n/(64 \lg n))9.13 \lg n + 1$.

The number of vertices involved in "unsuccessful" expansions up to the expansion of vertices labeled $\lg n - \mathrm{lglg}n - 5 + (i-1)$ $(i \geq 2)$ is at most

$$\left((a/2)A_0 + a\sum_{j=1}^{i-1} A_j\right)\log n \quad \leq \quad \left((a/2)A_0 + a2^i A_0\right)\log n.$$

For $i = d = \mathrm{lglg}n - \lceil \mathrm{lglglg}\, n \rceil$ this is at most $n/\,\mathrm{lglg}n = o(n)$. The size $s$ of $S$ during the call to Expand in step 5 of Phase3 remains equal to $(5n\,\mathrm{loglog}n)/(16\,\mathrm{lglglg}\,n \log n)$. The number of vertices involved in the generation of edges in the call to Expand in step 5 of Phase3 by "successful" expansions, whose number is $A_0/2 + \sum_{i=1}^{d-1} A_i$ is thus at most $(2^d n/(64 \lg n))6.25566\,\mathrm{lglg}n < (n/(64\,\mathrm{lglg}n))6.25566\,\mathrm{lglg}n$. If we add up all these terms their sum is at most $n/2$ for sufficiently large $n$.

As we have already claimed the second call to Expand fails with probability at most $1/n^2$. The first call fails with probability at most $1/n^{1+\zeta}$ giving an overall failure probability, by subadditivity, of at most $1/n^{1+\epsilon}$ for an $\epsilon > 0$ that depends on $\zeta$. ■

We can thus distinguish three types of vertices at the conclusion of step 5.

- Type 1 vertices. These are the at least $n(2-a)^{\mathrm{lglg}n - \lceil \mathrm{lglglg}\, n \rceil}/(32 \lg n)$ (from the expression for $B_i$) vertices that possess the message. Let us call this set of vertices $T_1$.

- Type 2 vertices. These are the at most $n/2 - n(2-a)^{\mathrm{lglg}n - \lceil \mathrm{lglglg}\, n \rceil}/(32 \lg n)$ vertices that participated in the finding of the broadcast tree but do not possess the message. Hereafter we shall assume that the number of Type 1 and Type 2 vertices add up to $n/2$ even if we had not needed so many vertices when $T$ was built in Phase3. Let us call this set of vertices $T_2$.

- Type 3 vertices. These are the at least $n/2$ vertices of the graph that were neither involved in the finding of the tree nor have received the message yet. Let us call this set of vertices $T_3$.

No revelation of edges among Type 3 vertices has occurred hence, edges among Type 3 vertices or among Type 3 and Type 1,2 vertices are independent of the edges revealed while the broadcast tree was being formed.

### 4.7.1 Close-to-Optimal Broadcast in random graphs

In this section we examine close-to-optimal broadcast in $G_{n,p}$ random graphs, where $p = c\ \mathrm{lglglg}\, n\ \log n/n$, for any constant $c > 16$. For simplicity we shall assume that $n$ is of the form $n = 2^{2^k}$. At the conclusion of the proof we shall examine the case of $n$ not being a power of two. We now prove the following.

**Theorem 18** *In almost all $G_{n,p}$ random graphs, with edge probability $p = c\,\mathrm{lglglg}\, n \log n/n$, for any constant $c > 16$, and for large enough $n$ close-to-optimal broadcast in $\lceil \lg n \rceil + 1$ time steps is feasible.*

The proof is very similar to that of Theorem 17. What changes here is procedure Phase. In procedure Phase3, steps 3 through 5 appear there in addition to steps 1 through 2. Despite such additions all five steps, require overall the generation of edges that involve no more than $n/2$ vertices.

**Proof:** In the proof to follow we assume $n = 2^{2^k}$, $k$ a positive integer. The remaining cases follow from this proof if we take into consideration the discussion at the end of section 4.2. The first stage of the close-to-optimal broadcast requires the execution of Phase3. When Broadcast is executed on $T$ (if run simultaneously with the construction of $T$) its labeled vertices will receive the message at the time step indicated by their particular label. In the first $\lg n - \mathrm{lglg} n - 5$ steps the message, originally held by the root of $T$, is broadcast to $n/(32\ \lg n)$ vertices just as in Theorem 17. At the end of $\mathrm{lglg} n - \lceil \mathrm{lglglg}\, n \rceil$ additional steps the message will be held by at least $n(2-a)^{\mathrm{lglg} n - \lceil \mathrm{lglglg}\, n \rceil}/(32 \lg n)$ vertices. The probability of failure of this stage is given by Proposition 13 and is $O(1/n^{1+\epsilon})$ for the $\epsilon$ claimed there.

The arguments in the remainder of the proof are essentially those of Theorem 17.

In the second stage we split the Type 1 and 3 vertices into $t+1$ sets ($t$ to be defined later) $V_0, \ldots, V_t$. Set $V_0$ is $T_1$ and thus consists of all Type 1 vertices that got the message in the first $\lg n - \lceil \mathrm{lglglg}\, n \rceil - 5$ time steps. It is also $|V_0| \geq n(2-a)^{\mathrm{lglg}n - \lceil \mathrm{lglglg}\, n \rceil}/(32 \lg n)$ and $|V_0| \leq n/(32\ 2^{\lceil \mathrm{lglglg}\, n \rceil})$ by the fact that $B_i \leq 2B_{i-1}$ as shown in the proof of Proposition 13.

Let $t-1 = \lceil \mathrm{lglg} n + 3 - (\mathrm{lglg} n - \lceil \mathrm{lglglg}\, n \rceil) \lg(2-a) \rceil$. We distinguish two cases.

**Case 1:** In case $t-1 = 3 + \lceil \mathrm{lglglg}\, n \rceil$ the claims to be proven are identical to the ones presented in Theorem 17 and omitted here. A term used to express the size of set $V_i$ there need only to be changed, that is we need to have $\lfloor n/(8(\lceil \mathrm{lglglg}\, n \rceil + 3)) \rfloor$ instead of $\lfloor n/(8(\mathrm{lglg} n + 3)) \rfloor$ there.

**Case 2:** In case $t-1 = 4 + \lceil \mathrm{lglglg}\, n \rceil$ (the only other alternative) the proof follows.

Let sets $V_1, \ldots, V_{t-1}, V_t$ with sizes to be determined shortly, consist of Type 3 vertices arbitrarily distributed among these sets. The cardinalities of these sets (similarly to [46]) are chosen to be $|V_i| = 2^{i-1}|V_0| + \lfloor n/(8\,(\lceil \mathrm{lglglg}\, n \rceil + 3)) \rfloor$, $i = 1, \ldots, t-2$.
We choose $t-1$ so that $2^{(t-1)} V_0 \geq n/4$ thus the choice $t-1 = \lceil \mathrm{lglg} n + 3 - (\mathrm{lglg} n - \lceil \mathrm{lglglg}\, n \rceil) \lg(2-a) \rceil$ claimed earlier. Since for this case $t-1$ takes the value $t-1 = 4 + \lceil \mathrm{lglglg}\, n \rceil$ we shall also choose $|V_{t-1}| = n/8$. Set $V_t$ holds the remaining Type 3 vertices of the graph.

We note that

$$
\begin{aligned}
\lg n - \lceil \mathrm{lglglg}\, n \rceil - 5 + (t-1) &= \lg n - \lceil \mathrm{lglglg}\, n \rceil - 5 \\
&+ \lceil \mathrm{lglg} n + 3 - (\mathrm{lglg} n - \lceil \mathrm{lglglg}\, n \rceil) \lg(2-a) \rceil \\
&\leq \lg n - 1 + \lceil \mathrm{lglglg}\, n \rceil (\lg(2-a) - 1) \\
&+ \mathrm{lglg} n\,(1 - \lg(2-a))\,.
\end{aligned}
$$

Asymptotically for large $n$ the last two terms of the last inequality tend to zero and thus an integer upper bound on the number of steps for the first two stages of the close-to-optimal broadcast algorithm is $\lg n - 1$. If we add to this time the two steps of the third stage to be described below, we get at most $\lg n + 1$ time steps for all three stages of the broadcast algorithm.

We now try to find a lower bound for $(2-a)^{\lg\lg n - \lceil \lg\lg\lg n \rceil}$ and subsequently for $2^{t-2}|V_0|$. We have that for $0 < x < 1$

$$\lg (2-x) = 1 - x/(2\log 2) + O(x^2),$$

and thus $\lg (2-x) \geq 1 - x/(2\log 2)$. For $x = a$, we get $(2-a) \geq 2^{1-a/(2\log 2)}$ and thus

$$\begin{aligned}
2^{t-2}|V_0| &\geq 2^{3+\lceil \lg\lg\lg n \rceil}(n(2-a)^{\lg\lg n - \lceil \lg\lg\lg n \rceil}/(32\lg n)) \\
&\geq (n/(32\lg n))2^{(1-a/(2\log 2))(\lg\lg n - \lceil \lg\lg\lg n \rceil)}\ 2^{3+\lceil \lg\lg\lg n \rceil} \\
&\geq (n/4)2^{-(a\lg\lg n)/(2\log 2)} \\
&\geq 0.75(n/4),
\end{aligned}$$

for $a = (1 + 1/\log n)/(\log^{1+\zeta} n)$, $n > 600$.
We thus get that the following holds.

$$\sum_{i=0}^{t-1} |V_i| = 2^{t-2}|V_0| + (n/8) + (t-2)\lfloor n/(8(\lceil \lg\lg n \rceil + 3)) \rfloor.$$

We note that $2^{t-2}|V_0| \leq n/4$ and from a previous inequality $2^{t-2}|V_0| \geq 0.75n/4$. Therefore, $|V_t| = n/2 - n/8 - 2^{t-2}|V_0| + |V_0| - (t-2)\lfloor n/(8(\lceil \lg\lg n \rceil + 3)) \rfloor \geq |V_0|$. We then run the following algorithm which is identical to step 2 of [46] and of Theorem 17.

**Procedure** Partial-Broadcast

**begin**

$A_0 = V_0$, $B = \emptyset$;

**for** $i = 1$ **to** $(t-2)$ **do**

  **begin**

  Find a matching between $A_{i-1}$ and $V_i$ that saturates $A_{i-1}$,

and transmit the message from each vertex of $A_{i-1}$ to its matched
one in $V_i$;
Let $A_i = A_{i-1} \cup \{x : x \in V_i , x$ matched to a vertex in $A_{i-1}\}$;
Let $B = B \cup (V_i - A_i)$;
**end**

Find a matching between $A_{t-2}$ and $V_{t-1}$ saturating $V_{t-1}$,
and transmit the message from each vertex of $A_{t-2}$ to its matched
one in $V_{t-1}$;
Let $A_{t-1} = A_{t-2} \cup V_{t-1}$;
**end**

This procedure implements the second stage of the optimal broadcast algorithm.

Theorem 15 with $x = \lfloor n/(8(\lceil \lg\lg\lg n \rceil + 3)) \rfloor$ gives that, for some $\epsilon_2 > 0$, with probability at least $1 - 1/n^{1+\epsilon_2}$ a matching between set $A_{i-1}$ (no connection with the $A_{i-1}$ of the proof of Proposition 13) and $V_i$ that saturates $A_{i-1}$, for $1 \leq i \leq t-2$, exists thus creating a set $A_i$ whose size is double of that of $A_{i-1}$. We have $\lceil \lg\lg\lg n \rceil + 3$ iterations in this procedure and thus the probability that for some $i$ there is no matching saturating $A_{i-1}$ is $O(\lg\lg n/n^{1+\epsilon_2}) = O(1/n^{1+\epsilon_3})$ for a positive constant $\epsilon_3 < \epsilon_2$. If this is the case, we declare failure of procedure Partial-Broadcast and the broadcast algorithm. For the last step involving $A_{t-2}$ and $V_{t-1}$ since the two sets are of size at least $n/8$ (note that $|A_{t-2}| = 2^{t-2}|V_0| \geq 0.75n/4$), Theorem 14 can be claimed and with probability at least $1 - 1/n^{1.5}$ there exists a matching saturating $V_{t-1}$. Overall the failure probability of this procedure is $O(1/n^{1+\epsilon_1})$ for some $\epsilon_1 > 0$ that depends on $\epsilon_3$.

The size of set $B$ after the completion of this stage is $(t-2)\lfloor n/(8(\lceil \lg\lg\lg n \rceil + 3)) \rfloor \leq n/8$.

The first two stages together required at most $\lg n - 1$ (and at least $\lg n - 2$) time steps for broadcasting. In the third stage, within the remaining two time steps, we shall broadcast the message to the vertices that have not yet received it.

At this point we examine the types of vertices encountered in the first two stages of the Algorithm. We have with high probability that:

1. Let $A$ be the set of vertices that have received the message so far. It is $|A| = 2^{t-2}|V_0| + n/8$ and $A$ consists of Type 1 and Type 3 vertices.

2. Let $B$ be the set of Type 3 vertices used in the construction of sets $V_1, \ldots, V_{t-1}$, but never got the message. We have that $|B| \leq n/8$.

3. $V_t$ is the set of Type 3 vertices that did not participate in the formation of sets $V_1, \ldots, V_{t-1}$ and we have $|V_t| \geq |V_0|$. From an earlier estimate of $|V_t|$ we get $|V_t| \leq (n/2) - (n/8) - 0.75n/4 + |V_0| - (n/8) + (t-2)$.

4. $T_2$ is the set of Type 2 vertices which never received the message in the first stage of the algorithm when the tree $T$ was found. We have that $|T_2| = n/2 - |V_0|$.

In the following two time steps of the third stage we disseminate the message to the vertices that have not yet received it. This is accomplished as follows.

**Time step $\lg$ *n*-1.** We find a complete matching between $B$ and the vertices in $V_{t-1} \cap A$ that saturates $B$. It is $n/8 - \lceil \mathrm{lglglg}\, n \rceil - 3 \leq |B| \leq n/8$ and $|V_{t-1} \cap A| = n/8$. No edges between the vertices in the two sets have been revealed although edges between $B$ and $A - V_{t-1}$ have been so. Thus Theorem 14 for two sets of size equal to that of set $B$, proves the existence of a complete matching that saturates $B$ with probability at least $1 - 1/n^{1.5}$.

Let the matched vertices in $V_{t-1} \cap A$ form set $D$. It is $|D| = |B|$. We then match the unmatched vertices of $A - D$ to an equal number of vertices of $V_t$. No edges between the two sets have been revealed. We note that $|A| - |D| \geq 0.75n/4$ while $|V_t| \leq 0.25n/4 + |V_0| + (t-2)$. The size of the difference of $|A - D|$ and $|V_t|$ make Theorem 15 with $p(x+r) > 3$ applicable so that with probability at least $1 - 1/n^2$ a matching that saturates $V_t$ exists.

After we have established the two matchings we simultaneously broadcast the message from each vertex in $D$ to its matched one in $B$ and from each vertex in $A - D$ to its matched one in $V_t$. The probability that some of the two matchings fail to exist is at most $2/n^{1.5}$.

At the conclusion of this step only vertices of set $T_2$ will have not received the message.

**Time step $\lg n$.** We match the $T_2$ vertices with an equal number of Type 3 vertices that have the message. Theorem 14 shows that such a matching exists with probability at least $1 - 1/n^{1.5}$.

Therefore the probability that the third stage fails to execute properly is $O(1/n^{1.5})$.

Hence the probability that some of the steps of some of the three stages fail is at most the sum of the failure probabilities of the three stages and thus $O(1/n^{1+\theta})$ for some positive constant $\theta$.

Therefore, in $\lg n$ (for $n$ a power of two) time steps we have broadcast, with high probability, the original message to all the vertices of the random graph. We have proved that from a given vertex we can broadcast a message to all the other vertices of a random graph, with probability of failure $O(1/n^{1+\theta})$. Then the probability that some vertex fails to broadcast a message it possesses in $\lg n$ steps (for $n$ a power of two) is $O(1/n^{\theta})$, and the broadcast number of almost all such graphs is $\lg n$.

If $n$ is not a power of two, that is $n = 2^k + l$ with $0 < l < 2^k$, then the $\lg n$ in the proof above must be replaced by $\lfloor \lg n \rfloor = k$ and in so many steps the message is broadcast to $2^k$ vertices. There are left $l$ vertices with no message received and their edges to the $2^k$ vertices have not been revealed. We can find a complete matching between the vertices that have received the message and the ones that have not. Then each matched vertex with the message transmits it to its matched one and the algorithm terminates. Theorem 14 if $l > n/4$, or Theorem 15 otherwise, shows the existence of such a matching with probability at least $1 - 1/n^{1.5}$. This case completes the proof of the Theorem. ■

### 4.7.2 Near-optimal broadcast in random graphs - An improved bound

In this section we examine near-optimal broadcast in $G_{n,p}$ random graphs, where $p = c \log n/n$, $c > 18.4$. We are going to improve the bounds claimed earlier in this section by using the technique described in the previous section. We need to prove first a sequence of Propositions for this $p$ analogous to Propositions 10 and 13.

Procedure Phase3 can be used for this case if the values of parameter $s$ change there as indicated in the Proof below.

**Proposition 14** *In the class $G_{n,p}$ of random graphs with $p = c \log n/n$, for any constant $c > 18.4$, when procedure Phase3 is executed, for $n > 600$ and some positive $\delta$ (respectively $\zeta$), with probability at least $1 - 1/n^{2+\delta}$ (respectively $1 - 1/\log^{1+\zeta} n$), any one fixed internal vertex of $T$ between step 1 (respectively step 3) and step 3 (respectively step 6) of Phase3, obtains at least $\lg n$ (respectively $\mathrm{lglg}n$) and at most $8.97 \lg n$ (respectively $6.21\,\mathrm{lglg}n$) children. The values of parameter $s$ in Procedure Phase3 need only be changed for this proposition to hold.*

**Proof:** The first part of the Proposition related to steps 1 through 3 was proven in the first part of the proof of Proposition 9 and we use here the $s$ used there. For steps 3 through 6 if we take $p|S| = 0.269\ 18.4\,\mathrm{loglog}n$, with $s = |S| = (0.269 n\,\mathrm{loglog}n)/\log n$, use the Angluin-Valiant bounds and retain the terminology used there, we have for the random variable $U$, through the call to Expand in step 5 of Phase3:

$$Prob(U \leq \mathrm{lglg}n) \leq \log^{-1.078} n,$$

and

$$Prob(U \geq 6.21\,\mathrm{lglg}n) \leq \log^{-1.075} n,$$

The Proposition then follows. ■

A Proposition similar to Proposition 13 directly follows from Proposition 14. Given this, the following Theorem would hold.

**Theorem 19** *In almost all $G_{n,p}$ random graphs with edge probability $p = c \log n/n$, for any constant $c > 18.4$, near optimal broadcast can be achieved in $\lg n + o(\mathrm{lglg}n) = \lg n + o(\lg n)$ time steps.*

The proof is identical to the one Theorem 16. The only difference is that $o(\lg n)$ here is $\Theta(\mathrm{lglglg}\, n)$ instead of $\Theta(\mathrm{lglg}n)$ in Theorem 16.

**Proof:** We refer to the Proof of Theorem 16 for details. The first stage of the broadcast algorithm is the only difference between this proof and the proof of that theorem. We execute in procedure Phase3 steps 3 through 5 in addition to steps 1 through 2 since the broadcast tree used in the proof of Theorem 16 is the tree built by steps 1 through 2 of Phase3. This way after $\lg n - \lceil \lg\lg\lg n \rceil - 5$ steps at least $n(2-a)^{\lg\lg n - \lceil \lg\lg\lg n \rceil}/(32 \lg n)$ vertices of $T$ (and the graph) receive the message.

The solution for $i$ of

$$k_i \geq (2(1-1/4))^i n(2-a)^{\lg\lg n - \lceil \lg\lg\lg n \rceil}/(32 \lg n) \geq n/4,$$

given that (if we also make use of a lower bound for the left hand side of the inequality below shown in the proof of Theorem 18)

$$n(2-a)^{\lg\lg n - \lceil \lg\lg\lg n \rceil}/(32 \lg n) \geq (0.75n/4)2^{-3-\lceil \lg\lg\lg n \rceil},$$

if added to the time of the first stage and the three time steps required for the third stage to follow gives the desired broadcast time.

The remaining three steps of the third stage, as in Theorem 16, require only the changes of the sizes of the various sets to reflect the changes indicated above and the claim that only three more steps are required to deliver the message to the vertices that do not possess it is similar to the one in Theorem 16. For more details we refer to that proof. ■

# Chapter 5

# Bulk-Synchronous Parallel Computation

## 5.1 The Model

The bulk-synchronous parallel or BSP model as described in [52] consists of three parts:
(i) a number of processor/memory components. Here we will assume that each consists of a sequential processor with a block of local memory.
(ii) a router that can deliver messages (typically to implement read and write operations) point to point among the components, and
(iii) facilities for globally synchronizing, in barrier style, all or a subset of the components.

The parameters of such a machine are $p$, the number of processor/memory components, $L$, the minimal time (measured in terms of basic local computation steps) between successive synchronization operations, and $g$ the ratio of the total throughput of the whole system in terms of basic computational operations, to the throughput of the router in terms of words of information delivered.

Computation on this model proceeds in a succession of supersteps. In each superstep each processor is given a task that can be executed using data already available there locally before the start of the superstep. The task may include local computation steps, message transmissions and message receipts. To simplify the description of the performance of our algorithms, in this chapter we shall assume that each superstep is either pure computation or pure communication (and not a mixture of both). In this way the two kinds of costs are easily separated. In this work, we shall introduce the notion of a *stage* that is, a sequence of one or more supersteps performing some task. A stage should not be considered as part of the definition and description of the BSP model. Its existence facilitates the exposition of various algorithms only.

The model has two variants, depending on how synchronization is treated, that affect performance to multiplicative constant factors. In one the synchronizer checks at every $L$ time units whether the tasks that are synchronized in the current superstep have all finished, and only when it discerns that they have does the machine proceed to the next superstep. In this chapter, again for simplifying the analysis, we will use the alternative model where a superstep can complete any time after the first $L$ units. More specifically, each superstep is charged $\max\{L, x+hg\}$ basic time units, where $x$ is the maximum number of local computation steps executed by any processor, and $h$ is the maximum number of packets that any processor sends or receives in that superstep. [N.B. If $h_1$ is the maximum number sent and $h_2$ the maximum number received we are taking $h = \max\{h_1, h_2\}$ here, although $h = h_1 + h_2$ is possible too [51]. Also how the expression for charging combines the $x$ with the $h$ is not crucial if, as we do here, we insist that each superstep is pure computation or pure communication. Also, we assume $g \geq 1$.] We note that any particular machine may impose a lower bound on $L$ for two reasons: There will be a minimum time requirement for doing synchronizations. Also the communication throughput specified by the chosen $g$ may be realisable only for large enough $L$ because of startup or latency considerations in routing. We note that although the model emphasizes global barrier style synchronization, pairs of processors are always free to synchronize pairwise by, for example, sending messages to and from an agreed memory location. These message transmissions, however, would have to respect the superstep rules.

## 5.2 Background

The BSP model was originally suggested as a possible "bridging" model to serve as a standard interface between the language and architecture levels in parallel computation. Two modes of programming were envisaged in [52]: *automatic mode* where programs are written in a high level language that hides memory distribution from the user (e.g. PRAM style), and *direct mode* where the programmer retains control of memory allocation.

This work is concerned with the direct mode. Since it is known that, under appropriate conditions, automatic mode achieves optimality to within constant factors, the primary question to ask is whether there are any circumstances at all in which the effort of programming in direct mode is warranted. We can identify four such circumstances:

(a) where small multiplicative constant factors in runtime are important,

(b) where smaller problem instances can be run efficiently in direct mode (i.e. less "slack" is sufficient) than in automatic mode,

(c) where the available machine has a high $g$ factor (since automatic mode generally requires $g$ to be close to unity), and

(d) where the available machine has an $L$ that is sufficiently high for direct, but not for automatic mode, for the problem instance in hand.

In light of the first of these conditions we emphasize multiplicative constant factors and in this work we restrict ourselves to *one-optimal* algorithms (i.e. where this factor is one asymptotically as the problem size $n \to \infty$). Although it is generally problematic to measure complexity to this accuracy, since the operations that are counted have to be carefully defined, it can be made meaningful, if, as we do here, we measure ratios between runtimes on pairs of models that have the same set of local instructions. Thus we shall say that a BSP algorithm is *one-optimal in computation* if it performs the same number of operations, to a multiplicative factor of $1 + o(1)$ as $n \to \infty$, as a specified corresponding sequential or PRAM algorithm. Insistence on one-optimality leads to algorithms that only work in restricted ranges of $n$ and

$L$ in terms of $p$. These ranges can be extended in most cases, however, if the requirements are relaxed to, say, "two-optimality".

We note that the role of a standard bridging model is as important in the direct programming context as in the automatic one. It will soon be inexpensive to augment workstations and personal computers with several processors. As long as efficient transportable software packages can be written for them, substantial reductions in runtime can be gained. We hypothesize that the BSP model, parametrized by $p$, $L$, and $g$ and with direct control of memory distribution among the processors, is a simple enough model to serve this bridging role. One can foresee software packages being written that can be run efficiently on a variety of machines for a wide range of values of these parameters. In these packages the algorithms would be parametrized by $p$, $L$, $g$, and the problem size $n$, in the same manner as we do in this work. An efficient instance of the algorithm can be selected by a combination of compile-time and run-time decisions based on these parameters.

The development and analysis of direct BSP algorithms has a secondary motivation, one relating to computer architecture. A currently unresolved question is that of determining whether, if BSP computers are constructed in currently foreseeable technologies, it is efficacious to augment the global operations supported in hardware beyond the router, to, for example, broadcasting or the parallel prefix. To resolve this the complexities of some of the most frequently computed problems with respect to these various options need to be understood.

## 5.3 Communication Optimal Simulations

In order to evaluate the usefulness of direct BSP algorithms we have to consider the best PRAM simulations with which they might compete. This competition is fairly stiff since, given sufficient slack, we can find simulations that are one-optimal in communication operations and optimal to within small multiplicative constant factors in computational operations. By a PRAM simulation being *one-optimal in communication* we mean in particular that $T(n)$ read/write operations of a PRAM can be simulated in $(2gT(n)/p)(1 + o(1))$ time units on the

BSP. The factor of $2g$ is necessary since each read operation is implemented as a message to a distant memory block followed by a return message, and time $g$ is charged per message (in terms of throughput, not necessarily latency). Asymptotics are in terms of the problem size $n$ as $n \to \infty$. The parameters $g$ and $p$ may or may not depend on $n$.

The results concern the simulation of CRCW and EREW PRAMs (which we call CPRAMs and EPRAMs for short) on either the BSP or the S*PRAM. The latter model is defined as follows: It consists of $p$ processor/memory components each of which can in each period of $g$ time units direct a message to any other component. Any component that has just one message directed at it receives it successfully and sends an acknowledgement to the sender in the same period. Any component that has more than one message directed at it fails to receive any of them and sends no acknowledgement. This model was suggested by Anderson and Miller [3] and also discussed by Hartmann and Redfield [29]. It models a variety of schemes for optical interconnects (eg [29], [41], [42]) that have been proposed for efficient all-to-all dynamic communication (although at present none of them is competitive with electronic methods).

Since the S*PRAM seems an elegant and potentially important model the question of simulating the BSP on it is of interest. In [51] it was shown how this could be done to within constant factors given logarithmic slack. Here we consider at what point one-optimality can be reached where the PRAMs are implemented directly on the S*PRAM (by-passing the BSP).

The time bounds hold with high probability assuming the availability of easily computed hash functions, that behave as perfect random functions when distributing addresses to memory blocks. We can use functions as in [15], or use pipelining to evaluate several applications at a time of a possibly less easily computed hash function.

Let $\log x$, $\lg x$ be the logarithms of $x$ to the base $e$ and two respectively.

**Theorem 20** *Let $w_p$ be any function of $p$ such that $w_p \to \infty$ as $p \to \infty$. Then the following amounts of slack are sufficient for simulating any one step of the EPRAM or CPRAM on the BSP or S*PRAM in one-optimal expected time for communication (and constant factor* *optimal time in local operations if $g = O(1)$):*

$$\begin{array}{ll} (i) & (w_p p \ \log p)\text{-}EPRAM \text{ on } p\text{-}BSP, \\ (ii) & (w_p p^2 \ \log p)\text{-}CPRAM \text{ on } p\text{-}BSP, \\ (iii) & (w_p p^2)\text{-}EPRAM \text{ on } p\text{-}S^*PRAM, \\ (iv) & (w_p p^2 \ \log p)\text{-}CPRAM \text{ on } p\text{-}S^*PRAM, \end{array}$$

*where the expressions before the hyphens denote the numbers of processors.*

**Proof:** In each of the four simulations we shall randomly choose an appropriate hash function $h$ that will randomize memory and distribute memory requests evenly among the $p$ memory modules of the *p-BSP* or $p$-$S^*PRAM$. Thus, the contents of memory address $m$ will be mapped by this hash function to a module $h(m)$, which we can assume from the previous discussion is random.

Each processor, at every step of a simulation will perform a number of tasks. For example, in case (i), each processor of the simulating *BSP* will perform $w_p \log p$ tasks of the simulated *EPRAM*, tasks that may need memory accesses distributed arbitrarily among the processors of the *BSP*.

We note that time $g$ is charged for each message delivery (when performed in bulk). When we discuss "time", in this context, as the number of such operations, this extra factor of $g$ is implied as far as actual time.

i) The simulation is the one given in [51]. After the distribution of $w_p \log p$ tasks to each processor of the $BSP$, as described above, these tasks are completed in one superstep. The duration of the superstep will be chosen large enough to accommodate the routing of an $(1+\epsilon)w_p \log p$-relation, thus allowing the choice of an $L$ as large as $(1+\epsilon)w_p \log p$, for any appropriate $\epsilon > 0$ that depends on the choice of $w_p$.

Under the perfect hash function assumption, all references to memory are mapped to memory modules independently of each other, and thus the probability that a memory location maps to some module (say that of processor $i$), is $1/p$. We then use the bound for the right tail of the binomial distribution given by Fact 7. The probability that the number of successes $X$ in $n$ independent Bernoulli trials, each with probability of success $P$, is greater than $(1+\epsilon)nP$, for some $\epsilon$ $(0<\epsilon<1)$ is at most

$$Prob(X > (1+\epsilon)nP) \leq e^{-\frac{1}{3}\epsilon^2 nP}.$$

Thus, the probability that among $w_p p \log p$ requests to memory, we have more than $(1+\epsilon)w_p \log p$ requests to a specific memory module is at most $p^{-w_p^{1/3}}$, for, say, $\epsilon = \sqrt{3}w_p^{-1/3}$. Since there are $p$ such modules, the probability that more than $(1+\epsilon)w_p \log p$ requests are directed to any of these, is at most $p^{1-w_p^{1/3}} = p^{-\Omega(w_p^{1/3})}$. It follows that reads or writes can be implemented as desired, in two or one supersteps respectively.

ii) Each processor of the BSP will simulate $w_p p \log p$ processors of the CPRAM. We shall show how, if one read request is made by each simulated processor, to an arbitrary other simulated processor, then they can be realised in two phases each of $(1+o(1))w_p p \log p$ steps of communication, and the same order, to within a constant factor of computation in the BSP. Write requests (with various conventions for concurrent writes) can be simulated in similar fashion, in just one phase.

Suppose we number the BSP processors $0, 1, \ldots, p-1$. The simulation consists of four parts.

a) Each BSP processor will sort the up to $w_p p \log p$ read requests it has according to the BSP processors numbers to which they are destined. This can be done in time linear in $w_p p \log p$ using bucket sort.

b) Each BSP processor hashes the actual addresses in the destination module of each request into a hash table to identify all such address (i.e. not module) collisions. This takes linear expected time. For each destination address one request is considered "unmarked" and the rest are "marked".

c) A round-robin algorithm is used to implement all the "unmarked" requests. In the $k$-th of the $p-1$ stages processor $j$ will transmit all its messages destined for memory module

$(j+k)(\text{mod} p)$, and receive back the values read.

d) The "marked" requests (which did not participate in (c)) are now given the values acquired by their "unmarked" representative in phase (c). This takes time linear in the number of requests per processor.

We can implement each of the $p-1$ stages of (c) in two supersteps, where $L = g(1+\epsilon)w_p \log p$, $\epsilon = w_p^{-1/3}$. This is because the probability that the up to $w_p p \log p$ read requests from a fixed processor will have more than $(1+\epsilon)w_p \log p$ addressed to any fixed module is then at most $\exp(-\Omega(w_p^{1/3} \log p))$ by a similar estimation to part (i). Since there are $p^2$ choices of the fixed processor and module, a bound of $p^2 \exp(-\Omega(w_p^{1/3} \log p)) = \exp(-\Omega(w_p^{1/3} \log p))$ follows. Finally we note that instead of doing the round robin in $2(p-1)$ supersteps we could do it in just two and allow $L$ as large as $L = g(1+\epsilon)w_p p \log p$.

iii) The simulation is a round robin process similar to (ii) but is asynchronous. The $w_p p^2$ EPRAM requests are distributed evenly among the $p$ $S^*PRAM$ processors, which bucket sort on the $S^*PRAM$ destination modules the $w_p p^2$ requests they are each given. Then the overall problem can be viewed as one of transmitting $p^2$ tasks $\{(j,k) \mid 0 \leq j,k < p\}$, where task $(j,k)$ is that of delivering the requests from processor $j$ destined for processor $k$.

Each processor $j$, after executing locally task $(j,j)$, will execute the tasks $(j, j+1(\text{mod}\ p))$, $(j, j+2(\text{mod} p)), \ldots, (j, j+p-1(\text{mod} p))$ in sequence. It will only execute task $(j, j+k(\text{mod} p))$ however, once it has received a "special" message from processor $j+1(\text{mod} p)$ stating that the latter processor has completed its task $(j+1(\bmod p), (j+1)+(k-1)(\bmod p))$ which is destined for the same processor. This ensures that there is never any contention at the destination processors.

We shall analyze the performance of this algorithm using the delay sequence argument introduced in [2], [48].

We construct a directed graph where each task $(j,k)$ is a node. There are directed edges from $(j,k)$ to $(j, k+1(\bmod p))$, and from $(j+1(\bmod p), k+1(\bmod p))$ to $(j, k+1(\bmod p))$ for each $j = 0, \ldots, p-1$, and $k = j, \ldots, j-2(\text{mod} p)$. Hence the nodes $(j,j)$ are the input nodes and nodes $(j, j-1(\text{mod} p))$ are the output nodes.

Consider one execution of this algorithm and suppose that the last task finished in $T$ units of time, where one unit is taken for each global message transmission. Suppose the task last finished was $(j_p, k_p)$. When this last task started it must have awaited the completion of either $(j_p, k_{p-1})$ or $(j_p + 1(\text{mod} p), k_p)$. Whichever task it was, we shall label $(j_{p-1}, k_{p-1})$. In this way we can trace back along a distinct path from an output task $(j_p, k_p)$ along a path $(j_i, k_i) \to (j_{i-1}, k_{i-1})$ to some input node $(j_1, k_1)$. This sequence of tasks we shall call a delay sequence $D$.

Let $t_j$ be the number of tasks $(j, *)$ in this delay sequence. Let $l_j$ be the total number of requests issued by these $t_j$ tasks. Then $l_j$ can be considered to be the number of successes in $pw_p$ Bernoulli trials with probability of individual success equal to $t_j/p$. Hence if $\Delta^D$ is the total number of requests in the tasks of delay sequence $D$, then

$$Prob(\Delta^D \geq T) = Prob(\sum_{j=0}^{p-1} l_j \geq T).$$

We can regard the $\Delta^D$ as the number of successes in $w_p p^2$ Bernoulli trials with various probabilities but with total expectation $w_p p$. Then the previously used bound on the tail of such distributions gives.

$$Prob(\Delta^D \geq (1+\epsilon) w_p p) \quad \leq \quad e^{-\epsilon^2 w_p p/3}.$$

if $0 < \epsilon < 1$. Since there are at most $p2^{p-1}$ possible delay sequences in the graph (i.e. $p$ destinations, and $2^{p-1}$ paths to each one) the probability that any one can cause a delay of $(1+\epsilon) w_p p$ is at most

$$p2^{p-1} e^{-\epsilon^2 w_p p/3}.$$

Choosing $\epsilon = w_p^{-1/3}$, for example, will make this probability approach zero as $p \to \infty$.

We note that in addition to the tasks in the delay sequence there may be up to $p-1$ additional "special" messages on the critical path. Hence the overall time is $(1+\epsilon) w_p p + p =$

$(1 + o(1))w_p p$.

(iv) The simulation follows that of part (ii), with the elimination of the now redundant step (b). ■

We note that for simulations such as ours, where the PRAM processors are shared evenly among the processors of the simulating machine, and the memory space is randomized by hashing, parts (i), (ii), and (iv) of the Theorem are optimal in the sense that the removal of $w_p$ in their statement would render them false. This follows from the fact (see [36], page 96, for a proof) that if, for any constant $c$, $cp \log p$ balls are thrown randomly into $p$ urns then with some constant probability there will be at least $c_1(c) \log p$ balls in at least one urn for some $c_1(c) > c$. For part (i) consider each memory request such a ball. For parts (ii) and (iv) consider the case that for each simulating processor the $cp \log p$ memory requests are to distinct memory locations, but the set of these locations are identical for the $p$ simulating processors. Then the $cp \log p$ requests can be viewed as balls thrown randomly into $p$ urns, but now $p$ copies of each arrive. (We note that randomizing the allocation of PRAM processors to the simulating processors does not help here.)

## 5.4 The Format

We shall specify the performance of a direct BSP algorithm $\mathcal{A}$ in three parts. First we specify the basic local operations allowed and an algorithm $\mathcal{A}^*$ (typically sequential or PRAM) with which we are comparing it. Second we specify two ratios $\pi$ and $\mu$. The former, $\pi$, is the ratio between the total operation counts of $\mathcal{A}$ and $\mathcal{A}^*$, and may be a function of $p, n$ and $L$. The latter, $\mu$, is the ratio between the parallel time required by the communication supersteps of $\mathcal{A}$, and the parallel computation time on the $p$ processors if $\pi$ were equal to one for the given $\mathcal{A}^*$. When specifying $\mu$ it is necessary to be explicit about the amount of information that can be conveyed in a single message. Finally we specify the conditions on $n, p$, and $L$ that are sufficient for the algorithm to make sense and the claimed bounds on $\pi$ and $\mu$ to hold. We state as corollaries some sufficient conditions for the most important optimality criteria, such as $\pi = 1 + o(1)$, $\mu = o(1)$, or $\mu \leq 1$. All asymptotic bounds refer to the problem size as $n \to \infty$. The other parameters may also $\to \infty$ if they are expressed explicitly in terms of $n$, but otherwise are not assumed to do so. Thus the results in general hold for values of $p, g$ and $L$ that may or may not depend on $n$.

**Fact 10** *Consider the arithmetic straight line model of computation and the standard algorithm $\mathcal{A}^*$ for multiplying an $n \times n$ matrix $M$ by an $n \times 1$ vector $x$ in $2n^2 - n$ steps. This can be implemented by algorithm $\mathcal{A}$ below on a BSP computer in the same arithmetic model with*

$$\pi \leq 1 + p/n, \text{ and } \mu \leq g(n+p)^2/(2n^2 - n),$$

*provided $L \leq n^2/p$ and $p \leq n$.*

**Corollary 1** *For algorithm $\mathcal{A}$ if $L \leq n^2/p$ and $p = o(n)$ then $\pi = 1 + o(1)$ and $\mu = g(\frac{1}{2} + o(1))$.*

**Proof:** We assume that the elements of both $M$ and $x$ are each initially distributed as evenly as possible among the $p$ processors. Algorithm $\mathcal{A}$ is the following: it divides the rows of $M$ into $p$ groups each of $\lfloor n/p \rfloor$ or $\lceil n/p \rceil$ rows. It routes all elements of the ith row group (up to $n\lceil n/p \rceil$ elements) to processor $i$. It transmits $p$ copies of each element of $x$, one copy to every processor. Finally each processor $i$ computes locally the values of the rows of $Mx$ that are in its group.

The actual computation, which is the last step, takes time $(2n-1)\lceil n/p \rceil = ((2n^2 - n)/p)(1 + p/n)$, which gives the claimed bound on $\pi$. The two routing operations take time at most

$$g(n\lceil n/p \rceil + p\lceil n/p \rceil) \leq g(n^2/p + n + n + p)$$

which gives the claimed bound on $\mu$. ■

## 5.5 Broadcasting and Parallel Prefix

Let a parallel computer (such as the BSP) have $p$ processors numbered $0 \leq i \leq p-1$, and let each processor $i$ have a distinguished register $r_i$ holding a value $x_i$. A *parallel prefix* *computation*, with an associative operator $*$ computes all values $x_0 * \ldots * x_i$, $i = 0, \ldots p-1$, so that value $x_0 * \ldots * x_i$ will be held by processor $i$ in the register $r_i$ originally holding $x_i$, at the completion of the parallel prefix computation.

By *broadcasting* we mean transmitting from any $r_i$ a copy of its contents to $r_j$ for every $j \neq i$. Broadcasting from $r_0$ can be simulated by the parallel prefix computation by defining $*$ such that for all $x, y$, $x * y = y$. For such an operation we call processor 0 the leader of the broadcast. A *parallel broadcast* operation is the transmission of elements from distinct registers to disjoint sets of registers.

A segmented parallel prefix computation (see also [40]) consists of simultaneous parallel prefix computations with the same associative operator $*$ but on disjoint subsequences of the registers $r_0, \ldots, r_{p-1}$.

BSP algorithms for broadcasting and parallel prefix were discussed in [52]. The broadcasting algorithm uses a $t$-ary tree, for some appropriate $t$. The initial message is at the root and in each superstep $t$ copies of each existing copy of the message is made and sent to $t$ distinct processors. Hence if there are $p$ processors to broadcast to, and $t = p^{1/k}$ then $k$ supersteps will suffice and the total communication time required will be

$$gk\max\{\tfrac{L}{g}, p^{1/k}\} \ .$$

The standard parallel prefix algorithm can be thought of also as using a tree but proceeds in two phases, first moving from the leaves to the root, and then back to the leaves, making some computations at each step. The computation time for a parallel prefix on a list of $p$ elements, one at each processor, will then be,

$$2k\max\{L, p^{1/k}\} \ ,$$

and the communication time

$$2gk\max\{\tfrac{L}{g}, p^{1/k}\} \ .$$

For either algorithm we can choose $k$ as large as $\lg p$.

## 5.6 Polynomial Evaluation

Suppose we are given a polynomial of degree $n$,

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 \quad , \quad a_n \neq 0$$

and we would like to evaluate $p(x)$ at the real point $x_0$. The application of Horner's rule provides a solution to this problem and gives an upper bound of $2n$ for the number of operations (additions and multiplications) required.

### 5.6.1 An Algorithm

We use here Dorn's scheme, as this is described in [11] to get a BSP algorithm for the polynomial evaluation problem. Our results are summarized in the following Theorem.

**Theorem 21** *On the arithmetic model of computation the evaluation of a polynomial of degree $n$ using Horner's rule at some real point $x_0$, requires $2n$ arithmetic operations. BSP algorithm PE1 below, for $p < n$ yields*

$$\pi \leq 1 + \frac{p(p-1)}{n} + \frac{p \lg p}{n} + \frac{Lp}{n} + \frac{p}{n},$$

*and*

$$\mu = g\max\{\tfrac{L \cdot p}{2 \cdot g \cdot n}, \tfrac{p(p-1)}{2n}\}.$$

**Corollary 2** *If in algorithm PE1, we choose $L = o(n/p)$ and $p^2 = o(n)$, then $\pi = 1 + o(1)$, and if in addition $g = o(n/(p(p-1)))$, then $\mu = o(1)$.*

**Proof:** For $i = 0, \ldots, p-1$ let $l = \lfloor \frac{n-i}{p} \rfloor$, and $q_i(x) = \sum_{j=0}^{l} a_{i+pj} x^{pj}$. Then,

$$p(x) = \sum_{i=0}^{p-1} q_i(x) x^i. \tag{5.1}$$

Algorithm PE1 is described below. Each of the $p$ processors computes separately one $q_i(x_0)$ by precomputing $x_0^p$. Subsequently, all the partial results are recombined by Equation

(5.1), which expresses $p(x)$ in terms of $q_i(x)$, and $p(x_0)$ is thus computed. We assume that the coefficients of $p(x)$ are local to the processors that will use them. Otherwise, the routing of a $\lceil n/p \rceil$-relation is required for delivering these coefficients to the processors that will need them.

**Procedure** *PE1*

**begin**

**Stage 1:** Each processor $i = 1, \ldots, p$ evaluates $q_{i-1}(x_0)$ sequentially by first evaluating $x_0^p$, and then using Horner's rule. After the computation is completed, the processors send all partial results to processor 1.

[ **Analysis of Stage 1:** This stage consists of two supersteps, one for computation, and one for communication. Each processor first computes $x_0^p$ in $2\lceil \lg p \rceil$ operations, and then evaluates $q_i(x_0)$ in $2\lfloor \frac{n-i}{p} \rfloor$ operations. This latter time is at most $2\lfloor n/p \rfloor$. For the communication part, we only need to be able to route a $(p-1)$-relation. ]

**Stage 2:** Processor 1, after it receives all partial results, uses Equation (5.1) to evaluate $p(x_0)$ by applying Horner's rule again.

[ **Analysis of Stage 2:** This stage requires $2(p-1)$ arithmetic operations and is executed in a single superstep. ]

**end**

From the analysis of algorithm PE1, we get that the total computation time for both stages is given by the following expression.

$$
\begin{aligned}
C(n) &= \max\{L, 2\lceil \lg p \rceil + 2\lfloor n/p \rfloor\} \; + \max\{L, 2(p-1)\} \\
&\leq (L + 2\lceil \lg p \rceil + 2\lfloor n/p \rfloor) + (L + 2(p-1)) \\
&\leq 2 \lg p + 2 + 2L + 2n/p + 2(p-1) \\
&= \frac{2n}{p}\left(1 + \frac{p(p-1)}{n} + \frac{p \lg p}{n} + \frac{Lp}{n} + \frac{p}{n}\right).
\end{aligned}
$$

The communication time is given by $g\max\{\frac{L}{g},(p-1)\}$ , thus giving

$$M(n) = \frac{2n}{p} g\max\{\tfrac{L\cdot p}{2\cdot g\cdot n}, \tfrac{p(p-1)}{2n}\}.$$

We need only choose $L = o(n/p)$, $p^2 = o(n)$, in order to have $\pi = 1 + o(1)$ and if in addition $g = o(n/p^2)$ we get $\mu = o(1)$ also.

In case the coefficients are not available locally at each processor, we need to spend $g\lceil n/p\rceil$ additional communication time for this routing thus yielding $g = O(1)$. ■

**Example 1** *Let $n = 20000$, and $p = 16$. Then the computation time is within 10% of the optimal for $L \leq 100$, and $\mu \leq 1$, if, in addition, $g \leq 80$.*

We can increase the upper bound for $p$ (from sub-$\sqrt{n}$ to sublinear) by using the variant described in the next section.

### 5.6.2 A Variant of the Algorithm

We present below a variant of PE1 that improves its slack.

**Theorem 22** *On the arithmetic model of computation the evaluation of a polynomial of degree $n$ using Horner's rule at some point $x_0$, requires $2n$ arithmetic operations. If $k$ is any positive integer such that $p^{1/k}$ is an integer and $p < n/k$, then algorithm PE2 gives*

$$\pi \leq 1 + \frac{kp^{1+1/k}}{n} + \frac{2p\lg p}{n} + \frac{L(k+1)p}{n} + \frac{(k+1)p}{n},$$

*and*

$$\mu \leq gk\max\{\tfrac{L\cdot p}{2\cdot g\cdot n}, \tfrac{p^{1+1/k}}{2n}\}.$$

**Corollary 3** *If in algorithm PE2 $L = o(n/(kp))$, $g = o(n/(kp^{1+1/k}))$, and $kp^{1+1/k} = o(n)$, then $\pi = 1 + o(1)$ and $\mu = o(1)$.*

**Proof:** Let us fix an integer number $k$ so that $m = p^{1/k}$ is an integer. In algorithm PE1, in the communication superstep of stage 1, we routed a $p$-relation in a single superstep. We then, in stage 2, combined these $p$ results into one by using Equation (5.1).

Here we replace the communication superstep of stage 1 and the single superstep of stage 2 of PE1 by $k$ new stages numbered $2, \ldots, k+1$. In each such stage $p^{1/k}$ partial results will be combined into one. After $k$ stages, that is, at the conclusion of stage $k+1$ we shall have the value of $p(x_0)$ previously obtained in stage 2 of PE1.

Polynomial $p(x)$ of degree $n$ was expressed by Equation (5.1) in terms of $p$ polynomials with coefficients those of $p(x)$. Let $q_i(x)$, $i = 0, \ldots, p-1$, be these polynomials as they were defined in the proof of Theorem 21.

We observe that it is possible, in a similar way, to express $p(x)$ in terms of $p/m = p^{1-1/k}$ polynomials as follows.

$$p(x) = \sum_{j=0}^{p/m-1} r_j(x) \, (x^m)^j \,, \tag{5.2}$$

where

$$r_j(x) = q_{jm}(x) + q_{jm+1}(x)x + \ldots + q_{(j+1)m-1}(x)x^{m-1}. \tag{5.3}$$

This way, instead of evaluating $p(x_0)$ by evaluating a polynomial of degree $p-1$ at point $x_0$ whose $p$ coefficients are $q_i(x_0)$, we can evaluate $p(x_0)$ by first computing $x_0^m$ and then evaluating at $x_0^m$ a polynomial of degree $p/m = p^{1-1/k} - 1$ whose coefficients are then $r_j(x_0)$. All $r_j(x_0)$, whose computation involve the evaluation of a polynomial of degree $m-1$ at $x_0$, and $x_0^m$ are computed in stage 2 of PE2. Although both $r_j(x_0)$ and $x_0^m$ are computed at this stage the evaluation of $p(x_0)$ is delayed.

This way we can similarly evaluate $p(x_0)$ by first computing $x_0^{m^2}$ and then evaluating at $x_0^{m^2}$ a polynomial of degree $p/m^2 - 1 = p^{1-2/k} - 1$, whose coefficients can be found by evaluating polynomials of degree $m-1$ at $x_0^m$ whose coefficients are $r_j(x_0)$. These coefficients and $x_0^{m^2}$ are computed in stage 3 of PE2 but the evaluation of $p(x_0)$ is further delayed. Thus similarly generalizing, at stage $k+1$ we can evaluate $p(x_0)$ by evaluating a polynomial of degree 0, its own value. Then at the conclusion of this stage the value of $p(x_0)$ can thus be found.

This way, we only need to route during a superstep a $p^{1/k}$-relation not a $p$-relation. This allows us to achieve a better slack than was possible with PE1.

In each of the $k$ new stages, we need to route a $p^{1/k}$-relation and thus spend $g\max\{\frac{L}{g}, p^{1/k}\}$ time in communication for each stage. Parallel time at most $\max\{L, 2p^{1/k} + 2\lceil \lg p^{1/k} \rceil\}$ is required to evaluate the intermediate coefficients by evaluating a polynomial of degree $m - 1 = p^{1/k} - 1$ such as, $r_j(x)$, in each processor and in each stage, and $2\lceil \lg p^{1/k} \rceil$ to precompute in each stage a power of the form $y^{p^{1/k}} = y^m$ such as, $x_0^m$ in stage 2.

Thus, each of the $k$ stages $2, \ldots, k+1$ consists of two supersteps, one for computation, and one for communication. Stage 1, consists of the computation superstep of PE1.

Thus the computation time of algorithm PE2, described above, is given by the following expression.

$$\begin{aligned}
C(n) &\leq \max\{L, 2\lceil \lg p \rceil + 2\lfloor n/p \rfloor\} \ + k\max\{L, 2p^{1/k} + 2\lceil \lg p^{1/k} \rceil\} \\
&\leq (L + 2\lceil \lg p \rceil + 2\lfloor n/p \rfloor) + k\Big(L + 2p^{1/k} + 2(1 + \tfrac{1}{k}\lg p)\Big) \\
&\leq 4\lg p + 2 + 2k + (k+1)L + 2n/p + 2kp^{1/k} \\
&= \frac{2n}{p}\left(1 + \frac{kp^{1+1/k}}{n} + \frac{2p\lg p}{n} + \frac{L(k+1)p}{n} + \frac{(k+1)p}{n}\right).
\end{aligned}$$

If $L = o(n/(pk))$, and $kp^{1+1/k} = o(n)$, then $\pi = 1 + o(1)$.
The communication time is given by the following expression.

$$\begin{aligned}
M(n) &\leq kg\max\{\tfrac{L}{g}, p^{1/k}\} \\
&\leq \frac{2n}{p} gk\max\{\tfrac{L\cdot p}{2\cdot g\cdot n}, \tfrac{p^{1+1/k}}{2n}\}.
\end{aligned}$$

If in addition to the previous restrictions on $L$ and $p$, we also have $g = o(n/(kp^{1+1/k}))$, we get $\mu = o(1)$. ■

**Example 2** *If* $n = 20000$*,* $p = 16$ *(*$k = 4$*), then the computation time is within* 10% *of the optimal for* $L \leq 21$*, and* $\mu \leq 1$ *if, in addition,* $g \leq 300$*.*

In order to get the best possible slack (the smallest ratio $n/p$ possible for optimal algorithms), we can choose $k = \lg p$.

These results are summarized in the following Corollary.

**Corollary 4** *Algorithm PE2 with $k = \lg p/\lg\lg p$ achieves,*

$$\pi = 1 + \frac{2p\lg p}{n} + \frac{2p\lg p}{n} + \frac{Lp}{n}(\lg p + 1) + \frac{p\lg p}{n} + \frac{p}{n},$$

*and*

$$\mu = g\lg p \max\{\tfrac{L\cdot p}{2\cdot g\cdot n}, \tfrac{p}{n}\}.$$

*If $L = o(\frac{n}{p\lg p})$, and $p\lg p = o(n)$, then $\pi = 1 + o(1)$, and if in addition, $g = o(\frac{n}{p\lg p})$, $\mu = o(1)$.*

## 5.7 Gauss-Jordan Elimination.

The Gauss-Jordan elimination algorithm can be used to solve a single system of linear equations, a multiple system of linear equations on the same coefficient matrix, or to invert a non-singular matrix $A$.

When applied to an $n \times n$ matrix $A$ it consists of $n$ iterations, where at iteration $i$, $1 \leq i \leq n$, it eliminates all non-diagonal elements of column $i$ of $A$. In particular suppose that at the end of iteration $i-1$, the elements of matrix $A$ are $a_{jl}^{(i-1)}$. Then in the $i$-th iteration the algorithm leaves row $i$ unchanged but changes each other row $j \neq i$ as follows:

$$a_{jl}^{(i)} = a_{jl}^{(i-1)} - \frac{a_{ji}^{(i-1)}}{a_{ii}^{(i-1)}} a_{il}^{(i-1)} \quad \forall l = i, \ \ldots, n. \tag{5.4}$$

After iteration $i$, the non-diagonal elements of the first $i$ columns are all zero.

The version we have described does no pivoting and performs $n^3 + O(n^2)$ arithmetic operations. If it is used to solve a system of linear equations $Ax = b$, the row operations specified above for $A$ are also performed on $b$, but this adds only an $O(n^2)$ term.

Below we show that even if pivoting is added the BSP can execute this algorithm one-optimally within a suitable range of parameters. The algorithm uses a scattered distribution of the data among the processors to even out processor utilization (c.f. [18]). For simplicity we shall assume here that $\sqrt{p}$ and $n/\sqrt{p}$ are both integers and that $p$ is a power of two.

We are not claiming that the given algorithm yields the best lower order terms, or indeed, that our analysis of the given algorithm yields the best possible bounds for these terms. Our objective is to show that a relatively simple algorithm has a relatively simple expression upper bounding its performance in terms of the parameters $n$, $p$, $g$, and $L$. Hence if such an algorithm is implemented in a transportable program package, then it can be run on any BSP machine efficiently, if the parameters $p$, $g$ and $L$ of the machine, and the input size $n$ are taken into account during compilation and runtime. Of course such a transportable algorithm may have general constraints on the parameters ($p \leq n^2$ in the case below). If the constraints are not satisfied the algorithm may not be well defined or the analysis applicable.

The arithmetic model of computation to be claimed in the theorems to follow refers to the arithmetic straight line model with arithmetic operations $\{+, -, *, /\}$. This model does not accommodate the variant of the Gauss-Jordan algorithm when pivoting is used. For this we need to extend to model so that it includes operations like $a := op(b, c)$, where $a, b, c$ are variables (indeterminates) and $op$ is a binary operator such as $=$ or $>$, or an arithmetic operator such as max which returns the maximum of $b, c$. The model should also accommodate **if** *cond* **then** *stat*1 **else** *stat*2 statements. If the value of the variable *cond* is non-zero statement *stat*1 is executed otherwise *stat*2 is executed. As far as communication is concerned a message consists of some header information, which is not taken directly (or taken indirectly through the values of $L$ and $g$) into account in the analysis of the various algorithms plus the body of the message, the piece of information being transmitted that is related to the problem under consideration. In one instance we transmit as a single message a pair of variables. We do this on the understanding that on a single message an array element along with its position within the array are transmitted in a single message and we charge one message for such a transmission.

**Theorem 23** *On the arithmetic model of computation, Gauss-Jordan elimination (with or without pivoting), for the solution of a single system of equations, uses* $n^3 + O(n^2)$ *operations.*

*There is a BSP algorithm for the same problem (with partial row pivoting) that for $p \leq n^2$, and any integer $k$ ($1 \leq k \leq \lg p$) has:*

$$\pi \leq 1 + 2\max\{\tfrac{Lp}{n^2}, \tfrac{\sqrt{p}}{n}\} \; + k\max\{\tfrac{Lp}{n^2}, \tfrac{p^{1+1/(2k)}}{n^2}\} \; + \frac{4\sqrt{p}}{n} + \frac{L^2p^2}{2n^4},$$

*and*

$$\mu = g\left(4k\max\{\tfrac{Lp}{gn^2}, \tfrac{p^{1+1/(2k)}}{n^2}\} \; + \max\{\tfrac{Lp}{gn^2}, \tfrac{\sqrt{p}}{n}\} \; + \lg\sqrt{p}\max\{\tfrac{Lp}{gn^2}, \tfrac{2\sqrt{p}}{n}\} \right).$$

**Proof:** We first split the processors into $\sqrt{p}$ groups $P_r$, $1 \leq r \leq \sqrt{p}$, of size $\sqrt{p}$ each. Let the processors of group $P_r$ be numbered

$$P_r = \{(r-1)\sqrt{p} + m \;\mid\; m = 1, \ldots, \sqrt{p}\}.$$

We will sometimes call processor $(r-1)\sqrt{p} + m$ the $m$-th processor of group $P_r$.

We assign the elements of column $i$ of $A$ to the group $P_{(i-1)(\mathrm{mod}\sqrt{p})+1}$ of processors, in such a way that the the $j$-th element of column $i$ will be assigned to the $[(j-1)(\mathrm{mod}\sqrt{p}) + 1]$-st processor of that group. We handle column $b$ as yet another column of $A$. Each processor therefore gets at most $\frac{n}{\sqrt{p}}\left(\frac{n}{\sqrt{p}} + 1\right)$ elements of $A$ or $b$.

We describe the algorithm in terms of the operations performed in a succession of stages. We are going to use the following terminology. Let *columnp*$(i)$ be the set of $\sqrt{p}$ processors holding elements of column $i$. Similarly, let *rowp*$(i)$ be the set of $\sqrt{p}$ processors that hold the elements of row $i$ of $A$ (and $b$). Let *proc*$(a_{ij})$ be the processor that holds element $a_{ij}$.

The algorithm consists of $n$ iterations, where, for $1 \leq i \leq n$, the $i$-th iteration executes calls of three procedures, *Pivot*$(i)$, *Communicate*$(i)$, and *Compute*$(i)$ in succession. These procedures are described below.

**Procedure** *Pivot*$(i)$

**begin**

**Stage 1:**

Every processor, among the ones in *columnp*$(i)$, finds the element with maximum absolute value among the elements that are simultaneously in column $i$ and row $j$, for some $j \geq i$, that are local to it. It then communicates this maximum and its row number to processor $proc(a_{ii})$.

[ **Analysis of Stage 1:** Two phases are required for this stage, one for computation and one for communication. The computational part requires time $n/\sqrt{p}$, while the communication part requires the routing of a $(\sqrt{p}-1)$-relation, if we regard an (element,row number) pair as one message. ]

**Stage 2:**

(i) Processor $proc(a_{ii})$ finds the maximum among the local maxima it received in stage 1, and identifies the original row number of this maximum as, say, $l$.

(ii) Processor $proc(a_{ii})$ communicates the value of $l$ to processors $rowp(i)$ and $rowp(l)$.

[ **Analysis of Stage 2:** The single computation phase requires time $\sqrt{p}$ while the communication requires the routing of a $2\sqrt{p}$ relation. ]

**Stage 3:**

For each column $k$, the processors of $rowp(i)$ and $rowp(l)$ containing elements from that column exchange the elements of rows $i$ and $l$ they possess.

[ **Analysis of Stage 3:** An $n/\sqrt{p}$-relation needs to be routed in a single superstep here. ]

**end**

**Procedure** *Communicate*$(i)$

**begin**

**Stage 1:**

Processor $proc(a_{ii})$ broadcasts element $a_{ii}$ to all $\sqrt{p}$ processors of $rowp(i)$.

[ **Analysis of Stage 1:** The routing of a $\sqrt{p}$-relation is required. ]

**Stage 2:**

Every processor of $rowp(i)$ divides the elements of row $i$ that it holds by $a_{ii}$.

[ **Analysis of Stage 2:** Each element of row $i$ must first be divided by $a_{ii}$ before it is broadcast. This takes $n/\sqrt{p}$ time. ]

**Stages** $3$ **,** $\ldots$**,** $\lceil \lg \sqrt{p} \rceil + 2$**:**

In $\lceil \lg \sqrt{p} \rceil$ stages, every processor $j \in rowp(i)$ broadcasts the element of row $i$ it computed in the previous stage to all the processors of the processor group $P_m$ it belongs to (i.e. $j \in P_m$). Simultaneously, the $j$-th processor of *columnp*$(i)$ broadcasts the elements of column $i$ it owns to the $j$-th processors of every other group. Since we need to broadcast an element to at most $\sqrt{p}$ processors, $\lceil \lg \sqrt{p} \rceil$ supersteps of binary replication suffice.

[ **Analysis of Stage** $3$ **,** $\ldots$**,** $\lceil \lg \sqrt{p} \rceil + 2$**:** In each of these $\lceil \lg \sqrt{p} \rceil$ supersteps a $2n/\sqrt{p}$-relation needs to be routed. ]

**end**

**Procedure** *Compute*$(i)$

**begin**

**Stage 1:**

Every processor performs the elimination step given by Equation (5.4) on the elements it owns (note that the ratio $a_{kl}/a_{kk}$, as given by Equation (5.4), need not be computed separately by each processor, since it has been evaluated in procedure $Communicate(i)$).

[ **Analysis of Stage 1:** The elimination of column $i$ requires $T_i = 2\frac{n}{\sqrt{p}}\lceil \frac{n-i}{\sqrt{p}} \rceil + \frac{2n}{\sqrt{p}}$ operations. The actual time to be devoted, however, to it will be $L$, if $L > T_i$. ]

**end**

Several times in the algorithm we need to broadcast items to $\sqrt{p}$ processors each. If $p$ is large, then it is best to do this in $k$ supersteps each of which does a $p^{1/(2k)}$-relation rather than in one superstep by doing a $\sqrt{p}$-relation. If we follow this philosophy also when we find the maximum in the pivot procedure, we will need to find maxima of sets of size $p^{1/(2k)}$ in each of $k$ successive stages.

We first find the total amount of time spent on computation for the elimination of all columns (pivoting operations included). We can easily get, from our previous discussion, that this time is upper bounded by

$$C(n) \quad \leq \quad 2n\max\{L, \tfrac{n}{\sqrt{p}}\} \; + nk\max\{L, p^{1/(2k)}\} \; + \sum_{i=1}^{n} \max\{L, T_i\} \; .$$

The cost of eliminating column $i$ is less than $L$, for

$$2\frac{n}{\sqrt{p}}\lceil \frac{n-i}{\sqrt{p}} \rceil + \frac{2n}{\sqrt{p}} < L.$$

This is true provided that the following condition holds

$$\frac{2n}{\sqrt{p}}\left(\frac{n-i}{\sqrt{p}} + 2\right) \quad < \quad L,$$

or, equivalently,

$$(n-i) \quad < \quad \frac{Lp}{2n} - 2\sqrt{p}.$$

Thus, for each of the last $\frac{Lp}{2n} - 2\sqrt{p}$ calls to procedure $Compute(i)$ we charge $L$ time steps.

Overall, we get

$$\begin{aligned} \sum_{i=1}^{n} \max\{L, T_i\} \quad &\leq \quad \sum_{i=1}^{n} T_i + \sum_{i=n-\frac{Lp}{2n}+2\sqrt{p}+1}^{n} L \\ &\leq \quad \left(\frac{n^3}{p} + \frac{4n^2}{\sqrt{p}} - \frac{n^2}{p}\right) + \left(\frac{L^2 p}{2n} - 2\sqrt{p}L\right). \end{aligned}$$

Then, the expression for $C(n)$ can be written in the following form.

$$\begin{aligned} C(n) \quad &\leq \quad \frac{n^3}{p}\left(2\max\{\tfrac{Lp}{n^2}, \tfrac{\sqrt{p}}{n}\} \; + k\max\{\tfrac{Lp}{n^2}, \tfrac{p^{1+1/(2k)}}{n^2}\} \; + 1 + \frac{4\sqrt{p}}{n} + \frac{L^2 p^2}{2n^4}\right) \\ &= \quad \pi\frac{n^3}{p}. \end{aligned}$$

Similarly, the total communication time can be bounded as follows.

$$
\begin{aligned}
M(n) &\leq ng\left(4k\max\{\tfrac{L}{g}, p^{1/(2k)}\} + \max\{\tfrac{L}{g}, \tfrac{n}{\sqrt{p}}\} + \lg\sqrt{p}\max\{\tfrac{L}{g}, \tfrac{2n}{\sqrt{p}}\}\right) \\
&\leq g\frac{n^3}{p}\left(4k\max\{\tfrac{Lp}{gn^2}, \tfrac{p^{1+1/(2k)}}{n^2}\} + \max\{\tfrac{Lp}{gn^2}, \tfrac{\sqrt{p}}{n}\} + \lg\sqrt{p}\max\{\tfrac{Lp}{gn^2}, \tfrac{2\sqrt{p}}{n}\}\right) \\
&= \mu\frac{n^3}{p}.
\end{aligned}
$$

■

Several corollaries can be deduced from the result as stated. As we expect, if we have substantial or even moderate slack, then the algorithm is efficient. The case $k = 1$, for example, gives:

**Corollary 5** *If $p = 100$, $n = 1000$, $L = 100$, then $\pi$ is about 7% from optimal, and $\mu \leq 1$ for $g$ up to about 15.*

It can be seen that for $k = 1$, $\pi = 1 + o(1)$ and $\mu = o(1)$ can be achieved under suitable conditions on $L$ and $g$ if $p = o(n^{4/3})$. Beyond that higher values of $k$ are needed. The case $k = \lg p$ yields:

**Corollary 6** *If $p \log p = o(n^2)$ and $L = o(\frac{n^2}{p \lg p})$, then $\pi = 1 + o(1)$. If in addition $g = o(\frac{n}{\sqrt{p}\lg p})$, then $\mu = o(1)$ also.*

These results show that even in the absence of combining or parallel prefix operations, Gauss-Jordan elimination can be solved efficiently even when $L$ is about as large as it can be, and communication is moderately slow.

## 5.8 Cholesky Factorization

Let $A = [a_{ij}]$ be an $n \times n$ positive definite (symmetric) matrix. $A$ then accepts a unique factorization $A = L \cdot L^t$ into the product of two triangular matrices, where $L$ is a lower triangular matrix, with positive diagonal elements. Results from linear algebra show that the diagonal elements of $A$ are also positive. Let us write $A$ into the following form,

$$A = \begin{bmatrix} a & u^t \\ u & B \end{bmatrix},$$

where $a$ is element $a_{11}$ of $A$, $u$ is the $(n-1) \times 1$ vector consisting of the elements in the remaining $n-1$ rows of column one of $A$, and $B$ is an $(n-1) \times (n-1)$ matrix derived by deleting the first row and column of $A$. In effect we are seeking to find a decomposition of $A$ into the product of two triangular matrices. Let us then write $A$ in the following form,

$$\begin{aligned} A &= \begin{bmatrix} a & u^t \\ u & B \end{bmatrix} \\ &= \begin{bmatrix} \beta & 0 \\ w & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & C \end{bmatrix} \cdot \begin{bmatrix} \beta & w^t \\ 0 & I_{n-1} \end{bmatrix} \\ &= L_1 \cdot A_1 \cdot L_1^t \end{aligned}$$

It is straightforward that $\beta = \sqrt{a}$, $w = u/\sqrt{a}$, and $C = B - w \cdot w^t$. We continue this decomposition by decomposing $A_1$ (which is also a symmetric positive definite matrix) into $L_2 \cdot A_2 \cdot L_2^t$, and continue similarly so that at the end we get $A_n = I_n$, and $L = L_1 \cdot L_2 \cdot \ldots \cdot L_n$.

The following pseudoprogram describes this factorization and is based on the one found in [27]. It decomposes $A$ into $L$ and $L^t$ and stores $L$ in place of $A$. Thus elements $a_{ij}^{(k)}$ and $a_{ij}^{(k-1)}$ occupy, at different iterations in the execution of the algorithm ($k$ and $k-1$ respectively for these two elements), the memory location originally assigned to element $a_{ij}$ of $A$. Later in this section we shall refer to the location $a_{ij}^{(k)}$ is stored, by that of $a_{ij}$, or call it in short $(i, j)$.

Program Description for Cholesky Factorization.

**begin**

1. Let $A^{(0)} = A$.
2. for $k = 1, \ldots, n$ do

**begin**

3. for all $i, j$ such that $i < k$ or $j < k$, $a_{ij}^{(k)} = a_{ij}^{(k-1)}$

4. $a_{kk}^{(k)} = \sqrt{a_{kk}^{(k-1)}}$.

5. for $i = k+1, \ldots, n$ do

**begin**

6. $a_{ik}^{(k)} = a_{ik}^{(k-1)} / a_{kk}^{(k)}$

7. $a_{ki}^{(k)} = a_{ki}^{(k-1)} / a_{kk}^{(k)}$

**end**

8. for $j = k+1, \ldots, n$ do

9. for $i = j, \ldots, n$ do

**begin**

10. $a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k)} \cdot a_{kj}^{(k)}$.

**end**

**end**

**end**

The BSP algorithm for Cholesky Factorization is similar to the algorithm for Gauss Jordan elimination of [26]. When applied to an $n \times n$ positive definite matrix $A$ it consists of $n$ iterations. Each such iteration corresponds to the "elimination" of a column of $A$ say, $l$, and the formation of a corresponding column of $L_l$ which will be stored in the memory locations occupied by the elements of column $l$ of $A$. A known fact (see [27]) is that the $l$-th column of $A^{(l)}$, formed after the completion of the loop of step 5, for $k = l$, gives the $l$-th column of $L$.

The algorithm uses a scattered distribution of the data among the processors to even out processor utilization (c.f. [18]). For simplicity we shall assume here that $\sqrt{p}$ and $n/\sqrt{p}$ are both integers and that $p$ is a power of two.

**Theorem 24** *On the arithmetic model of computation with operations* $\{+, -, *, /, \sqrt{}\}$, *Cholesky factorization requires* $n^3/3 + O(n^2)$ *operations. There is a BSP algorithm for this problem* *that for $p \leq n^2$ and any integer $l$ $(1 \leq l \leq \lg p)$ has:*

$$\pi \leq 1 + \frac{3 \cdot L \cdot p}{n^2} + \frac{3 \cdot p}{n^2} + 3 \cdot \max\{\tfrac{Lp}{n^2}, \tfrac{p}{n^2}\} \ + 3 \cdot \max\{\tfrac{Lp}{n^2}, \tfrac{\sqrt{p}}{n} + \tfrac{p}{n^2}\} \ + \frac{3 \cdot \sqrt{p}}{n},$$

*and*

$$\mu \leq g \cdot \left(3l\max\{\tfrac{Lp}{gn^2}, \tfrac{p^{1+1/(2l)}}{n^2}\} \ + 3 \cdot \lceil \lg \sqrt{p} \rceil \cdot \max\{\tfrac{Lp}{gn^2}, \tfrac{2\sqrt{p}}{n} + \tfrac{2p}{n^2}\} \right).$$

**Proof:** We first split the processors into $\sqrt{p}$ groups $P_r$, $1 \leq r \leq \sqrt{p}$, of size $\sqrt{p}$ each. Let the processors of group $P_r$ be numbered

$$P_r = \{(r-1)\sqrt{p} + m \ \mid \ m = 1, \ldots, \sqrt{p}\}.$$

We will sometimes call processor $(r-1)\sqrt{p} + m$ the $m$-th processor of group $P_r$.

We assign the elements of column $i$ of $A$ to the group $P_{(i-1)(\bmod \sqrt{p})+1}$ of processors, in such a way that the the $j$-th element of column $i$ will be assigned to the $[(j-1)(\bmod \sqrt{p}) + 1]$-st processor of that group. We handle column $b$ as yet another column of $A$. Each processor therefore gets at most $\frac{n}{\sqrt{p}}\left(\frac{n}{\sqrt{p}} + 1\right)$ elements of $A$ or $b$.

We describe the algorithm in terms of the operations performed in a succession of stages. We are going to use the following terminology. Let *columnp*$(i)$ be the set of $\sqrt{p}$ processors holding elements of column $i$. Similarly, let *rowp*$(i)$ be the set of $\sqrt{p}$ processors that hold the elements of row $i$ of $A$ (and $b$). Let *proc*$(a_{ij})$ be the processor that holds element $a_{ij}$ or whatever element is stored there in subsequent iterations of the program description given earlier.

**Procedure** *CHOLESKY*

**begin**

**for** $k = 1$ **to** $n$ **do**

**begin**

**Stage 1:**

(i) The computation described by step 4 of the program description is performed. Processor $proc(a_{kk})$ updates $a_{kk}$ to be the square root of its previous value.

(ii) Processor $proc(a_{kk})$ broadcasts the new $a_{kk}$ to the $\sqrt{p}$ processors of $columnp(k)$.

[ **Analysis of Stage 1:** Stage 1 is further divided into two stages, one for computation, and one for communication. The computation stage is a single superstep, which involves the execution of a single square root operation. Steps 1 through 3 of the program description are implied by the way matrix elements are stored.

The communication stage requires the broadcasting of an element to $\sqrt{p}$ processors. Several times we need to broadcast items to $\sqrt{p}$ processors each. If $p$ is large, then it is best to do this in $l$ supersteps each of which does a $p^{1/(2l)}$-relation rather than in one superstep by doing a $\sqrt{p}$-relation. ]

**Stage 2:** The $columnp(k)$ processors execute step 6 of the program description, limited only to column $k$.

[ **Analysis of Stage 2:** Since $A$ (and $A_1, \ldots, A_n$ for that matter, as these were defined in the discussion prior to the program description) is a symmetric matrix, the work performed by step 7 of the program description duplicates that of step 6. It is thus redundant and can be omitted. This stage thus consists of a single computation superstep, and $\lceil (n-k)/\sqrt{p} \rceil \leq n/\sqrt{p} + 1$ arithmetic operations per processor are performed. ]

**Stage 3:** In iteration $k$, after the completion of stage 2 of this iteration, some processors hold elements of the $k$-th column of $A^{(k-1)}$ and thus of $A^{(k)}$ and $L$. Let $r$ be a processor among them that holds an element of some row $i$, where $i \in \{k+1, \ldots, n\}$, of this column $k$. Then processor $r$ broadcasts this element to the processors that hold elements of row $i$ on columns $k+1, \ldots, n$.

[ **Analysis of Stage 3:** This stage prepares the update steps 8, 9 and 10 of the program description, which are limited to an $(n-k) \times (n-k)$ lower right submatrix of $A^{(k)}$. Only the elements of rows $k+1, \ldots, n$ of column $k$ need to be broadcast, since this column $k$ as a column of $L$ has its elements of rows $1, \ldots, k-1$ all 0, while the element in row $k$ is not required for the update steps 8, 9, and 10.

In a single communication superstage the broadcasting of at most $2 \cdot \lceil \frac{n-k}{\sqrt{p}} \rceil$ elements held by a single processor to at most $\sqrt{p}$ processors is required, since an element in position $a_{ij}$ must get both $a_{ik}$ and $a_{kj} = a_{jk}$ for step 10 of the program description to complete. Since we need to broadcast an element to at most $\sqrt{p}$ processors, $\lceil \lg \sqrt{p} \rceil$ supersteps of binary replication suffice. Thus a broadcasting in $\lceil \lg \sqrt{p} \rceil$ supersteps, by routing a $2 \cdot \lceil \frac{n-k}{\sqrt{p}} \rceil$-relation, is performed. ]

**Stage 4:** This stage executes the update of step 10 of the Program description, and is the most computation time consuming one.

[ **Analysis of Stage 4:** In a single computation superstep $\lceil \frac{n-k}{\sqrt{p}} \rceil^2$ operations (multiplications and subtractions) per processor are performed. ]

**end**

**end**

The communication time of the algorithm is given by the following equation.

$$\begin{aligned}
M(n) &\leq n \cdot g \cdot \left( l \cdot \max\{\tfrac{L}{g}, p^{1/(2l)}\} + \lceil \lg \sqrt{p} \rceil \cdot \max\{\tfrac{L}{g}, 2 \cdot \tfrac{n-k}{\sqrt{p}} + 2\} \right) \\
&\leq \frac{n^3}{3 \cdot p} \cdot g \cdot \left( 3l \cdot \max\{\tfrac{Lp}{gn^2}, \tfrac{p^{1+1/(2l)}}{n^2}\} + 3 \cdot \lceil \lg \sqrt{p} \rceil \cdot \max\{\tfrac{Lp}{gn^2}, \tfrac{2 \cdot \sqrt{p}}{n} + \tfrac{2 \cdot p}{n^2}\} \right) \\
&= \frac{n^3}{3 \cdot p} \cdot \mu
\end{aligned}$$

In order to get $\mu = o(1)$, me must choose $L = o(n^2/(p \cdot \lg p))$, and the value of $g$ is determined identically as in GJEN.

From the analysis of the algorithm, the computation time is given by the following expression.

$$\begin{aligned}
C(n) &\leq n \cdot \left( \max\{L, 1\} + \max\{L, \lceil \tfrac{n-k}{\sqrt{p}} \rceil\} \right) + \sum_{k=1}^{n} \max\{L, \lceil \tfrac{(n-k)}{\sqrt{p}} \rceil^2\} \\
&\leq \frac{n^3}{3 \cdot p} \cdot \left( 3 \cdot \max\{\tfrac{Lp}{n^2}, \tfrac{p}{n^2}\} + 3 \cdot \max\{\tfrac{Lp}{n^2}, \tfrac{\sqrt{p}}{n} + \tfrac{p}{n^2}\} + 1 + \frac{3 \cdot L \cdot p}{n^2} + \frac{3 \cdot p}{n^2} + \frac{3 \cdot \sqrt{p}}{n} \right),
\end{aligned}$$

where we treat sum in a way similar to that of the GJEN algorithm.

In order to get $\pi = 1 + o(1)$, we need to choose $L = o(\frac{n^2}{p})$, $p = o(n^2)$.

The two restrictions on $L$, one from the discussion of $M(n)$, the other from that of $C(n)$, are combined in $L = o(n^2/(p \cdot \lg p))$. ■

It can be seen that for $l = 1$, $\pi = 1 + o(1)$ and $\mu = o(1)$ can be achieved under suitable conditions on $L$ and $g$ if $p = o(n^{4/3})$. Beyond that higher values of $l$ are needed. The case $l = \lg p$ yields:

**Corollary 7** *If $p \log p = o(n^2)$ and $L = o(\frac{n^2}{p \lg p})$, then $\pi = 1 + o(1)$. If in addition $g = o(\frac{n}{\sqrt{p} \lg p})$, then $\mu = o(1)$ also.*

These results show that even in the absence of combining or parallel prefix operations, Cholesky factorization can be solved efficiently even when $L$ is about as large as it can be, and communication is moderately slow.

### 5.8.1 Solution of a Triangular System. Back-substitution

Cholesky factorization was involved with the decomposition of a positive definite matrix $A$ into lower and upper triangular ones (and one matrix is the transpose of the other). If we were to solve for $x$ a system of equations $A \cdot x = b$, we would still need to solve two systems each involving a triangular matrix.

We describe below a back substitution method that can be called twice within the second stage of a linear system solution algorithm whose first stage involved the Cholesky Decomposition of a positive definite matrix. Let $A$ in the discussion to follow be an $n \times n$ upper triangular matrix with positive diagonal elements and let $Ax = b$ be a system of linear equations. We would like to solve $Ax = b$ for $x$. Given the non-singularity of $A$ such a solution exists and is unique. We divide the elements of $A$ among the $p$ processors of the BSP the way we did in GJEN and Cholesky. We also use here the notations used in those sections. Element $b_i$ of vector $b$ is stored in processor $proc(a_{ii})$.

The BSP algorithm consists of $n$ stages and the description of the $(n-i+1)$th stage, $i = 1, \ldots, n$, follows below.

a) Processor $proc(a_{ii})$ finds $x_i = b_i / a_{ii}$.

b) The computed value $x_i$ is broadcast to the $\sqrt{p}$ processors of column $i$ in $k$ supersteps by routing $p^{1/(2k)}$-relations in each such superstep.

c) Each of the $columnp(i)$ processors, for every element $a_{ji}$, $j = 1, \ldots, i$, that is local to it computes $-a_{ji} \cdot x_i$. The result is then sent to processor $proc(a_{jj})$.

d) Processor $proc(a_{jj})$, $j = 1, \ldots, i$ updates $b_j = b_j - a_{ji} x_i$, and proceeds to the following stage $n - i + 2$.

It is easy to see that for $p = o(n^2)$, and for values of $L$ and $g$ those given by the Corollaries summarizing the results for the BSP Cholesky factorization algorithm, we get that the communication and computation times of the Elimination procedure described here are $o(1)$ of the corresponding times for Cholesky factorization.

**Corollary 8** *There exist a BSP back substitution algorithm that can be used in conjunction with Cholesky factorization which requires, for any integer $k$ ($1 \leq k \leq \lg p$), computation time*

$$C(n) \leq \frac{n^3}{3 \cdot p} \left( \frac{6 \cdot \sqrt{p}}{n} + \frac{6p}{n^2} + \frac{9 \cdot L \cdot p}{n^2} \right),$$

*and communication time*

$$M(n) \leq \frac{n^3}{3 \cdot p} \cdot 3 \cdot g \left( k \max\{ \tfrac{Lp}{gn^2}, \tfrac{p^{1+1/(2k)}}{n^2} \} + \max\{ \tfrac{Lp}{gn^2}, \tfrac{\sqrt{p}}{n} + \tfrac{p}{n^2} \} \right).$$

### 5.8.2 Comparison of performance of various linear system algorithms

It is known, for the sequential versions of the algorithms, that Gaussian Elimination (without pivoting) of an $n \times n$ single linear system requires approximately $2n^3/3$ operations (ignoring $O(n^2)$ terms). Cholesky factorization requires half as much time, since the original matrix is a symmetric one. Gauss-Jordan Elimination requires $n^3$ operations, and is therefore, 50% more expensive than Gaussian elimination. The former does not require back substitution, and has the advantage that it is more efficient when more than one systems on the same coefficient matrix need to be solved or the inverse of the coefficient matrix is required.

The BSP version of Gaussian elimination, though not discussed in detail here, except for the back substitution phase, can be directly derived from the Gauss-Jordan elimination and the Cholesky algorithm. The analysis is similar to that of those two algorithms, and the computational time contains a sum that is twice the sum of the corresponding term in the analysis of the BSP version of the Cholesky algorithm.

## 5.9 Recurrences

We describe below BSP algorithms for linear recurrences of order $k-1$ (for constant $k$) by providing an example that discusses the $k-1=2$ case.

Suppose we are given the following recurrence:

$$x_i = a_i x_{i-1} + b_i x_{i-2} + c_i \ , \text{ where } x_0 = A \text{ and } x_1 = B \ .$$

We would like to find $x_n$ for some $n$, and in some cases all terms $x_j, 1 < j \leq n$. The adaptation of such a method to the BSP model is based on the observation of [47] that we can write this recurrence in matrix form as:

$$y_i = \begin{bmatrix} x_i \\ x_{i-1} \\ 1 \end{bmatrix} = \begin{bmatrix} a_i & b_i & c_i \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{i-1} \\ x_{i-2} \\ 1 \end{bmatrix} = M_{i-1} \cdot y_{i-1}, \tag{5.5}$$

where $M_{i-1}$ is a $k \times k$ matrix and $y_i$ is a $k \times 1$ vector. Therefore, $x_n$ can be computed by first computing vector $y_n$ and then reading off its first coordinate. An expression that gives $y_n$ in terms of $y_1$ is given below:

$$y_n = M_{n-1} \cdot \ldots \cdot M_1 \cdot y_1, \tag{5.6}$$

where $y_1$ is given by:

$$y_1 = \begin{bmatrix} B \\ A \\ 1 \end{bmatrix}.$$

If we want to compute $x_n$ only, the computation of the iterated product given by Equation (5.6) suffices. If on the other hand we would like to find all values $x_i, i \geq 2$, Equation (5.6), if we substitute $i$ for $n$ there, suggests that a parallel prefix computation needs to be performed (on an associative operator which operates either on two matrices or on a matrix and a vector).

We present an algorithm that implements this iterated matrix vector product given by Equation (5.6) on a BSP Computer with $p$ processors. We also describe how this algorithm can be extended to compute all $y_i$ simultaneously. The model of computation considered is the arithmetic one.

All vectors and matrices used in this section have dimensions $k \times 1$, and $k \times k$ respectively, and $k-1$ is the order of the recurrence in question.

The space requirements of matrix $M_i$ are linear to the order of the recurrence since from $a_i, b_i, c_i$ alone, we can construct $M_i$.

A product of two $M$ matrices we shall call an $M$-$M$ matrix.

For a recurrence relation of order $k-1$, we need $2k-1 \leq 2k$ operations to multiply two $M$ matrices, $(2k-1)(k-1) \leq 2k^2$ operations to multiply an $M$ matrix and an $M$-$M$ one, since for any $M$ matrix there are $k-1$ rows whose elements are zero or one and known in advance and similarly, for any $M$-$M$ matrix its $k$th row is also known. We need $2k-1$ arithmetic operations to multiply an $M$ matrix by a vector (such as $y_1$, the product of an $M$ matrix with $y_1$, or the product of an $M$-$M$ matrix with $y_1$) to get the result since it suffices to multiply the first row of this $M$ matrix by a vector whose last coordinate is one and the other coordinates of the result vector can be read off the input vector.

In a $(k-1)$-st order recurrence the initial values $x_0, \ldots, x_{k-2}$ are given. The following result due to Chen is described in a more general form in [38] as Corollary 4.

**Fact 11** *A $k$th order linear recurrence system of $n$ equations is defined for $k \leq n$ by*

$$x_i = \begin{cases} 0 & \textit{for } i \leq 0 \\ c_i + \sum_{j=i-m}^{i-1} a_{ij}x_j & \textit{for } 1 \leq i \leq n \end{cases} .$$

*It was proved in [12] that such a system on the arithmetic model of computation can be computed on $p$ processors of unrestricted power, where $k < p \leq n$, in time $T_p$ which is at most*

$$T_p \leq (2k^2 + 3k)\frac{n}{p} + O(k^2 \log p/k).$$

*If $p \leq k$ the obvious algorithm that directly uses the recurrence gives a bound of $T_p = 2\lceil k/p \rceil (n-1)$.*

**Theorem 25** *On the arithmetic model of computation the computation of the $n$-th term of a recurrence relation of order $k-1$ in a single processor requires $2k(n-k+2)$ arithmetic operations if we compute in turn the values $x_{k-1}, x_k, \ldots$ by direct substitution in the recurrence relation of the previously found values, as required. On the BSP model, for $p$ any integer power of integer $d$, there exists an algorithm whose computation and communication times are*

$$C(n) \leq \left( \frac{2kn}{p} \max\{L, 2k^2(1+n/p)\} \ \frac{p}{2kn} + \max\{L, 2k^3(d-1)\} \ \frac{p}{2kn}\frac{\lg p}{\lg d} \right),$$

*and*

$$M(n) = g\max\{\tfrac{L}{g}, dk^2\} \ \frac{\lg p}{\lg d}.$$

**Proof:** We assume that the input has already been distributed evenly among the processors. The input is in the form $(a_i, b_i, c_i)$ since an $M$ matrix can be easily formed from these triples. The algorithm to be presented is essentially Valiant's BSP broadcasting algorithm in reverse.

We first describe informally the algorithm and then present it in pseudocode. The first phase of the algorithm involves the multiplication of $M$ matrices as suggested by Equation (5.6). At most $(2k^2\lceil n/p \rceil)$ arithmetic operations per processor are required for this stage to complete, depending on whether we multiply two $M$ matrices, an $M$ matrix and an $M$-

$M$ matrix, two $M$-$M$ matrices, or a matrix and a vector. Within a single processor, a product of $M$ matrices like the one $M_j \cdot M_k \ldots M_l \cdot M_m$, is computed left associatively as in $((M_j \cdot M_k) \ldots M_l)M_m$.

At the completion of this phase there remain $p$ partial results (one for each processor) that need to be multiplied, in the order implied by Equation (5.6).

A complete $d$-ary tree of depth $\lg p / \lg d$ has $p$ leaves (for $p$ a power of $d$). Such a tree will represent the computations that will be performed in this phase of the algorithm, each internal vertex representing the result of a multiplication (matrix-matrix or matrix-vector, as required) performed on the values evaluated at the children of this vertex. Thus each such computation involves $d-1$ such multiplications. Each of the $p$ partial results obtained in the previous phase of the algorithm is represented by a leaf of that tree, and the order of the leaves indicates the order these results will be multiplied. The second phase of the algorithm thus consists of $\lg p / \lg d$ stages, and an internal vertex of the tree at distance $i$ from its root represents a computation that multiplies in stage $d-i$ of this phase $d$ partial results into one.

Each of these stages consists of two supersteps, one for communication and one for computation. The computation superstep requires at most $2k^3(d-1)$ operations per processor for the multiplication of $(d-1)$, $k \times k$ matrices. The communication superstep requires the routing of a $dk^2$-relation, since in preparation for the execution of the following stage $d$, $k \times k$ matrices, are routed to a single processor. The communication time of such a superstep is thus equal to $gdk^2$.

On the arithmetic model of computation the problem that is being solved requires $O(kn)$ operations for its solution. The BSP algorithm we are to present achieves $O(k^2 n/p)$ performance.

The algorithm is given in pseudocode below. We assume that $p$ is an integer power of integer $d$.

**Procedure** *RECUR*

**begin**

/$*$ **Comment :** The first phase of RECUR follows $*$/

**Superstep 1:** Each processor $i$ such that $1 \leq i \leq n - \lfloor n/p \rfloor p$ has $\lceil n/p \rceil$ $M$ matrices stored in its local memory, that is, processor $i$ has matrices $M_{n-1-(i-1)\lceil \frac{n}{p} \rceil}, \ldots, M_{n-i\lceil \frac{n}{p} \rceil}$ stored in its memory. Processor $i$ such that $i + (n - \lfloor n/p \rfloor p) < p$, $i \geq 1$, stores in its memory the $\lfloor n/p \rfloor$ matrices $M_{n-1-(n-\lfloor n/p \rfloor p)\lceil n/p \rceil-(i-1)\lfloor \frac{n}{p} \rfloor}$ , ... , $M_{n-(n-\lfloor n/p \rfloor p)\lceil n/p \rceil - i\lfloor \frac{n}{p} \rfloor}$ and processor $p$ matrices $M_{\lfloor n/p \rfloor - 1}, \ldots, M_1$, and vector $y_1$. A matrix product like $M_j M_{j-1} \ldots M_{l+1} M_l$ is evaluated left associatively, that is, in the order $((M_j M_{j-1}) \ldots M_{l+1}) M_l$ and according to Equation (5.6).

[ **Analysis of Superstep 1:** This superstep requires at most $2k^2 \lceil n/p \rceil$ arithmetic operations per processor to complete. ]

/$*$ **Comment :** The second phase of RECUR follows $*$/

**for** $k = 1$ **to** $\frac{\lg p}{\lg d}$ do

**Stage k+1:**

**begin**

**Superstep a:** Processor $i$, for $i = 1, \ldots, p/d^{k-1}$, routes its partial result, obtained in superstep 1, to processors $\lceil \frac{i}{d} \rceil$.

[ **Analysis of Superstep a:** This step describes the communication part of the second phase of the algorithm. A processor sends $k^2$ elements to another processor, but overall, a $(d-1)k^2$-relation is routed. ]

**Superstep b:** The processors that received input data in superstep a, multiply the received matrices (or vectors) in the order suggested by Equation (5.6).

[ **Analysis of Superstep b:** This step requires the multiplication of (at most) $d$, $k \times k$, matrices per processor, that is, at most $2(d-1)k^3$ arithmetic operations. ]

**end**

**end**

We now examine the performance of the algorithm.

The computation time for both phases of the algorithm is given by the following expression.

$$C(n) \quad \leq \quad \max\{L, 2k^2 \lceil n/p \rceil\} \; + \max\{L, 2k^3 (d-1)\} \, \frac{\lg p}{\lg d}$$

$$
\begin{aligned}
&= \frac{2k(n-k+2)}{p}\Big(\max\{L, 2k^2(1+n/p)\}\ \frac{p}{2k(n-k+2)} \\
&+ \max\{L, 2k^3(d-1)\}\ \frac{p}{2k(n-k+2)}\frac{\lg p}{\lg d}\Big) \\
&= \frac{2k(n-k+2)}{p}\pi.
\end{aligned}
$$

This time is $(1+o(1))(2k^2 n)/p$ for $L = o((nk^2 \lg d)/(p \lg p))$, and $p \lg p = o((n \lg d)/(kd))$.

The following expression gives the communication time of the algorithm.

$$
\begin{aligned}
M(n) &= g\max\{\tfrac{L}{g}, dk^2\}\ \frac{\lg p}{\lg d} \\
&= \frac{2k(n-k+2)}{p}\mu.
\end{aligned}
$$

This is $o(1)\frac{k^2 n}{p}$ for $g = L/(dk^2) = o(\frac{n \lg d}{dp \lg p})$. ■

We thus get the following Corollary to the Theorem.

**Corollary 9** *If in RECUR, $L = o((nk^2 \lg d)/(p \lg p))$ and $p \lg p = o((n \lg d)/(kd))$, then $\pi = k(1+o(1))$ and if, in addition, $g = o((n \lg d)/(dp \lg p))$, then $\mu = o(k)$.*

Parameter $\pi$ is, as expected, a factor of $k$ off from the performance (divided by $p$) of a sequential algorithm for the same problem. But it is within a $(1+o(1))$ factor from the performance of a PRAM algorithm for this problem as Fact 11 indicates.

Simultaneous computation of all $x_i$'s requires an extension of RECUR. The new algorithm, RECUR2, consist of two parts, the second part of which involves computations related to the second phase of a parallel prefix computation. Valiant's algorithm for parallel prefix on the BSP will be our guide. Following Valiant's paradigm on the BSP, such computations move data from the root towards the leaves of the $d$-ary tree of the Proof of Theorem 25 so that all the partial products $M_i M_{i-1} \ldots M_1 y_1$, $i = 1, \ldots, n-1$, can be computed. In terms of processor communication this second part is the inverse image of RECUR. It consists of two phases. In the first phase – the mirror image of the second phase of RECUR – data move from the root of the $d$-ary to its $p$ leaves and partial products are computed. In the second phase – the mirror image of the first phase of RECUR – at most $\lceil n/p \rceil$ of the products $M_i M_{i-1} \ldots M_1 y_1$,

$i = 1, \ldots, n-1$, are computed in each processor. We note that all computations involved in this second part are matrix-vector related ones, thus each such product requires at most $2k^2$ operations.

The communication supersteps of the second part of RECUR2 involve the routing of vectors, rather than matrices, resulting in a $k$ factor speed up compared to RECUR.

Thus the second part of RECUR2, which computes all $x_i$'s as opposed to RECUR which computes only $x_n$, requires as many arithmetic operations as RECUR alone and $1/k$ the communication time of RECUR.

Thus RECUR2 leads to conclusions similar to those of Corollary 9.

**Corollary 10** *Algorithm RECUR2 which finds all $x_i$, $i = 1, \ldots, n$ requires twice the computation time of RECUR, and communication time which is by a factor $1/k$ greater than that of RECUR. The parameters $L$, $g$, $p$ of RECUR2 are restricted as in RECUR.*

## 5.10 Tridiagonal Matrices and Systems

### 5.10.1 Solution of Tridiagonal Systems

Through back substitution, we can solve a tridiagonal system sequentially in at most $6n$ operations. A lower bound due to Hyafil and Kung but in the arithmetic model of computation without division suggests that the ratio of execution time of a sequential algorithm over the minimum execution time of the corresponding parallel algorithm on $p \leq n$ processors is at most $2p/3$. Thus, the best we can hope for is to have a parallel algorithm for tridiagonal system solution of $9n/p$ running time.

#### An Algorithm

**Claim 7** *Back substitution can be used to solve a tridiagonal system in at most $6n$ sequential operations on the arithmetic model of computation. On the BSP, algorithms RECUR and RECUR2 can be used to solve a tridiagonal system of equations. The $L$ and $g$ parameters of* *a BSP tridiagonal system solver are thus restricted as in these two algorithms. For the values of* $L$, $g$, $p$ *described in Corollary 9 we get a* $\pi = 29(1 + o(1)/6$.

**Proof of Claim 7:** We present the algorithm by providing an example for a $4 \times 4$ tridiagonal matrix. Suppose we want to solve the system:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}.$$

Then, from this system we get that in general, $x_{i+1} = \frac{d_i}{c_i} - \frac{b_i}{c_i} x_i - \frac{a_i}{c_i} x_{i-1}$. Using matrix notation we get:

$$y_{i+1} = \begin{bmatrix} x_{i+1} \\ x_i \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{b_i}{c_i} & -\frac{a_i}{c_i} & \frac{d_i}{c_i} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ x_{i-1} \\ 1 \end{bmatrix} = M_i y_i, \tag{5.7}$$

if we call the corresponding vectors and matrices $y_i$, $M_i$ respectively. It is clear from this relation that $x_n$, can be computed by considering:

$$y_n = M_{n-1} \ldots M_1 y_1 = M_{n-1} y_1, \tag{5.8}$$

where $y_1$ is given by:

$$\begin{bmatrix} x_1 \\ 0 \\ 1 \end{bmatrix}.$$

That way, we can use RECUR2 and Equation (5.8) to compute in parallel all $M_i \cdot \ldots \cdot M_1$, $i = 1, \ldots, n-1$, products. Given that only matrix-matrix products are involved in such a computation, in the second part of RECUR2 if for example the product $M_3 M_2 M_1$ is available, even if the products $M_4$ and $M_5 M_4$ are also available we recombine the partial results by first finding $N_4 = M_4(M_3M_2M_1)$ and then $N_5 = M_5N_4$ rather than using the already computed product $M_5M_4$ to compute $N_5 = (M_5M_4)(M_3M_2M_1)$. This is due to the fact that the product of an $M$-$M$ and an $M$ matrix is less expensive than the product of two $M - M$ matrices.

We can then express $y_n, y_{n-1}$ in terms of $y_1$ by Equation (5.8) and consequently express $x_n$, $x_{n-1}$ in terms of $x_1$. We have three unknowns, $x_n, x_{n-1}$ and $x_1$ and two equations that relate them. The third equation is $a_nx_{n-1}+b_nx_n = d_n$. This corresponds to the last equation of the linear system, given by the $n$th row of its coefficient matrix. In constant time we can solve these equations for $x_1$.

Given $x_1$ we compute in parallel all the $y_i$'s, since the products $M_i \cdot \ldots \cdot M_1$, $i = 1, \ldots, n-1$, have been available and we can read off the first coordinate, $x_i$, of each $y_i$ vector.

We need 5 operations to multiply two $M$ matrices, 10 operations to multiply an $M$-$M$ matrix and an $M$ matrix, and 4 operations to multiply a row of an $M$-$M$ (or the first row of an $M$) matrix and a vector (either $y_1$ or a product of an $M$ or $M$-$M$ matrix with $y_1$).

We need $5\lceil n/p \rceil$ arithmetic operations per processor to prepare the $M$ matrices (three divisions and two negations). In order to compute all $M_i \cdot \ldots \cdot M_1$, $i = 1, \ldots, n-1$ products we need at most $20\lceil n/p \rceil(1+o(1))$ operations per processor ($10\lceil n/p \rceil(1+o(1))$ operations are due to the first part of RECUR2 and an equal amount due to the second one). $4\lceil n/p \rceil(1+o(1))$ operations per processor are required to compute all $y_i$'s, giving a total of at most $29\lceil n/p \rceil(1+o(1)) + O(1)$ operations per processor (the $O(1)$ is due to the solution of the system of three equations). ■

### Another Algorithm

In this section we present another method for a triangular system solution which can be found in [47]. It decomposes a triangular matrix $A$ into three matrices, a diagonal, an upper, and a lower triangular, and thus simplifies the solution of the system. We present this method below. We use a $4 \times 4$ system as an example.

Let us write $A$ is terms of $L$, $D$, $U$ as follows:

$$A = \begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} = LDU =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_2 & 1 & 0 & 0 \\ 0 & l_3 & 1 & 0 \\ 0 & 0 & l_4 & 1 \end{bmatrix} \cdot \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix} \cdot \begin{bmatrix} 1 & m_1 & 0 & 0 \\ 0 & 1 & m_2 & 0 \\ 0 & 0 & 1 & m_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then we get:

$$A = LDU = \begin{bmatrix} d_1 & d_1 m_1 & 0 & 0 \\ l_2 d_1 & l_2 d_1 m_1 + d_2 & d_2 m_2 & 0 \\ 0 & l_3 d_2 & l_3 d_2 m_2 + d_3 & d_3 m_3 \\ 0 & 0 & l_4 d_3 & l_4 d_3 m_3 + d_4 \end{bmatrix}.$$

The relations among $a_i$, $b_i$, $c_i$ and $l_i$, $m_i$, $d_i$ that need to be satisfied are:

$$c_i = d_i m_i \ , \ i = 1, 2, \ldots, n-1,$$

$$a_i = l_i d_{i-1} \ , \ i = 2, \ldots, n,$$

$$b_i = l_i d_{i-1} m_{i-1} + d_i \Rightarrow b_i = \frac{a_i c_{i-1}}{d_{i-1}} + d_i \quad \text{where } b_1 = d_1.$$

If we introduce $t_{i-1} = d_1 d_2 \ldots d_{i-1}$ (with a certain loss to the stability of the resulting algorithm) we get that:

$$b_i t_{i-1} = a_i c_{i-1} t_{i-2} + t_i,$$

which can be rewritten as a second order recurrence relation

$$t_i = b_i t_{i-1} + a_i c_{i-1} t_{i-2}, \text{ where } t_0 = 1,\ t_1 = b_1. \tag{5.9}$$

**Claim 8** *Algorithms RECUR and RECUR2 can be used to solve a tridiagonal system of equations on the BSP. The $L$ and $g$ parameters of this BSP tridiagonal system solver are thus restricted as in these two algorithms. For the values of $L$, $g$, $p$ described in Corollary 9 we get a $\pi = 25(1 + o(1))/6$.*

**Proof of Claim 8:** The LDU decomposition on the BSP of a triangular matrix $A$ follows. We are given matrix $A$ (in the form of elements $a_i$, $b_i$, $c_i$), and from $A$ we compute $t_i$, $i = 1, \ldots, n$, by solving the recurrence given by Equation (5.9). As soon as we compute the $t_i$'s, we get the $d_i$'s using the formula $d_i = t_i/t_{i-1}$, and given the $d_i$'s, the values of $l_i$'s and $m_i$'s are computed. This way the LDU decomposition of $A$ can be completed.

We then have to solve two systems of equations with coefficient matrices matrices $L$ and $U$ (a third system, with coefficient matrix $D$ is solved in $\lceil n/p \rceil$ operations per processor). Matrices $L$ and $U$ are banded, that is elements $l_{ij}$ say of matrix $L$ such that $(i - j) > d$ are zero. Then, $d$ is called the bandwidth of $L$.

A recurrence of the form:

$$t_i = A_i t_{i-1} + B_i t_{i-2}$$

can be solved by the method exhibited in the previous section, the only difference being that matrix $M_i$ is now of a simpler form:

$$\begin{bmatrix} t_i \\ t_{i-1} \end{bmatrix} = M_i \cdot \begin{bmatrix} t_{i-1} \\ t_{i-2} \end{bmatrix} = \begin{bmatrix} A_i & B_i \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} t_{i-1} \\ t_{i-2} \end{bmatrix}.$$

The product of two $M$ matrices requires 3 (arithmetic) operations, while the product of an $M$ and an $M$-$M$ matrix requires 6 operations, and the multiplication of any two other $2 \times 2$ matrices requires 12 operations. On the other hand the product of an $M$ matrix and a vector used in these computations requires only 3 operations since the second coordinate of the result vector can be read off the input vector.

We need $\lceil n/p \rceil$ (arithmetic) operations per processor to prepare, by performing the multiplication $a_i c_{i-1}$, the recurrence given by Equation (5.9).

Hence following the discussion of RECUR2, for the first part of RECUR2 we need $6(1 + o(1))\lceil n/p \rceil$ operations per processor, while in the second part we need $3(1+o(1))\lceil n/p \rceil$ operations per processor (for matrix-vector multiplications). In so many operations the values of $t_i$, $i = 1, \ldots, n$, are computed.

In $3\lceil n/p \rceil$ additional operations the values of $d_i$, $m_i$, $l_i$ can be computed from the $t_i$'s.

We present such an algorithm in terms of the solution of $Lx = b$, where, $L$ is the $4 \times 4$ matrix derived earlier.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_2 & 1 & 0 & 0 \\ 0 & l_3 & 1 & 0 \\ 0 & 0 & l_4 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

From this system we can infer that $x_i = -l_{i-1}x_{i-1} + b_i$, $i > 1$, $x_1 = b_1$. Therefore, matrix $M_i$ of the previous section gets the form:

$$M_i = \begin{bmatrix} -l_{i-1} & b_i \\ 0 & 1 \end{bmatrix}.$$

The product of any two such $M$ matrices (or an $M$ matrix and and $M$-$M$ matrix for that matter) requires three operations, while a matrix vector product requires only two. We need $\lceil n/p \rceil$ operations per processor to prepare the $M$ matrices (one negation per matrix). We use RECUR2 for the solution of this recurrence and thus get an operations count of $5(1 + o(1))\lceil n/p \rceil$ per processor. An equal amount of operations per processor is required for the solution of the system with coefficient matrix $U$, and an amount of $\lceil n/p \rceil$ for $D$.

Therefore if we count all contributions we get that the running time of the algorithm is $25(1 + o(1))\lceil n/p \rceil$.

$\blacksquare$

## 5.11 Sorting

In this section we present an algorithm for sorting $n$ keys on the $p$ processor BSP computer. The algorithm is one-optimal in computation, when comparison and arithmetic operations are counted, for a range of the parameters $p$, $n$ and $L$, and also one-optimal in communication if $g$ is appropriately bounded. The algorithm derives from quicksort [19] and uses the technique of over-sampling previously used in ([33], [45]) in a parallel context.

Let $X = \{x_1, \ldots, x_N\}$ be a set of input keys ordered such that $x_i < x_{i+1}$, for $1 \leq i \leq N-1$. We are assuming that the keys are distinct since we can always make them so by, for example, appending to them the code for the memory locations in which they are stored. For positive integers $k \geq 2$ and $s \geq 1$, let $Y = \{y_1, \ldots, y_{ks-1}\}$ be a randomly chosen subset of $ks - 1 \leq N$ elements of $X$, also ordered so that $y_i < y_{i+1}$. Having so chosen $Y$ we form the $k$ subsets $X_0, \ldots, X_{k-1}$ where

$$X_0 = \{x : x < y_s\},$$

$$X_i = \{x : y_{is} < x < y_{(i+1)s}\} \quad 0 < i < k-1,$$

$$X_{k-1} = \{x : y_{(k-1)s} < x\}.$$

For $1 \leq i \leq ks-1$ and $1 \leq j \leq N$, let $q_i(j)$ be the probability that $y_i = x_j$. i.e. that $y_i$ is the $j$-th element of the sorted input. It is straightforward to verify that

$$q_i(j) = \frac{\binom{j-1}{i-1}\binom{N-j}{ks-1-i}}{\binom{N}{ks-1}}.$$

We follow the standard notation for the binomial coefficients, taking $\binom{i}{j} = 0 \quad \text{if} \quad j < i$.

For $0 \leq i \leq k-1$ let $n_i = |X_i|$ and for $0 \leq i \leq k-1$, $s-1 \leq j \leq N-(k-1)s$, let $p_i(j)$ be the probability that $n_i = j$. The following was proved in [19]:

**Claim 9** *For all* $i$,

$$p_i(j) = p(j) = \frac{\binom{j}{s-1}\binom{N-j-1}{(k-1)s-1}}{\binom{N}{ks-1}}.$$

**Proof:** The $i = 0$ and $i = k-1$ cases are straightforward. For all other cases we have that:

$$p_i(j) = \sum_t q_{is}(t) q_{(i+1)s}(t+j+1 \mid y_{is} = x_t).$$

The conditional probability is just $q_s(j+1)$ but redefined for a sample of size $ks - is - 1$ from a set of size $(N-t)$. Substituting the expressions in this probability from the formula for $q_i(j)$ derived earlier, we get that:

$$\begin{aligned}
p_i(j) &= \sum_{t=is}^{N-j-(k-i)s+s} \frac{\binom{t-1}{is-1}\binom{N-t}{(k-i)s-1}}{\binom{N}{ks-1}} \frac{\binom{j}{s-1}\binom{N-t-j-1}{(k-i)s-s-1}}{\binom{N-t}{(k-i)s-1}} \\
&= \sum_{t=is}^{N-j-(k-i)s+s} \frac{\binom{t-1}{is-1}\binom{j}{s-1}\binom{N-t-j-1}{(k-i)s-s-1}}{\binom{N}{ks-1}} \\
&= \binom{j}{s-1} \sum_{t=is}^{N-j-(k-i)s+s} \frac{\binom{t-1}{is-1}\binom{N-t-j-1}{(k-i)s-s-1}}{\binom{N}{ks-1}} \\
&= \frac{\binom{j}{s-1}\binom{N-j-1}{(k-1)s-1}}{\binom{N}{ks-1}}.
\end{aligned}$$

We got the last equation from the following combinatorial identity (see [35], page 58)

$$\sum_{n-u \le v \le r-m} \binom{u+v}{n}\binom{r-v}{m} = \binom{r+u+1}{m+n+1},$$

after making the substitution in the identity $r = N-j-2$, $v = t-1$, $m = (k-i)s - s - 1$, $n = is - 1$, $u = 0$. ■

We would like to prove that the probability of having a set of size greater than $(1+\epsilon)$ times the mean, $(N-k+1)/k$, is very small, in particular that

$$\sum_{j \ge \lceil (1+\epsilon)(N-k+1)/k \rceil + 1} p(j)$$

is sufficiently small, where $0 < \epsilon < 1$.

We first observe that the combinatorial identity quoted above implies also that

$$\sum_{s-1 \le j \le N-(k-1)s} \binom{j}{s-1} \binom{N-j-1}{(k-1)s-1} = \binom{N}{ks-1}.$$

This identity can be combinatorially interpreted as follows. The number of ways to choose $ks-1$ objects out of $N$ (labeled 1 through $N$), is the number of ways to choose $s-1$ objects among those labeled $1, \ldots, j$, and $(k-1)s-1$ among these labeled $j+2, \ldots, N$, summed over all choices of $j+1$ that represent the various possible values of the $s$-th object.

We are interested in the portion of the sum for $j \ge \lceil (1+\epsilon)(N-k+1)/k \rceil + 1$, that is the number of ways we can choose $ks-1$ objects so that among them, the object with the $s$-th lowest label is labeled at least $\lceil (1+\epsilon)(N-k+1)/k \rceil + 2$.

In order to establish an upper bound for the term

$$\sum_{j \ge \lceil (1+\epsilon) \frac{N-k+1}{k} \rceil + 1} p(j)$$

we shall approximate the $p(j)$ by a Binomial term and use bounds on the left tail of the Binomial distribution. Let $q = (ks-1)/N$. The probability that if we make a succession of $N$ Bernoulli trials, each with a probability $q$ of success that we get $s-1$ successes among the first $j$, a success at the $(j+1)$-st trial, and $(k-1)s-1$ successes in the remainder is

$$r(j) = \binom{j}{s-1} q^{s-1} (1-q)^{j-s+1} \; q \; \binom{N-j-1}{(k-1)s-1} q^{(k-1)s-1} (1-q)^{N-j-(k-1)s}.$$

Let $A$ be the ratio of $r(j)$ over $p(j)$. Then

$$A = \frac{r(j)}{p(j)} = \binom{N}{ks-1} q^{ks-1} (1-q)^{N-ks+1}.$$

We are going to use a fact deducible from Stirling's approximation (see also [10], page 4) that holds for $a = c/d$, $b = 1 - a$, $c < d/2$, and gives that

$$e^{-\frac{1}{6c}} \frac{1}{\sqrt{2\pi abd}} \left(a^a b^b\right)^{-d} \leq \binom{d}{c}.$$

If we set $d = N$, $c = ks - 1$, $a = q = \frac{ks-1}{N}$, and $b = 1 - \frac{ks-1}{N}$ then

$$e^{-\frac{1}{6(ks-1)}} \frac{1}{\sqrt{2\pi(ks-1)(1-\frac{ks-1}{N})}} q^{-ks+1} (1-q)^{-N+ks-1} \leq \binom{N}{ks-1}.$$

Then, we get that

$$A \quad \geq \quad e^{-\frac{1}{6(ks-1)}} \frac{1}{\sqrt{2\pi(ks-1)(1-\frac{ks-1}{N})}}.$$

Since by definition $r(j) \leq \binom{j}{s-1} q^{s-1}(1-q)^{j-s+1} q$ and $p(j) = r(j)/A$,

$$\sum_{j \geq \lceil (1+\epsilon)\frac{N-k+1}{k} \rceil + 1} p(j) \quad \leq \quad \frac{q}{A} \sum_{j \geq \lceil (1+\epsilon)\frac{N-k+1}{k} \rceil + 1} \binom{j}{s-1} q^{s-1}(1-q)^{j-s+1}.$$

We shall now try to find an upper bound for the term

$$\sum_{j \geq \lceil (1+\epsilon)\frac{N-k+1}{k} \rceil + 1} \binom{j}{s-1} q^{s-1}(1-q)^{j-s+1}.$$

Let $B = \lceil (1+\epsilon)(N-k+1)/k \rceil + 1$. The method to be described uses generating functions. In particular, the term $\binom{j}{s-1} q^{s-1}(1-q)^{j-s+1}$ is the coefficient of $x^{s-1}$ in the polynomial $((1-q)+qx)^j$.

$$\begin{aligned}
\sum_{j \geq B} \binom{j}{s-1} q^{s-1}(1-q)^{j-s+1} \quad &= \quad \text{coefficient of } x^{s-1} \text{ in } \sum_{j \geq B} ((1-q)+qx)^j \\
&\leq \quad \text{coefficient of } x^{s-1} \text{ in} \\
&\qquad \left( \frac{1}{1-[(1-q)+qx]} - \sum_{0 \leq j < B} ((1-q)+qx)^j \right) \\
&= \quad \text{coefficient of } x^{s-1} \text{ in}
\end{aligned}$$

$$
\begin{aligned}
& \left(\frac{1}{1-[(1-q)+qx]} - \frac{((1-q)+qx)^B - 1}{[(1-q)+qx]-1}\right) \\
= & \quad \text{coefficient of } x^{s-1} \text{ in } \left(\frac{((1-q)+qx)^B}{q(1-x)}\right) \\
= & \quad \frac{1}{q}\sum_{i=0}^{s-1} \text{coefficient of } x^i \text{ in } \left(((1-q)+qx)^B\right) \\
= & \quad \frac{1}{q}\sum_{i=0}^{s-1} \binom{B}{i} q^i (1-q)^{B-i}.
\end{aligned}
$$

This sum can be viewed as the left tail of a Binomial distribution having $B$ trials and probability $q$ of success at each trial. But in general Chernoff's upper bound on the probability of having no more than $(1-\beta)Bq$ successes if the expectation is $Bq$ is (see [4]):

$$e^{-\frac{1}{2}\beta^2(Bq)}.$$

Since we need $s-1 = (1-\beta)Bq$, the above bound becomes

$$e^{-\frac{1}{2}\frac{(Bq-(s-1))^2}{Bq}}.$$

We first find a lower bound for the term $Bq-(s-1)$, for $ks \leq N/2$.

$$
\begin{aligned}
Bq-(s-1) &= \left(\lceil(1+\epsilon)\frac{N-k+1}{k}\rceil + 1\right)\frac{ks-1}{N} - (s-1) \\
&\geq s(1+\epsilon) - (1/k)(1+\epsilon) - (ks-1)\epsilon/N + \frac{(1+\epsilon)}{k}\frac{ks-1}{N} - (s-1) \\
&\geq \epsilon s + \frac{N(k-1-\epsilon) - (ks-1)(k-\epsilon-1)}{kN},
\end{aligned}
$$

if $\epsilon < 1$.

We also need an upper bound for $Bq$.

$$
\begin{aligned}
Bq &\leq \left(\lceil(1+\epsilon)\frac{N-k+1}{k}\rceil + 1\right)\frac{ks-1}{N} \\
&\leq s(1+\epsilon) + \frac{-N(1+\epsilon) + (ks-1)(1+k+\epsilon-k\epsilon)}{kN}.
\end{aligned}
$$

Combining the lower bound for $Bq-(s-1)$ and the upper bound for $Bq$, gives that

$$\begin{aligned}
e^{-\frac{1}{2}\beta^2(Bq)} &\leq e^{-\frac{1}{2}\frac{(Bq-(s-1))^2}{Bq}} \\
&\leq e^{-\frac{1}{2}\frac{\left(\epsilon s+\frac{(N-(ks-1))(k-1-\epsilon)}{kN}\right)^2}{s(1+\epsilon)+\frac{-N(1+\epsilon)+(ks-1)(1+k+\epsilon-k\epsilon)}{kN}}} \\
&\leq e^{-\frac{1}{2}\frac{\epsilon^2 s^2+\frac{2\epsilon s(N-(ks-1))(k-1-\epsilon)}{kN}}{s(1+\epsilon)+\frac{-N(1+\epsilon)+(ks-1)(1+k+\epsilon-k\epsilon)}{kN}}} \\
&\leq e^{-\frac{1}{2}\frac{\epsilon^2 s}{(1+\epsilon)}}.
\end{aligned}$$

The last inequality can be derived by some calculations, that use the assumptions that $ks \leq N/2$, $k \geq 2$, $0 < \epsilon < 1$. Thus,

$$\begin{aligned}
\sum_{j \geq \lceil (1+\epsilon)\frac{N-k+1}{k}\rceil+1} \binom{j}{s-1} q^{s-1}(1-q)^{j-s+1} &= \frac{1}{q}\sum_{i=0}^{s-1}\binom{B}{i} q^i (1-q)^{B-i} \\
&\leq \frac{1}{q} e^{-\frac{1}{2}\;\frac{\epsilon^2}{1+\epsilon}\;s}.
\end{aligned}$$

Thus if we substitute for $A$, $B$, and if we note that $q=(ks-1)/N$, we get the following.

$$\begin{aligned}
\sum_{j \geq \lceil (1+\epsilon)\frac{N-k+1}{k}\rceil+1} p(j) &\leq q\,\sqrt{2\pi(ks-1)}\,e^{\frac{1}{6(ks-1)}} \sum_{j \geq \lceil (1+\epsilon)\frac{N-k+1}{k}\rceil+1} \binom{j}{s-1} q^{s-1}(1-q)^{j-s+1} \\
&\leq \sqrt{2\pi(ks-1)}\,e^{\frac{1}{6(ks-1)}}\; e^{-\frac{1}{2}\;\frac{\epsilon^2}{1+\epsilon}\;s}.
\end{aligned}$$

We can now deduce the following which we shall use subsequently for $n \geq N$ and $s$ polylogarithmic in $n$.

**Claim 10** *Let $k \geq 2$, $s \geq 1$, $ks < N/2$, $n \geq 1$, $0 < \epsilon < 1$, $\rho > 0$, and*

$$s \geq \frac{1+\epsilon}{\epsilon^2}\left(2\rho \log n + \log\left(2\pi k^2(ks-1)e^{1/(3(ks-1))}\right)\right).$$

*Then the probability that any one of the $p_i(j)$ for $0 \leq i < k$ is of size more than $\lceil (1+\epsilon)\frac{N-k+1}{k}\rceil$ is at most $1/n^\rho$.*

**Proof:** The probability that a given set, among the $k$ possible ones, is of size more than $\lceil (1+\epsilon)\frac{N-k+1}{k} \rceil$ is, following our previous discussion, at most

$$\sqrt{2\pi(ks-1)}\, e^{\frac{1}{6(ks-1)}}\; e^{-\frac{1}{2}\,\frac{\epsilon^2}{1+\epsilon}\,s}.$$

The probability that some, among the $k$ sets, are so is at most $k$ times this term, that is,

$$k\,\sqrt{2\pi(ks-1)}\, e^{\frac{1}{6(ks-1)}}\; e^{-\frac{1}{2}\,\frac{\epsilon^2}{1+\epsilon}\,s}.$$

If we substitute for the value of $s$, we get that the probability of having some sets of sizes more than $\lceil (1+\epsilon)\frac{N-k+1}{k} \rceil$ is at most $1/n^\rho$. ■

The following are two corollaries that we shall use, where $p$ will denote the number of processors.

**Remark 1** *If $N = n$, $k = p \geq 2$, $p^3 \leq n/\lg^2 n$, $\epsilon = 1/\sqrt{\log n}$, and $s = 1.8 \log^2 n$, then, for $n > 3500$ and a suitable $\rho$, the conditions of Claim 10 are satisfied and the probability that any of the $k$ buckets into which the input set was split has size more than $\lceil (1+\epsilon)\frac{N-k+1}{k} \rceil$ is $o(1)$.*

**Proof:** Since $s = 1.8 \log^2 n$, $n > 3500$, and $k \geq 2$, we get $e^{1/6(ks-1)} \leq 1.0004$. For $N = n$, $k = p$, $p^3 \leq n/\lg^2 n$, $\epsilon = 1/\sqrt{\log n}$, if we substitute in the expression for $s$ in Claim 10 we get the following

$$s \geq \left(2\rho + \left(1 + \frac{\log 2}{\log n}\right) + \frac{\log 2\pi}{\log n} + 2 \cdot \frac{\log 1.0004}{\log n}\right)\;\left(1 + 1/\sqrt{\log n}\right)\log^2 n.$$

For $n > 3500$ we need only choose an

$$s \geq (2\rho + 1.3104)\;\left(1 + 1/\sqrt{\log n}\right)\log^2 n.$$

Therefore any $\rho > 0$ such that $(2\rho + 1.3104)(1 + 1/\sqrt{\log 3500}) \leq 1.8$ will satisfy the expression in Claim 10. It also satisfies $ks < N/2$. ■

**Remark 2** *If $N = n$, $k = p \geq 2$, $p^2 \leq n/(6\log^3 n)$, $\epsilon = 1/\sqrt{\log n}$, and $s = 2.25\log^2 n$, then, for $n > 25000$ and a suitable $\rho$, the conditions of Claim 10 are satisfied and the probability that any of the $k$ buckets into which the input set was split has size more than $\lceil (1+\epsilon)\frac{N-k+1}{k} \rceil$ is $o(1)$.*

**Proof:** Since $s = 2.25\log^2 n$, $n > 25000$ and $k \geq 2$, we get $e^{1/6(ks-1)} \leq 1.0004$. For $N = n$, $k = p$, $p^2 \leq n/(6\log^3 n)$, $\epsilon = 1/\sqrt{\log n}$, if we substitute in the expression for $s$ in Claim 10 we get the following

$$s \quad = \quad \left(2\rho + 1.5 + \frac{\log 2\pi}{\log n} + 2 \cdot \frac{\log 1.0004}{\log n}\right)\left(1 + 1/\sqrt{\log n}\right)\log^2 n.$$

For $n > 25000$ we need only choose

$$s \geq (2\rho + 1.69)\left(1 + 1/\sqrt{\log n}\right)\log^2 n.$$

Therefore any $\rho > 0$ such that $(2\rho+1.69)(1+1/\sqrt{\log 25000}) \leq 2.25$ will satisfy the expression in Claim 10. It also satisfies $ks < N/2$. ■

The algorithm below for sorting $n$ items performs $m$ iterations, where $m$ may equal 1 or 2 in practical situations. In each iteration $k-1$ "splitters" are chosen , where $k = p^{1/m}$, and the set partitioned into $k$ subsets of approximately equal size by these splitters. The next iteration performs a similar process of subpartitioning each of these $k$ subsets into $k$ further subsets, etc.

At each stage if the subset to be partitioned is of size $N$ the $k$ splitters are selected by first selecting $ks-1$ random elements from the set, sorting this selection and picking out the $k-1$ elements that partition this selection equally. By Claim 10, if we choose $s$ appropriately then we will be guaranteed that these splitters split the whole set of size $N$ almost equally (i.e. to within a factor of $(1+\epsilon)$), with high probability. To specify the size of $s$ we will select a slowly growing function $\omega_n \to \infty$ (e.g. $\sqrt{\log n}$) and choose $\epsilon = (m\omega_n)^{-1}$ in Claim 10.

Then the size of every subset produced in the $m$ iterations will be within a factor $(1+\epsilon)^m = (1 + 1/(m\omega_n))^m \leq e^{1/\omega_n}$ of the mean with probability at least $1 - n^{-\rho}(p-1)/(k-1)$. Since we will want this to be $1 - o(1)$ it will be sufficient to choose $\rho$ to exceed $\log(p/k)/\log n$.

The model of computation considered for sequential sorting is the comparison model. For BSP sorting we extend this model to include arithmetic operations and in the analysis of our sorting algorithm we take into account such operations that are related to various parallel prefix computations required by this BSP sorting algorithm. In the course of the sorting algorithm a random sample also needs to be generated. We consider in this model the time needed to generate a sample of some size $s > 0$ to be $s$ time steps. This assumes that an operation *random*(p,k,x), where $0 \leq k < p - 1$, and $x \geq 0$, exists that returns in one time step a randomly chosen number between 0 and $p - 1$, where the random number generator is slightly biased so that a number between (inclusive) 0 and $k - 1$ has probability of appearance equal to $(x + 1)/(px + k)$ and a number greater than $k - 1$ has probability of appearance equal to $x/(px + k)$. A justification for the need for such an operator appears in the proof to follow. Comparisons in the sorting algorithm are followed by the execution of "exchange" operations. We thus assume the existence of a compare-and-store operation. Thus *compare-and-store*(key,bucket.leftkey,bucket.rightkey,bucket.size) decides whether *key* is within the bucket whose left and right boundaries are determined by the two splitters bucket.leftkey and bucket.rightkey and whose size is bucket.size. If *key* is within these boundaries it is appended on a queue storing the elements of this bucket. The value of bucket.size is then incremented.

**Proposition 15** *On the comparison model of computation sorting $n$ items requires $n \lg n$ comparisons. Let $\omega_n$ be any function $\omega_n \to \infty$ as $n \to \infty$. For any integers $p$, $k$, $s$, such that $p$ is an integral power of $k$ and the conditions of Claim 10 are satisfied with $\epsilon = (\omega_n \lg p / \lg k)^{-1}$, $N = n$ and $\rho > \log(p/k)/\log n$, there is a BSP sorting algorithm that for any integer $l \geq 1$, has, with probability $1 - o(1)$,*

$$\begin{aligned}
\pi \quad &\leq \quad \frac{p \lg p}{n \lg n \lg k} \max\{L, \lceil (1 + \tfrac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} \tfrac{n}{p} \rceil (\lceil \lg k + 1 \rceil)\} \\
&+ \quad \frac{p}{n \lg n} \max\{L, \lceil \frac{(1 + \frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil \lg \lceil \frac{(1 + \frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil\}
\end{aligned}$$

$$
\begin{aligned}
&+ \quad \frac{p \lg p}{n \lg n \lg k} \max\{L, (ks-1) \lg (ks-1)\} \\
&+ \quad \frac{p \lg p}{n \lg n \lg k} \max\{L, s\} \\
&+ \quad \frac{p}{n \lg n} \left( \frac{3 \lg p}{\lg k} - 1 \right) l \max\{L, k \lceil p^{1/l} \rceil\} \ ,
\end{aligned}
$$

*and*

$$
\begin{aligned}
\mu \quad = \quad & g \, ( \frac{p \lg p}{n \lg n \ \lg k} \max\{\tfrac{L}{g}, (ks-1)\} \ + \frac{p}{n \lg n} \left( \frac{4 \lg p}{\lg k} - 1 \right) l \max\{\tfrac{L}{g}, k \lceil p^{1/l} \rceil\} \\
+ \quad & \frac{p \lg p}{n \lg n \ \lg k} \max\{\tfrac{L}{g}, 2s\} \ + \frac{p \lg p}{n \lg n \lg k} \max\{\tfrac{L}{g}, \lceil (1 + \tfrac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} \ n/p \rceil\} \ ).
\end{aligned}
$$

*The expression for $\pi$ includes all arithmetic as well as comparison operations.*

**Proof:** The algorithm is recursive and each call is divided into a sequence of stages. Initially, the variables *Left*, *Right* have the values $Left = 1$, $Right = p$. In any recursive call of a procedure, these two variables give the range of the processors that will participate. The input is initially evenly distributed among the $p$ processors before the beginning of the execution of the algorithm.

**Program** *Sort(Left,Right)*

**begin**

**Stage 0 :**

If *Left* = *Right* then

**begin**

Every processor, in parallel with all the other processors, sorts the elements in its possession;

Exit-Program;

**end**

[ **Analysis of Stage 0:** When this step is executed, every processor will hold at most $\lceil (1 + 1/(\omega_n (\lg p / \lg k)))^{\lg p / \lg k} \frac{n}{p} \rceil$ keys. The cost is that of sequential sorting by some algorithm such as [31], [53]. ]

**Stage 1 :**

Among the $P = Right - Left + 1$ processors participating in this call of the algorithm, $k$ processors choose at random with replacement $s$ processor identifiers each. Each such identifier will correspond to a sampling within the identified processor.

During the first call to this algorithm (when $P = p$), as well as in each recursive call, the input is evenly distributed among the processors involved in the call, in the sense that for some $x$ each one has either $x$ or $x+1$ keys. Hence the random sampling among the identifiers will be done by simulating a coin that is slightly biased to compensate. The information collected during stage 3 to follow will be sufficient for computing this bias. In each call stage 4 will place in each group of $P/k$ processors $x+1$ keys in the first $r$, and $x$ in the remainder. The information that has to be distributed, to enable the bias to be computed, is the value of $r$ for each of the $k$ groups. This information is available to processor *Right* at the end of stage 3.

The $k$ processors then route these identifiers to the processors identified by them. The identified processors, as soon as they receive the messages, randomly sample their input as many times as the number of messages received.

The $ks$ elements so identified are routed to processor $Left$.

[ **Analysis of Stage 1:** This stage consists of two parts. Prior to the first part $k$ numbers will be broadcast to all $p$ processors. If we broadcast in $l$ supersteps, then in each superstep it is sufficient to route a $k\lceil p^{1/l}\rceil$-relation. In the first part each of $k$ processors randomly chooses $s$ identifiers (numbers in the range *Left* through *Right*). We assume that this requires $s$ time steps (though the actual time will depend on the implementation of the random number generator used). In the following superstep each identifier is routed to the processor identified by it. With probability $1 - o(1/n)$ a $2s$-relation will be routed. In the second part we need to route a $ks$-relation.

]

**Stage 2 :**

(i) Processor $Left$ sorts the sample, and selects the $k-1$ splitters.

(ii) It then initiates a broadcast of the $k-1$ splitters to the other $P-1$ processors of this recursive call as well as, the index of every splitter in the set of splitters, so that the processors receive the splitters effectively sorted.

[ **Analysis of Stage 2:** Sequential sorting requires $(ks-1)\lg(ks-1)$ comparisons. If a (splitter, index) pair can be sent as one message and we want to broadcast in $l$ supersteps, then in each superstep it is sufficient to route a $k\lceil p^{1/l}\rceil$-relation.

]

**Stage 3 :**

(i) Each processor, in a single superstep, decides the position of every element it holds with respect to the $k-1$ splitters. It then counts the number of its elements that fall into each of the $k$ intervals.

(ii) We perform $k$ (one for each bucket) independent parallel prefix computations in order to determine how to split the elements of each bucket as evenly as possible among the processors that will hold them, using the information collected in step (i) of this stage.

[ **Analysis of Stage 3:** For (i) we need time $\lceil (1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k} n/p\rceil$ $(\lceil \lg k\rceil + 1)$. The analysis of the parallel prefix computations is similar to that for the broadcasting in the previous stage. ]

**Stage 4 :**

Every processor knows where to route every key in its possession, and performs this routing. Processors route keys in "row-major" form, that is, keys destined to some bucket fill the processors assigned to this bucket in cyclic order, the first processor of that order being determined by the information gathered in part (ii) of stage 3.

[ **Analysis of Stage 4:** We need to route an $\lceil (1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k}\ n/p\rceil$-relation here. ]

**Stage 5 :**

After the routing has been completed, a recursive call has been completed as well.

**begin**

for $i = 1$ to $k$ do in parallel

*Sort* ( $(i-1)\frac{p}{k}+1, i\ \frac{p}{k}$ )

**end**

[ **Analysis of Stage 5:** The depth of the recursion is $\lg p/\lg k$. ]

**end**

**Specification of $\pi$ and $\mu$:** A $\frac{\lg p}{\lg k}\max\{L,(ks-1)\lg(ks-1)\}$ number of comparisons is required for sorting the sample only. We also need to spend $2\max\{L,kp^{1/l}\}\ \frac{\lg p}{\lg k}l$ time steps for addition in the parallel prefix computation. A $\max\{L,kp^{1/l}\}\ (\frac{\lg p}{\lg k}-1)l$ time steps are needed to decide which processors are unbalanced. We also charge $\max\{L,s\}\ \frac{\lg p}{\lg k}$ time steps for the generation of the sample. At most

$$\max\{L,\lceil(1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k}n/p\rceil(\lceil\lg k\rceil+1)\}\ \frac{\lg p}{\lg k}$$

time is spent on locating the bucket to which every key of a given processor belongs. Finally,

$$\max\{L,\lceil\frac{(1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k}n}{p}\rceil\lg\lceil\frac{(1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k}n}{p}\rceil\}$$

comparisons are required at the end of the recursion.

The sum of all the partial factors gives the computation time $C(n)$.

$$\begin{aligned}
C(n) &= \frac{n\lg n}{p}(\frac{p\lg p}{n\lg n\lg k}\max\{L,(ks-1)\lg(ks-1)\}\\
&+ \frac{p\lg p}{n\lg n\lg k}\max\{L,s\}\ +\frac{p}{n\lg n}\ \left(\frac{3\lg p}{\lg k}-1\right)\ l\max\{L,k\lceil p^{1/l}\rceil\}\\
&+ \frac{p\lg p}{n\lg n\lg k}\max\{L,\lceil(1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k}n/p\rceil(\lceil\lg k\rceil+1)\}\\
&+ \frac{p}{n\lg n}\max\{L,\lceil\frac{(1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k}n}{p}\rceil\lg\lceil\frac{(1+1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k}n}{p}\rceil\}\ )\\
&= \pi\frac{n\lg n}{p}.
\end{aligned}$$

The communication part is derived similarly. We need $g\max\{\frac{L}{g},ks-1\}\ \frac{\lg p}{\lg k}$ time just to route the sample to a given processor for all recursive calls. Broadcasting and parallel prefix related computations require total communication time of $g \max\{\frac{L}{g}, k\lceil p^{1/l}\rceil\}\ l(4\frac{\lg p}{\lg k} - 1)$, the distribution of the sample requires $g \max\{\frac{L}{g}, 2s\}\ \frac{\lg p}{\lg k}$, while the routing of the elements to be sorted to the appropriate processors through all the recursive calls, requires

$$g \max\{\tfrac{L}{g}, \lceil (1 + 1/(\omega_n (\lg p/\lg k)))^{\lg p/\lg k} n/p\rceil\}\ \tfrac{\lg p}{\lg k}$$

time. Thus we get for $M(n)$ the following expression.

$$\begin{aligned}
M(n) &= g\frac{n\lg n}{p}\Big(\frac{p\lg p}{n\lg n\lg k}\max\{\tfrac{L}{g}, (ks-1)\} \\
&+ \frac{p\lg p}{n\lg n\lg k}\max\{\tfrac{L}{g}, 2s\} \\
&+ \frac{p}{n\lg n}\left(\frac{4\lg p}{\lg k} - 1\right) l \max\{\tfrac{L}{g}, k\lceil p^{1/l}\rceil\} \\
&+ \frac{p\lg p}{n\lg n\lg k}\max\{\tfrac{L}{g}, \lceil (1 + 1/(\omega_n(\lg p/\lg k)))^{\lg p/\lg k} n/p\rceil\} \\
&= \mu\frac{n\log n}{p}.
\end{aligned}$$

■

By substituting appropriate values for the parameters in the above proposition a variety of bounds on the runtime of the algorithm can be obtained. The best bounds are obtained by choosing $m$, $\omega_n$ with regard to the actual values of $n$, $p$ and $L$, and substituting these numerical values. By making more generic choices for $m$ and $\omega_n$ and worst case bounds for $p$ and $L$, more general but less tight results in terms of $n$ alone can be obtained. Theorems 26 and 27 are two such bounds that are nevertheless, still applicable for reasonable values of $p$, $n$ and $L$.

We can prove a number of technical lemmas that will help us in analyzing the performance of various algorithms.

**Lemma 5** *Let $k = p \le n$, $L \le n/p$. Then*

$$\frac{p\lg p}{n\lg n\lg k}\max\{L, \lceil (1 + \tfrac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} \tfrac{n}{p}\rceil(\lceil \lg k\rceil + 1)\}\ +$$

$$\begin{aligned}
\frac{p}{n\lg n}\max\{L, \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil \lg \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil \} \quad & \leq \\
& 1 + \frac{1}{\omega_n} + \left(1 + \frac{1}{\omega_n}\right)\frac{1}{\log 2\omega_n \lg n} \\
& + \left(1 + \frac{1}{\omega_n}\right)\frac{2}{\lg n} + \frac{p}{n} \\
& + \frac{p}{\log 2\omega_n n \lg n} + \frac{2p}{n\lg n} \\
& + \frac{p}{n\log n} + \frac{1}{(1 + 1/\omega_n)\log n}.
\end{aligned}$$

**Proof:** Since $L \leq n/p$, $L \leq \lceil (1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} \frac{n}{p} \rceil (\lceil \lg k \rceil + 1)$, and $L \leq \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil \lg \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil$. For $p = k$ we then get

$$\frac{p\lg p}{n\lg n \lg k}\max\{L, \lceil (1 + \frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} \frac{n}{p} \rceil (\lceil \lg k \rceil + 1)\} +$$

$$\frac{p}{n\lg n}\max\{L, \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil \lg \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p} \rceil \} \leq$$

$$\frac{p}{n\lg n}\left(1 + \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right)(2 + \lg k) + \frac{p}{n\lg n}\left(1 + \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right)\lg\left(1 + \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right) \leq$$

$$\frac{p}{n\lg n}\left(1 + \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right)(2 + \lg k) + \frac{p}{n\lg n}\left(1 + \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right) \cdot$$

$$\cdot \left(\lg\left(\left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right) + 1/\left(\log 2\left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right)\right),$$

where we used in the last expression the general inequality $\lg(1 + x) \leq \lg x + 1/(x \log 2)$. Thus a substitution $x = \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}$, yields our claim.

Calculations then give that

$$\begin{aligned}
& \leq \quad \frac{p}{n\lg n}\left(2 + \lg k + \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\lg k + 2\left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right) \\
& + \quad \frac{p}{n\lg n}\left(\lg\left(\left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right) + \left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\lg\left(\left(1 + \frac{1}{\omega_n}\right)\frac{n}{p}\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&+ \quad \frac{p}{n\log n} + \frac{1}{\log n(1+1/\omega_n)} \\
&\leq \quad 1 + \frac{1}{\omega_n} + \left(1+\frac{1}{\omega_n}\right)\frac{1}{\omega_n \lg n \log 2} + \left(1+\frac{1}{\omega_n}\right)\frac{2}{\lg n} \\
&+ \quad \frac{p}{n} + \frac{p}{\omega_n n \lg n \log 2} + \frac{2p}{n\lg n} + \frac{p}{n\log n} + \frac{1}{(1+1/\omega_n)\log n}.
\end{aligned}
$$

This completes the proof of the lemma. ■

In case $L \leq n/p$ we need to take into account the contributions of $L$ in the maxima that were previously ignored. We then get the following more generalized version of this Lemma.

**Corollary 11** *Let $k = p$. Then*

$$
\begin{aligned}
&\frac{p\lg p}{n\lg n\lg k}\max\{L, \lceil(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}\frac{n}{p}\rceil(\lceil\lg k\rceil+1)\} \ + \\
&\frac{p}{n\lg n}\max\{L, \lceil\frac{(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}n}{p}\rceil\lg\lceil\frac{(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}n}{p}\rceil\} \quad \leq \\
&\qquad 1 + \frac{1}{\omega_n} + \left(1+\frac{1}{\omega_n}\right)\frac{1}{\log 2\omega_n\lg n} \\
&+ \quad \left(1+\frac{1}{\omega_n}\right)\frac{2}{\lg n} + \frac{p}{n} \\
&+ \quad \frac{p}{\log 2\omega_n n\lg n} + \frac{2p}{n\lg n} \\
&+ \quad \frac{2pL}{n\lg n} + \frac{1}{\log n(1+1/\omega_n)} + \frac{p}{n\log n}.
\end{aligned}
$$

**Lemma 6** *Let $k < p \leq n$, $L \leq n/p$. Then*

$$
\begin{aligned}
&\frac{p\lg p}{n\lg n\lg k}\max\{L, \lceil(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}\frac{n}{p}\rceil(\lceil\lg k\rceil+1)\} \quad + \\
&\frac{p}{n\lg n}\max\{L, \lceil\frac{(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}n}{p}\rceil\lg\lceil\frac{(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}n}{p}\rceil\} \\
&\leq \quad e^{1/\omega_n} + e^{1/\omega_n}\frac{1}{\log 2\omega_n\lg n} \\
&+ \quad e^{1/\omega_n}\frac{2\lg p}{\lg n\lg k} + \frac{p}{n} \\
&+ \quad \frac{p}{\log 2\omega_n n\lg n} + \frac{2p\lg p}{n\lg n\lg k}
\end{aligned}
$$

$$+ \quad \frac{p}{n\log n} + \frac{1}{e^{1/\omega_n}\log n}.$$

**Proof:** This lemma follows the steps of the Proof of the previous one. For $L \leq n/p$ the two maxima are defined, and if we use $(1+1/(\omega_n \frac{\lg p}{\lg k}))^{\lg p/\lg k} \leq e^{1/\omega_n}$ the claim follows as in Lemma 5. ■

**Theorem 26** *There exists a BSP sorting algorithm that for any integer $p$, $n$ and $L$ such that $p \geq 2$, $n \geq 3500$, $p^3 \leq n/\lg^2 n$, $L \leq n/p$, and $g \leq \lg n$ has with probability $1-o(1)$:*

$$\begin{aligned}
\pi &\leq 1 + \frac{1}{\sqrt{\log n}} + \frac{2\log 2 + 0.741}{\log n} + \frac{4pL}{n\lg n} \\
&+ \frac{2.765}{\log^{1.5} n} + \frac{0.8\log^{2/3} n}{n^{1/3}} + \frac{1.33532}{n^{2/3}\lg^{2/3} n} + \frac{1.8\log^2 n}{n^{2/3}\lg^{5/3} n} \\
&\leq 1 + \frac{1}{\sqrt{\log n}} + \frac{7.7}{\log n},
\end{aligned}$$

*and*

$$\begin{aligned}
\mu &\leq \frac{g}{\lg n}\left(1 + \frac{1}{\sqrt{\log n}}\right) + \frac{5pL}{n\lg n} + \frac{3}{\lg^2 n} + \frac{1.105\log^{2/3} n}{n^{1/3}} + \frac{p}{n} + \frac{2ps}{n} \\
&\leq \frac{g}{\lg n}\left(1 + \frac{1}{\sqrt{\log n}}\right) + \frac{7.8}{\log n}.
\end{aligned}$$

**Proof:** If $N = n$, $k = p$, $s = 1.8\log^2 n$, and $\epsilon^{-1} = \omega_n = \sqrt{\log n}$, then by Remark 1 the conditions of Claim 10 can be satisfied and hence Proposition 15 applies. The expression for $\pi$ in Proposition 15 simplifies according to Lemma 5. Substituting $l = 1$, $\omega_n = \sqrt{\log n}$ and using the assumed bounds on the variables yield the claimed result. Following these steps we first get:

$$\begin{aligned}
\pi &\leq \frac{p\lg p}{n\lg n\lg k}\max\{L, \lceil (1+\tfrac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} \tfrac{n}{p}\rceil (\lceil \lg k\rceil + 1)\} \\
&+ \frac{p}{n\lg n}\max\{L, \lceil \tfrac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p}\rceil \lg \lceil \tfrac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p}\rceil\}
\end{aligned}$$

$$
\begin{aligned}
&+ \quad \frac{p\lg p}{n\lg n\lg k}\max\{L,(ks-1)\lg(ks-1)\} \\
&+ \quad \frac{p\lg p}{n\lg n\lg k}\max\{L,s\} \\
&+ \quad \frac{p}{n\lg n}\left(3\frac{\lg p}{\lg k}-1\right)l\max\{L,k\lceil p^{1/l}\rceil\} \\
&\leq \quad 1+\frac{1}{\omega_n}+\left(1+\frac{1}{\omega_n}\right)\frac{1}{\log 2\omega_n\lg n}+\left(1+\frac{1}{\omega_n}\right)\frac{2}{\lg n}+\frac{p}{n} \\
&+ \quad \frac{p}{\log 2\omega_n n\lg n}+\frac{2p}{n\lg n}+\frac{1}{\log n(1+1/\omega_n)}+\frac{p}{n\log n} \\
&+ \quad \frac{p}{n\lg n}\max\{L,(1.8p\log^2 n-1)\lg(1.8p\log^2 n-1)\} \\
&+ \quad \frac{p\lg p}{n\lg n\lg k}\max\{L,s\} \\
&+ \quad \frac{2pl}{n\lg n}\max\{L,p\lceil p^{1/l}\rceil\}\ ,
\end{aligned}
$$

where we substituted the values for $k$, $s$.

If we upper bound a quantity of the form $\max\{L,Y\}$ by $L+Y$, and substitute $l=1$, we get the following expression.

$$
\begin{aligned}
\pi \quad &\leq \quad 1+\frac{1}{\omega_n}+\left(1+\frac{1}{\omega_n}\right)\frac{1}{\log 2\omega_n\lg n}+\left(1+\frac{1}{\omega_n}\right)\frac{2}{\lg n}+\frac{p}{n} \\
&+ \quad \frac{p}{\log 2\omega_n n\lg n}+\frac{2p}{n\lg n}+\frac{1}{\log n(1+1/\omega_n)}+\frac{p}{n\log n}+\frac{4pL}{n\lg n} \\
&+ \quad \frac{p}{n\lg n}(1.8p\log^2 n)\lg(1.8p\log^2 n)+\frac{2p^3}{n\lg n}+\frac{ps}{n\lg n}.
\end{aligned}
$$

Since $p^3 \leq n/\lg^2 n$, $n > 3500$,

$$
\frac{p}{n}+\frac{p}{\log 2\omega_n n\lg n}+\frac{2p}{n\lg n}\leq\frac{1.212755}{n^{2/3}\lg^{2/3} n}.
$$

and

$$
\frac{1}{\log n(1+1/\omega_n)}+\frac{p}{n\log n}\leq\frac{0.741}{\log n}+\frac{0.122542}{n^{2/3}\lg^{2/3} n}.
$$

In order to bound the term $\frac{p}{n\lg n}(1.8p\log^2 n)\lg(1.8p\log^2 n)$ we use the inequalities $p^3 \leq$

$n/\lg^2 n$, $n > 3500$, so that

$$\begin{aligned}
\frac{p}{n\lg n}(1.8p\log^2 n)\lg(1.8p\log^2 n) &\le \frac{p}{n\lg n}(1.8p\log^2 n)0.72\lg n \\
&\le \frac{1.8\cdot 0.72\ (\log 2)^{4/3}\log^{2/3} n}{n^{1/3}} \\
&\le \frac{0.8\ \log^{2/3} n}{n^{1/3}}.
\end{aligned}$$

If we now use the inequalities $L \le n/p, p \ge 2$, $p^3 \le n/\lg^2 n$, consider that $n > 3500$, and substitute $\omega_n = \sqrt{\log n}$ we get.

$$\begin{aligned}
\pi &\le 1 + \frac{1}{\sqrt{\log n}} + \left(1 + \frac{1}{\sqrt{\log n}}\right)\frac{1}{\log 2\sqrt{\log n}\lg n} + \left(1 + \frac{1}{\sqrt{\log n}}\right)\frac{2}{\lg n} \\
&+ \frac{4pL}{n\lg n} + \frac{0.741}{\log n} + \frac{2p^3}{n\lg n} + \frac{ps}{n\lg n} + \frac{0.8\ \log^{2/3} n}{n^{1/3}} + \frac{1.33532}{n^{2/3}\lg^{2/3} n}.
\end{aligned}$$

In the derivation of the term before the last one, we used the fact that $n > 3500$. This last inequality can be further simplified, if we take into account that $\lg n = \log n/\log 2$.

$$\begin{aligned}
\pi &\le 1 + \frac{1}{\sqrt{\log n}} + \frac{2}{\lg n} + \frac{1}{\log^2 n} + \frac{1}{\log^{1.5} n} + \frac{2}{\sqrt{\log n}\lg n} \\
&+ \frac{4pL}{n\lg n} + \frac{0.741}{\log n} + \frac{1.8\log^2 n}{n^{2/3}\lg^{5/3} n} + \frac{2}{\lg^3 n} + \frac{0.8\log^{2/3} n}{n^{1/3}} + \frac{1.33532}{n^{2/3}\lg^{2/3} n} \\
&\le 1 + \frac{1}{\sqrt{\log n}} + \frac{2\log 2}{\log n} + \frac{4pL}{n\lg n} + \frac{2.765}{\log^{1.5} n} \\
&+ \frac{0.8\ \log^{2/3} n}{n^{1/3}} + \frac{1.33532}{n^{2/3}\lg^{2/3} n} + \frac{0.741}{\log n} + \frac{1.8\log^2 n}{n^{2/3}\lg^{5/3} n} \\
&\le 1 + \frac{1}{\sqrt{\log n}} + \frac{2\log 2}{\log n} + \frac{4}{\lg n} + \frac{2.765}{\log^{1.5} n} + \frac{0.741}{\log n} + \frac{1.8\log^2 n}{n^{2/3}\lg^{5/3} n} \\
&+ \frac{0.8\ \log^{2/3} n}{n^{1/3}} + \frac{1.33532}{n^{2/3}\lg^{2/3} n} \\
&\le 1 + \frac{1}{\sqrt{\log n}} + \frac{6\log 2}{\log n} + \frac{2.765}{\log^{1.5} n} + \frac{0.741}{\log n} \\
&+ \frac{0.8\ \log^{2/3} n}{n^{1/3}} + \frac{1.33532}{n^{2/3}\lg^{2/3} n} + \frac{1.8\log^2 n}{n^{2/3}\lg^{5/3} n}.
\end{aligned}$$

Similarly, we get for $\mu$, if we substitute in Proposition 15, $k$, $s$,$l$.

$$\begin{aligned}\mu &= g(\frac{p}{n\lg n}\max\{\tfrac{L}{g},(1.8p\log^2 n-1)\} \;+\frac{3p}{n\lg n}\max\{\tfrac{L}{g},p^2\}\\ &+\;\frac{p}{n\lg n}\max\{\tfrac{L}{g},2s\} \;+\frac{p}{n\lg n}\max\{\tfrac{L}{g},\lceil(1+\tfrac{1}{\omega_n})n/p\rceil\}\;).\end{aligned}$$

Since $L/g \leq n/p$, and if we follow the method of bounding the maxima described in the discussion for $\pi$, we get for $g \leq \lg n$, $p^3 \leq n/\lg^2 n$

$$\begin{aligned}\mu &\leq \frac{5pL}{n\lg n}+\frac{pg}{n\lg n}1.8p\log^2 n+\frac{2psg}{n\lg n}+\frac{3pg}{n\lg n}p^2\\ &+ \frac{pg}{n\lg n}(1+(1+\frac{1}{\omega_n})n/p)\\ &\leq \frac{5pL}{n\lg n}+\frac{2psg}{n\lg n}+\frac{3p^3g}{n\lg n}+\frac{pg}{n\lg n}+\frac{g}{\lg n}\left(1+\frac{1}{\omega_n}\right)+\frac{1.105\log^{2/3} n}{n^{1/3}}.\end{aligned}$$

Thus, for $L \leq n/p$, $p^3 \leq n/\lg^2 n$, $g \leq \lg n$, $\omega_n = \log n$, we have

$$\begin{aligned}\mu &\leq \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{5pL}{n\lg n}+\frac{3}{\lg^2 n}+\frac{1.105\log^{2/3} n}{n^{1/3}}+\frac{p}{n}+\frac{2ps}{n}\\ &\leq \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{5}{\lg n}+\frac{3}{\lg^2 n}+\frac{1.105\log^{2/3} n}{n^{1/3}}+\frac{p}{n}+\frac{1}{n^{1/3}\lg^{4/3} n}\\ &\leq \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{7.8}{\log n}.\end{aligned}$$

■

**Theorem 27** *There exists a BSP sorting algorithm that for any integer $p$, $n$ and $L$ such that $p \geq 2$, $n \geq 25000$, $p^2 \leq n/(6\log^3 n)$, $L \leq n/(p\lg p)$, and $g \leq \lg n$, has with probability $1-o(1)$:*

$$\begin{aligned}\pi &\leq 1+\frac{1}{\sqrt{\log n}}+\frac{2\log 2+0.7609}{\log n}+\frac{2pL}{n\lg n}+\frac{2pL\lg p}{n\lg n}+\frac{2.75}{\log^{1.5} n}+\frac{0.22725}{\log n}\\ &\leq 1+\frac{1}{\sqrt{\log n}}+\frac{6.1}{\log n},\end{aligned}$$

*and*

$$
\begin{aligned}
\mu &\leq \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{2pL}{n\lg n}+\frac{3pL\lg p}{n\lg n}+\frac{\lg p}{\log^3 n}+\frac{0.375}{\log n}+\frac{p}{n}+\frac{2ps}{n} \\
&\leq \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{4.30}{\log n}.
\end{aligned}
$$

**Proof:** If $N = n$, $k = p$, $s = 2.25\log^2 n$, $0 < \rho < 1$, and $\epsilon^{-1} = \omega_n = \sqrt{\log n}$, then by Remark 2 $s = 2.25\log^2 n$, the conditions of Claim 10 can be satisfied and hence Proposition 15 applies. The expression for $\pi$ in Proposition 15 simplifies according to Lemma 5. Substituting $l = 1$, $\omega_n = \sqrt{\log n}$ and using the assumed bounds on the variables yield the claimed result. Following these steps we first get.

$$
\begin{aligned}
\pi &\leq \frac{p\lg p}{n\lg n\lg k}\max\{L, \lceil(1+\tfrac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}\tfrac{n}{p}(\rceil\lceil\lg k\rceil+1)\} \\
&+ \frac{p}{n\lg n}\max\{L, \lceil\frac{(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}n}{p}\rceil\lg\lceil\frac{(1+\frac{1}{\omega_n\frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}}n}{p}\rceil\} \\
&+ \frac{p\lg p}{n\lg n\lg k}\max\{L, (ks-1)\lg(ks-1)\} \\
&+ \frac{p\lg p}{n\lg n\lg k}\max\{L, s\} \\
&+ \frac{p}{n\lg n}\left(3\frac{\lg p}{\lg k}-1\right)l\max\{L, k\lceil p^{1/l}\rceil\} \\
&\leq 1+\frac{1}{\omega_n}+\left(1+\frac{1}{\omega_n}\right)\frac{1}{\log 2\omega_n\lg n}+\left(1+\frac{1}{\omega_n}\right)\frac{2}{\lg n}+\frac{p}{n} \\
&+ \frac{p}{\log 2\omega_n n\lg n}+\frac{2p}{n\lg n}+\frac{p}{n\log n}+\frac{1}{\log n(1+1/\omega_n)} \\
&+ \frac{p}{n\lg n}\max\{L, (2.25p\log^2 n-1)\lg(2.25p\log^2 n-1)\} \\
&+ \frac{p}{n\lg n}\max\{L, s\} + \frac{2pl}{n\lg n}\max\{L, p(1+p^{1/l})\}\ ,
\end{aligned}
$$

where we substituted the values for $k$, $s$.

If we upper bound a quantity of the form $\max\{L, Y\}$ by $L+Y$, and substitute $l = \lg p$, we get the following expression.

$$\begin{aligned}
\pi &\le 1+\frac{1}{\omega_n}+\left(1+\frac{1}{\omega_n}\right)\frac{1}{\log 2\omega_n \lg n}+\left(1+\frac{1}{\omega_n}\right)\frac{2}{\lg n}+\frac{p}{n}\\
&+ \frac{p}{\log 2\omega_n n\lg n}+\frac{2p}{n\lg n}+\frac{2pL}{n\lg n}+\frac{2pL\lg p}{n\lg n}+\frac{p}{n\log n}+\frac{1}{\log n(1+1/\omega_n)}\\
&+ \frac{p}{n\lg n}(2.25p\log^2 n)\lg(2.25p\log^2 n)+\frac{4p^2\lg p}{n\lg n}+\frac{p\cdot 2.25\log^2 n}{n\lg n}.
\end{aligned}$$

Since $p^2 \le n/(6\log^3 n)$, $n > 25000$,

$$\frac{p}{n}+\frac{p}{\log 2\omega_n n\lg n}+\frac{2p}{n\lg n}\le\frac{0.5}{n^{1/2}\log^{3/2} n},$$

and

$$\frac{p}{n\log n}+\frac{1}{\log n(1+1/\omega_n)}\le\frac{0.7609}{\log n}+\frac{0.05}{\sqrt{n}\log^{3/2} n}.$$

If we now use the inequalities $L \le n/(p\lg p)$,$p \ge 2$, $p^2 \le n/(6\log^3 n)$, (and for such values $\lg(2.25p\log^2 n) \le 0.606\lg n$) and substitute $\omega_n = \sqrt{\log n}$ we get.

$$\begin{aligned}
\pi &\le 1+\frac{1}{\sqrt{\log n}}+\left(1+\frac{1}{\sqrt{\log n}}\right)\frac{1}{\log 2\sqrt{\log n}\lg n}+\left(1+\frac{1}{\sqrt{\log n}}\right)\frac{2}{\lg n}\\
&+ \frac{2pL}{n\lg n}+\frac{2pL\lg p}{n\lg n}+\frac{4p^2\lg p}{n\lg n}+\frac{p\cdot 2.25\log^2 n}{n\lg n}+\frac{0.22725+0.7609}{\log n}+\frac{0.55}{n^{1/2}\log^{3/2} n}.
\end{aligned}$$

This last inequality can be further simplified, if we take into account that $\lg n = \log n/\log 2$, and $n > 25000$.

$$\begin{aligned}
\pi &\le 1+\frac{1}{\sqrt{\log n}}+\frac{2}{\lg n}+\frac{1}{\log^2 n}+\frac{1}{\log^{1.5} n}+\frac{2}{\sqrt{\log n}\lg n}\\
&+ \frac{2pL}{n\lg n}+\frac{2pL\lg p}{n\lg n}+\frac{1}{3\log^3 n}+\frac{0.98815}{\log n}+\frac{2.25\log^2 n}{\sqrt{6}\sqrt{n}\log^{3/2} n\lg n}+\frac{0.55}{n^{1/2}\log^{3/2} n}\\
&\le 1+\frac{1}{\sqrt{\log n}}+\frac{2\log 2}{\log n}+\frac{2pL}{n\lg n}+\frac{2pL\lg p}{n\lg n}+\frac{2.75}{\log^{1.5} n}+\frac{0.98815}{\log n}\\
&\le 1+\frac{1}{\sqrt{\log n}}+\frac{2\log 2}{\log n}+\frac{2}{\lg n\lg p}+\frac{2}{\lg n}+\frac{0.98815}{\log n}+\frac{2.75}{\log^{1.5} n}
\end{aligned}$$

$$\leq \quad 1 + \frac{1}{\sqrt{\log n}} + \frac{6.1}{\log n}.$$

Similarly, we get for $\mu$, if we substitute in Proposition 15, $k$, $s$, and $l$.

$$\begin{aligned}\mu \quad &= \quad g(\frac{p}{n\lg n}\max\{\tfrac{L}{g},(2.25p\log^2 n-1)\} \;+\frac{3p\lg p}{n\lg n}\max\{\tfrac{L}{g},2p\} \\ &+ \quad \frac{p}{n\lg n}\max\{\tfrac{L}{g},2\cdot 2.25\log^2 n\} \;+\frac{p}{n\lg n}\max\{\tfrac{L}{g},\lceil(1+\tfrac{1}{\omega_n})n/p\rceil\}\;).\end{aligned}$$

Since $L/g \leq n/p$, and if we follow the method of bounding the maxima described in the discussion for $\pi$, we get for $g \leq \lg n$, $p^2 \leq n/(6\log^3 n)$

$$\begin{aligned}\mu \quad &\leq \quad \frac{2pL}{n\lg n}+\frac{3pL\lg p}{n\lg n}+\frac{pg}{n\lg n}2.25p\log^2 n+\frac{6p^2 g\lg p}{n\lg n}+\frac{2\cdot 2.25\log^2 npg}{n\lg n} \\ &+ \quad \frac{pg}{n\lg n}(1+(1+\frac{1}{\omega_n})n/p) \\ &\leq \quad \frac{2pL}{n\lg n}+\frac{3pL\lg p}{n\lg n}+\frac{6p^2\lg pg}{n\lg n}+\frac{2ps}{n}+\frac{pg}{n\lg n}+\frac{g}{\lg n}\left(1+\frac{1}{\omega_n}\right)+\frac{0.375}{\log n}.\end{aligned}$$

Thus, for $L \leq n/(p\lg p)$, $p^2 \leq n/(6\log^3 n)$, $g \leq \lg n$, $\omega_n = \log n$, we have

$$\begin{aligned}\mu \quad &\leq \quad \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{2pL}{n\lg n}+\frac{3pL\lg p}{n\lg n}+\frac{2ps}{n}+\frac{\lg p}{\log^3 n}+\frac{0.375}{\log n}+\frac{p}{n} \\ &\leq \quad \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{2}{\lg p\lg n}+\frac{3}{\lg n}+\frac{2ps}{n}+\frac{\lg p}{\log^3 n}+\frac{0.375}{\log n}+\frac{p}{n} \\ &\leq \quad \frac{g}{\lg n}\left(1+\frac{1}{\sqrt{\log n}}\right)+\frac{4.30}{\log n}.\end{aligned}$$

■

By allowing more than one iteration the algorithm achieves one-optimality for values of $n$ much closer to $p$ than these. For $p = n^{1-t}$ for any constant $t$, a constant number $m$ of iterations, that depends on $t$ suffices. If $m$ is allowed to grow with $n$ then one-optimality can be achieved asymptotically with $n$ exceeding $p$ by only a polylogarithmic factor. The former case can be stated as follows.

**Theorem 28** *For any constant $t$ $(0 < t < 1)$ there is a BSP sorting algorithm that for all* *integers $p$, $n$, and $L$ satisfying $p = n^{1-t}$ and $L = o(n \lg n/p)$ has, with probability $1 - o(1)$, $\pi = 1 + o(1)$, and if $g = o(\lg n)$ also, $\mu = o(1)$.*

**Proof:** Let $l \geq 1$ be a positive integer constant, and $\lambda < 1$ the reciprocal of a positive integer constant, chosen so that $(1-t)(1+\lambda+1/l) < 1$. Let $k = p^{\lambda}$. Let $\omega_n^2 = \log n$, $\epsilon = (\omega_n \log p/\log k)^{-1}$, $0 < \rho < 1$, and $s = 5 \log n \omega_n^2 (1/\lambda^2)$, so that the conditions of Claim 10 are satisfied for large enough $n$. The conditions of Proposition 15 are thus satisfied and the first two terms of $\pi$ there can be simplified according to Lemma 6. A direct substitution into this simplified expression for $\pi$ yields an algorithm with $\pi = 1+o(1)$, provided $L = o(n \lg n/p)$, and $p^{1+\lambda+(1/l)} = o(\lambda n \lg n)$. The last restriction is equivalent to $n^{(1-t)(1+\lambda+1/l)} = o(n \lg n)$, which is true by the choice of $\lambda$, $l$.

If we substitute in the expression for $\mu$, we get that for $g = o(\lg n)$, and $p^{1+\lambda+(1/l)} = o(n)$ it is the case that $\mu = o(1)$. ■

We note that not only can the bounds given above be expressed in a variety of ways, but also the algorithm itself leaves room for several improvements. As stated, it sorts the samples sequentially at one processor. In appropriate circumstances its performance can be improved by any one of several alternative methods. For example the sorting of the samples could be performed by parallel algorithms either by using our algorithm recursively, or some completely different one (e.g. [43]) as we describe below. Also, the randomly chosen samples need not be assembled at one processor, prior to being distributed for parallel sorting. Their random selection could be performed in parallel also (e.g. [45]). Among these possibilities we shall now give one that applies to the case $p = n^{1-t}$, where $t = o(1)$.

We shall make use of an algorithm due to Nassimi and Sahni (see [43]) for sorting $N$ elements using $N^2$ processors. The algorithm can be implemented directly on the BSP, and reduces essentially to a sequence of parallel broadcasting and parallel prefix computations.

Suppose we have an input of size $N$ and $N^2$ processors. We number the processors in a mesh-like fashion, so that we can refer to processor $(i, j)$ $i, j = 1, \ldots, N$. Let the input be $x_1, \ldots, x_N$. Input $x_i$ resides in processor $(1, i)$. Through a segmented parallel prefix computation (as discussed earlier in this work), for each $i = 1, \ldots, N$, $x_i$ is broadcast to all processors of column $i$, that is to processors $(j,i)$, for all $j$. At the next step, for each $i$ processor $(i,i)$ broadcasts to all processors of the corresponding row $i$, the element $x_i$ that it holds, in similar fashion. In one superstep, for all $i$, $j$, processor $(i,j)$ compares $x_i$ and $x_j$. A segmented parallel prefix decides the rank $r(i)$ of each $x_i$ in the input, and this rank is disseminated to all processors of column $i$ via a parallel broadcast. The $r(i)$-th processor of column $i$ routes $x_i$ to processor $(1,r(i))$ (which we can call processor $r(i)$, by renumbering in a row-major fashion). In the BSP we need only route a 1-relation during this superstep. Overall, four segmented parallel prefix computations and a single parallel comparison step are required. Each segmented parallel prefix computation may require the routing of a one-relation so that the elements on which the computation is to be performed are in their proper order.

This variation of Proposition 15, when the sample is sorted by the Nassimi-Sahni algorithm rather than greedily in a single processor, gives rise to the following proposition (derived from Proposition 15) which holds for an input evenly and randomly distributed among the processors.

**Proposition 16** *On the comparison model of computation sorting $n$ items requires $n\lg n$ comparisons. Let $\omega_n$ be any function $\omega_n \to \infty$ as $n \to \infty$. For any integers $p$, $k$, $s$, such that $p$ is an integral power of $k$, $ks \leq \sqrt{p}$, and the conditions of Claim 10 are satisfied with $\epsilon = (\omega_n \lg p/\lg k)^{-1}$, $N = n$ and $\rho > \log(p/k)/\log n$, there is a BSP sorting algorithm that for any integer $l \geq 1$, has, with probability $1 - o(1)$,*

$$
\begin{aligned}
\pi \quad \leq \quad & \frac{p\lg p}{n\lg n\lg k}\max\{L, \lceil (1+\tfrac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} \tfrac{n}{p}\rceil(\lceil \lg k\rceil + 1)\} \\
+ \quad & \frac{p}{n\lg n}\max\{L, \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p}\rceil \lg \lceil \frac{(1+\frac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n}{p}\rceil\} \\
+ \quad & \frac{p\lg p}{n\lg n\lg k}\max\{L, 1\} \\
+ \quad & \frac{10p\lg p}{n\lg n\lg k} l\max\{L, k\lceil p^{1/l}\rceil\}\ ,
\end{aligned}
$$

*and*

$$\begin{aligned}\mu &\leq g(\frac{11p\lg p}{n\lg n\lg k} l \max\{\tfrac{L}{g}, k\lceil p^{1/l}\rceil\} \\ &+ \frac{p\lg p}{n\lg n\lg k}\max\{\tfrac{L}{g}, \lceil(1+\tfrac{1}{\omega_n \frac{\lg p}{\lg k}})^{\frac{\lg p}{\lg k}} n/p\rceil\} + \frac{5p\lg p}{n\lg n\lg k}\max\{\tfrac{L}{g}, 1\}\ ).\end{aligned}$$

*The expression for $\pi$ includes all arithmetic as well as comparison operations.*

**Proof:** The implementation of the Nassimi-Sahni algorithm for sample sorting on the BSP contributes the third term in $\pi$, and a factor in the fourth one, as well as the third term in $\mu$ and a factor in the first term.

We note that the terms that appeared in Proposition 15 relating to the selection of the random sample do not appear here, because of the way sampling is performed. A sample of size $ks$ is selected by having $ks$ processors, among the ones that will initiate the execution of the Nassimi-Sahni algorithm, choose one arbitrary element each among the elements local to each such processor. This suffices since the input elements are assumed randomly distributed among the processors, and this randomness can be preserved throughout the algorithm, if in stage 3, part (i), we assume that the comparisons performed there do not change the relative position of any two elements of the input, so that if the input were randomly distributed it would remain so. ■

**Theorem 29** *There is a BSP sorting algorithm such that for all integers $p$, $n$, and $L$ satisfying $p = O(n/\lg^{a+1} p)$, for some constant $a > 0$, and $L = o(n\lg n\,\mathrm{lglg}p/(p\lg^2 p))$ has, with probability $1 - o(1)$, $\pi = 1 + o(1)$, and if in addition $g = o(\mathrm{lglg}p)$, $\mu = o(1)$.*

**Proof:** In case the input is not evenly and randomly distributed among the processors we can still use Proposition 16 by adding a randomizing preprocessing stage (contributing $o(1)$ to $\mu$) and invoking the result for $n(1+\delta)$ for some $\delta = o(1)$ rather than $n$ to allow for the random initial distribution not being exactly even.

Let us choose $k = p^\lambda$, for $\lambda = a\,\mathrm{lglg}p/\lg p$, where $a > 0$ is a constant. Thus $k = \lg^a p$. Let $\omega_n = \sqrt{\log n}$. We choose $s = 5\omega_n^2 \log n \frac{\lg^2 p}{\lg^2 k} = 5\log^2 n \frac{\lg^2 p}{a^2\,\mathrm{lglg}^2 p}$, $\epsilon = (\omega_n \lg p/\lg k)^{-1}$ and any $\rho$

$(0 < \rho < 1)$. Then the conditions of Claim 10 and subsequently those of Proposition 16 are satisfied for large enough $n$. Lemma 6 then simplifies the first two terms of $\pi$ in Proposition 16. Finally, we choose $l = \lg p$.

Substituting these values in the expression for $\pi$ and $\mu$, we get that in order to have $\pi = 1 + o(1)$, or $\mu = o(1)$, we need to choose $L = o(n \lg n \lg k/(pl \lg p)) = o(n \lg n \,\mathrm{lglg}p/(p \lg^2 p))$. For constant $a$, $\pi = 1 + o(1)$ if in addition $p \lg^{a+2} p/ \,\mathrm{lglg}p = o(n \lg n)$.

If we choose $g = o(\lg k) = o(\mathrm{lglg}p)$ and $p < n/(22 \lg^{a+1} p)$, in addition to the constraint on $L$, then we get $\mu = o(1)$. ■

## 5.12 The BSP model with Augmentations

There are various global operations such as broadcasting, the parallel prefix and combining for concurrent memory accesses, that have been suggested as ones useful to implement in hardware in parallel machines, in addition to the basic routing function. The efficacy of such additions can be explored in the BSP context by adding these extra capabilities to the model and seeing how some basic algorithms would be sped up. While it is known that with sufficient slack only constant factor improvements can be expected it is of interest to consider what happens with small slack. For this purpose we define such augmented BSP models, in each case giving the most optimistic reasonable charging policy for the added function.

A $PPF$-BSP is a BSP computer with the power of performing parallel prefix (see [40]) with an elementary associative operator (such as addition or multiplication or finding the maximum or minimum of integers or reals) on an array of registers, one from each processor. Segmented parallel prefix computations (also defined earlier in this work) will also be supported by the $PPF$-BSP. At most $L/g$ parallel prefix computations (segmented or not) can be completed within a single superstep of length $L$. We then refer to $M_p$, $C_p$, $L_p$, $g_p$, $\mu_p$, $\pi_p$ as the parameters of this variation of the BSP model.

The segmented parallel prefix is convenient for performing matrix operations. For example, a single such operation can form the sums of all the columns, or all the rows. The initial distribution of the matrix elements needed is, however, different in these two cases (i.e. row major versus column major order). Hence if both are to be executed then a routing operation is required before the execution of the second.

A $b$-BSP is a BSP computer that allows broadcasting from any one location to any subset of the processors. $L/g$ such broadcasts can be performed in a superstep of length $L$. The corresponding parameters of such a machine are denoted by $M_b$, $C_b$, $L_b$, $g_b$, $\mu_b$, $\pi_b$.

As we have already noted, a $PPF$-BSP is capable of performing broadcasting through parallel prefix provided a suitable associative operator is included in its instruction set.

A $c$-BSP is a BSP computer that allows concurrent reads and writes to various memory locations. We charge as an $h$-relation a set of requests such that at most $h$ are made from any module, and at most $h$ requests to distinct addresses are made to any one module. In other words concurrent reads or writes of one word are charged exactly as the standard BSP model would charge a single read or write. The parameters of this model will be denoted by $M_b$, $C_b$, $L_b$, $g_b$, $\mu_b$, $\pi_b$.

In general we have not found that these augmented BSP models make dramatic increases in efficiency possible. Instead, we have found that for the problems we have considered one-optimality can be achieved sometimes for smaller values of $n$ as compared to $p$, or with larger values of $L$, than for the standard BSP model. We shall give two examples. For matrix vector multiplication Corollary 1 becomes:

**Corollary 12** *There is a $PPF$-BSP algorithm for matrix vector multiplication that for $L = o(n^2/p)$ and $p = o(n^2)$ has $\pi_p = 1 + o(1)$ and $\mu_p = g(1/2 + o(1))$.*

The adaptation of algorithm PE1 for polynomial evaluation to the $PPF$-BSP is straightforward. We retain the computation superstep of stage 1 of PE1. We then compute in a separate stage, using parallel prefix, $x^i$ for all $i = 0, \ldots, p-1$, and following this we multiply in a third stage $q_i(x) \cdot x^i$, one such multiplication per processor, and sum the partial results using another parallel prefix computation.

We get the following corollary that summarizes these results.

**Corollary 13** *There exists a PPF-BSP algorithm for polynomial evaluation that achieves*

$$\pi_p = 1 + \frac{p \lg p}{n} + \frac{Lp}{n} + \frac{p}{n},$$

*and*

$$\mu_p = g\max\{\tfrac{L \cdot p}{2 \cdot g \cdot n}, \tfrac{p}{n}\}.$$

*If* $L = o(n/p)$ *with* $p \lg p = o(n)$*, then* $\pi_p = 1 + o(1)$*, and if in addition,* $g = o(n/p)$*, then* $\mu_p = o(1)$*.*

**Example 3** *For* $n = 20000$*,* $p = 16$*, the computation time is within* 10% *of the optimal for* $L \leq 120$ *and* $\mu \leq 1$*, for* $g \leq 1200$*.*

For Gauss-Jordan elimination Corollary 5 becomes.

**Corollary 14** *There is a PPF-BSP algorithm for Gauss-Jordan elimination that for* $L = o(n^2/p)$ *and* $p = o(n^2)$ *has* $\pi_p = 1 + o(1)$*. If in addition* $g = o(n/\sqrt{p})$ *then also* $\mu_p = o(1)$*.*

The algorithm for multiplying an $n \times n$ matrix $M$ by a vector $x$ that establishes Corollary 12 is the following. The rows of $M$ and $x$ and the columns of $M$ are each divided into $\sqrt{p}$ parts as evenly as possible. This partitions $M$ into $p$ blocks. Each processor is assigned the job of performing the multiplications that have to be done on the elements of one of these blocks. Assuming that $M$ and $x$ are distributed evenly (but otherwise arbitrarily) among the $p$ and some subset of $\sqrt{p}$ processors, respectively, all the data can be routed to the processors where the necessary multiplications are to be performed, by means of an $(\lceil n/\sqrt{p}\rceil^2 + \lceil n/\sqrt{p}\rceil)$-relation. In one computational superstep one performs all the multiplications, as well as the additions $\sum_j M_{ij}x_j$ ranging over sets of values of $j$ belonging to one matrix block. For this $2n^2/p$ operations in total suffice. Finally in order to sum the $\sqrt{p}$ contributions to each component of the answer a segmented parallel prefix computation is issued, where each segment consists of some set of $\sqrt{p}$ processors assigned to a row of $M$, and is disjoint of other segments of processors. A routing operation may be necessary to place the data where needed by the parallel prefix computation.

The algorithm for Gauss-Jordan elimination described in section 5.7 can be adapted as follows.

In **procedure** *Pivot(i)*.

In stage 1, in the communication part of the stage, we perform a parallel prefix computation, with maximum as the associative operator, on the processors of column $i$. Since only these processors need to participate in this superstep, the register of every other non-participating processor may hold a very small dummy value. The result is then communicated to $proc(a_{ii})$. Overall two communication supersteps of length $L$ are needed.

Step (i) of stage 2 becomes redundant, while step (ii) performs a broadcasting through the use of parallel prefix (as we have explained in the description of a $PPF$-BSP). Processor $proc(a_{ii})$ first communicates the value to be broadcast to the leader and then broadcasting is implemented through the parallel prefix. Overall two communication supersteps of length $L$ are needed.

Procedure *Pivot(i)* requires overall four supersteps of length $L$.

In **procedure** *Communicate(i)*.

In stage 1 a parallel prefix computation that implements broadcasting is issued. Two supersteps are needed for this, just as in step (ii) of stage 2 in $Pivot(i)$.

We replace the last $\lceil \lg \sqrt{p} \rceil$ stages of this procedure, by a single one in which we charge a $n/\sqrt{p}$-relation (for so many independent parallel prefix computations that implement broadcasting through segmented parallel prefix). There need to be performed two segmented prefices but on different processor segments. In one of them the $rowp(i)$ processors broadcast values, while in the other, the $columnp(i)$ broadcast them, as described in section 5.7. In the first broadcast all processors storing a column of the matrix get the same value, while in the second one all processors storing a row of the matrix get the same value. Before the parallel prefix computation is initiated the values held by the $rowp(i)$, $columnp(i)$ processors respectively are communicated to the segment leaders. A one-relation needs to be broadcast. Since the processor segments of the two parallel prefices to be issued are different and only one can be directly implemented we can reduce the other to this one by routing a one-relation at the conclusion of the parallel prefix computation.

Thus we need three more supersteps to complete *Communicate*$(i)$.

In the $PPF$-BSP we then get the following expressions for $C(n)$, and $M(n)$.

$$\begin{aligned} C(n) &= 2n\max\{L, \tfrac{n}{\sqrt{p}}\} \ + \sum_{i=1}^{n} \max\{L, T_i\} \\ &\leq 2n\Big(L + \tfrac{n}{\sqrt{p}}\Big) + \sum_{i=1}^{n} (L + T_i) \\ &\leq \frac{n^3}{p}\left(2\Big(\tfrac{L\cdot p}{n^2} + \tfrac{\sqrt{p}}{n}\Big) + 1 + \frac{4\sqrt{p}}{n} + \frac{L^2p^2}{2n^4}\right) \\ &= \pi\frac{n^3}{p}. \end{aligned}$$

In order to have $\pi = 1 + o(1)$ we must have $p = o(n^2)$, and $L = o(n^2/p)$.

Similarly, the communication time for the elimination of all columns is given by the following equation.

$$\begin{aligned} M(n) &= ng\left(9\max\{\tfrac{L}{g}, 1\} \ + \max\{\tfrac{L}{g}, \tfrac{n}{\sqrt{p}}\} \ + \max\{\tfrac{L}{g}, \tfrac{2n}{\sqrt{p}}\} \ \right) \\ &\leq g\frac{n^3}{p}\left(9\max\{\tfrac{Lp}{gn^2}, \tfrac{p}{n^2}\} \ + \max\{\tfrac{Lp}{gn^2}, \tfrac{\sqrt{p}}{n}\} \ + \max\{\tfrac{Lp}{gn^2}, \tfrac{2\sqrt{p}}{n}\} \ \right) \\ &= \frac{n^3}{p}\mu. \end{aligned}$$

From the latter expression we get that in order to have $\mu = o(1)$ we must have $g = o(n/\sqrt{p})$, and Corollary 14 follows.

## 5.13 Simulations and Hash functions

We examine in this section the universal class of hash functions introduced by Dietzfelbinger and auf der Heide in [15] and show how this class can be used for the simulations described earlier in this chapter of classical models of parallel computation on other ones such as, the

*BSP* and the $S^*PRAM$.

### 5.13.1 Results on Hash Functions

Earlier we showed simulation results under the assumption that truly random hash functions computable in constant time were available. We eliminate this assumption by showing that the class of hash functions introduced in [15] offers a way to perform one-optimally in communication the simulations suggested by Theorem 20, parts (i), (ii), (iv), and if the slack is increased by a logarithmic factor, also (iii). We do not take into consideration the initial set-up time of the hash function. If the execution of a simulation is long enough, then the set-up time will be only a lower order term.

The restatement below of Theorem 20 reflects the results we can prove if we use this class of hash functions.

**Theorem 30** *Let $w_p$ be any function of $p$ such that $w_p \to \infty$ as $p \to \infty$. Then the following amounts of slack are sufficient for simulating any one step of the EPRAM or CPRAM on the BSP or S*PRAM in one-optimal expected time for communication (and constant factor optimal time in local operations if $g = O(1)$):*

$$
\begin{array}{ll}
(i) & (w_p p \ \log p)\text{-}EPRAM \text{ on } p\text{-}BSP, \\
(ii) & (w_p p^2 \ \log p)\text{-}CPRAM \text{ on } p\text{-}BSP, \\
(iii) & (w_p p^2 \ \log p)\text{-}EPRAM \text{ on } p\text{-}S^*PRAM, \\
(iv) & (w_p p^2 \ \log p)\text{-}CPRAM \text{ on } p\text{-}S^*PRAM,
\end{array}
$$

*where the expressions before the hyphens denote the number of processors.*

Each simulation of one machine on another works as follows. We randomly choose an appropriate hash function $h$ from a universal class of hash functions that will randomize memory and distribute memory requests evenly among the $p$ memory components of the *p-BSP* (or the $p$-$S^*PRAM$). Thus, the contents of memory address $m$ will be mapped by this hash function to a module $h(m)$.

The proof of Theorem 20 used the following property of truly random hash functions.

**Property 1** *For any $p$ and $n$ and function $w_n$ such that $w_n \to \infty$ as $n \to \infty$, if $n/p \geq w_n \log n$ then any set of $n$ memory accesses of the simulated machine will be mapped, by a truly random function to the $p$ memory components of the simulating machine, such that with probability $1 - O(1/n)$ no more than $(1+\epsilon)\frac{n}{p}$, where $\epsilon \to 0$ as $n \to \infty$, distinct memory requests are directed to any of the $p$ memory components of the simulating machine.*

We prove here that this Property is possessed by the universal class of hash functions suggested in [15]. Theorem 31 to follow shows Property 1 for this class of hash functions. Theorem 30 can then be proved similarly to Theorem 20.

### 5.13.2 Additional Results from Probability Theory

We shall need Proposition 1 shown by McDiarmid in [13], and proven for the binomial case by Angluin and Valiant in [4].

We will then be able to get the following proposition.

**Proposition 17** *Let the random variables $X_1, \ldots, X_n$ be independent, with $0 \leq X_i \leq z$, $z > 0$, for each $i$. Let $\overline{X} = (1/n)\sum_i X_i$, $\mu = \mathcal{E}[\overline{X}]$, $q = z - \mu$. Then, for $0 < \epsilon < 1$, $\epsilon \cdot \mu < q$:*

$$Prob(\overline{X} \geq (1+\epsilon) \cdot \mu) \leq \exp(-\frac{1}{3 \cdot z} \cdot \epsilon^2 \cdot n \cdot \mu).$$

Proposition 17 can be derived by applying Proposition 1 to the random variables $Y_i$, for $Y_i = X_i/z$.

### 5.13.3 A class of Hash Functions

The definitions to follow are taken from [15]. Let $m$ be a prime, $M = \{0, \ldots, m-1\}$ be the memory space, and let us consider two parameters, $d \geq 1$, and $p \geq 1$. Define:

$$H_p^d := \{h_a \mid a = (a_0, \ldots, a_d) \in M^{d+1}\}$$

and let

$$h_a(x) = \left( \sum_{i=0}^{d} a_i x^i \bmod m \right) \bmod p, \text{ for } x \in M \ .$$

Function $h$ will be chosen uniformly at random from $H_p^d$, and probabilities will be taken with respect to this probability space.

We use the following definition that has previously appeared in [15] as Definition 2.1. Let $S \subseteq M$, be a set of accesses such that $|S| = n$. Let $< p >= \{0, 1, \ldots, p-1\}$.

**Definition 12** *Let $h : M \to < p >$ as defined above.*

*(a) The i-th bucket is $B_i^h = \{x \in S \mid h(x) = i\}$, and let $b_i^h = \left| B_i^h \right|$.*

*(b) Function $h$ is called c-perfect for $S$ if $b_i^h \leq c$ for all $i \in < p >$.*

*(c) If $h$ is chosen at random from some class, we write $B_i$ for the random set $B_i^h$ and $b_i$ for the random variable $b_i^h$.*

We shall also need the following fact from [15] proved in [14].

**Fact 12** *Let $S \subseteq M$, be a set of accesses such that $|S| = n$. Let $h$ be randomly chosen from $H_p^d$. Then*

$$Prob(h \text{ is } d\text{-perfect for } S) \geq 1 - n \cdot (2n/p)^d.$$

The following given in [15] was shown in [37].

**Fact 13 ([37])** *Let $0 < \delta < 1$ be fixed, and let $s = n^{1-\delta}$. Let $h \in H_s^d$ be randomly chosen. Then*

$$Prob(b_i \leq 2n^\delta \text{ for all } i) \geq 1 - O(n^{1-\delta-\delta d/2})$$

*(With high probability, all buckets will have size close to the average $n^\delta$.)*

We now briefly describe the class of hash functions introduced by Dietzfelbinger and Meyer auf der Heide in [15].

**Definition 13** *For primes $r, p \geq 1$, and $d_1, d_2 \geq 1$ define $R(r, p, d_1, d_2)$ to be*

$$\{h : M \to <p>| \;\; h = h[f, g, a_0, \ldots, a_{r-1}] \text{ for some } f \in H_r^{d_1}, g \in H_p^{d_2}, a_0, \ldots, a_{r-1} \in <p> \}$$

*where $h = h[f, g, a_0, \ldots, a_{r-1}]$ is defined as follows.*

$$h(x) = (g(x) + a_{f(x)}) \bmod p, \text{ for } x \in M$$

Two functions $h = h[f, g, a_0, \ldots, a_{r-1}]$, $h' = h[f', g', a'_0, \ldots, a'_{r-1}]$, are considered different if they differ in any of the $f$, $g$, and $a_i$'s. A function $h$ so defined can be set up in time and space $O(d_1 + d_2 + r)$.

Let $S$, the set of memory locations to be hashed, be any subset of $M$ of size $n$. Let $R_S(r, p, d_1, d_2)$ be all functions $h$ in $R(r, p, d_1, d_2)$ such that the $r$ buckets $B_i^f$ that $S$ is split into by $f$ are of size $b_i^f \leq 2n/r$, and $g$ is $d_2$-perfect for $B_i^f$ for all $i \in <r>$. We get the following lemma by changing the requirements of Lemma 4.4 of [15].

**Lemma 7** *Let $p = n/(\tau_n \log n)$, where $\tau_n = o(n^\lambda)$ for some $0 < \lambda < 1$ and $\tau_n \to \infty$ as $n \to \infty$. Let $k \geq 1$ be arbitrary and $0 < \delta < 1$ depends on $\lambda$. Then, for $d_1, d_2 \geq 1$ that depend on $\delta, k$ and large enough we get:*

$$Prob(h \in R_S(n^{1-\delta}, p, d_1, d_2)) \geq 1 - O(n^{-k}).$$

The proof for $p = n$ is given in [15]. A proof of the case $p = o(n)$ which indicates how $d_1$, $d_2$ depend on $k$ follows.

**Proof([15])**: Let $r = n^{1-\delta}$, where $\delta$ is chosen so that $\delta < 1 - \lambda$ and $n^\delta < p$ so that $\log (p/n^\delta)/\log n$ is bounded below by some positive constant. From Fact 13 we get that a randomly chosen $f \in H_r^{d_1}$ satisfies $b_i^f \leq 2 \cdot n^\delta$, for all $i \in <r>$, with probability $1 - O(n^{1-\delta-\delta \cdot d_1/2})$.

This is $1 - O(n^{-k})$ for

$$d_1 \geq \frac{k + 1 - \delta}{\delta/2}.$$

For such an $f$, a randomly chosen $g \in H_p^{d_2}$ will be $d_2$-perfect on all $B_i^f$, $i \in < r >$, with probability

$$1 - O\left(n^{1-\delta} \cdot n^{\delta} \cdot \left(\frac{4 \cdot n^{\delta}}{p}\right)^{d_2}\right) = 1 - O(2 \cdot 4^{d_2} \cdot n^{1+\delta \cdot d_2} \cdot p^{-d_2}).$$

If we substitute $p = n/(\tau_n \log n)$, for the previous bound to be $1 - O(n^{-k})$ we need to have

$$d_2 \geq \frac{k+1}{1 - \delta - \frac{\log 4}{\log n} - \frac{\mathrm{loglog} n}{\log n} - \frac{\log \tau_n}{\log n}},$$

provided that $1-\delta-\frac{\log 4}{\log n}-\frac{\mathrm{loglog} n}{\log n}-\frac{\log \tau_n}{\log n} > 0$. This is possible since for $1-\lambda > 0$ we can always choose a positive constant $\delta$ such that $\delta < 1-\lambda$, and subsequently $1-\delta-\frac{\log 4}{\log n}-\frac{\mathrm{loglog} n}{\log n}-\frac{\log \tau_n}{\log n} > 0$. For constant and large enough such choices of $d_1$, $d_2$ (that depend on $k$ and $\delta$) we derive the desired conclusion. ■

In the discussion to follow hash functions will be chosen from the class $R_S(n^{1-\delta}, p, d_1, d_2)$. We also call this class $R_S$. We are thus ready to prove a result similar to Theorem 4.6, part (a) of [15], but stronger than that for the case of $p$ used here.

**Theorem 31** *Let $S \subseteq M$ fixed, $|S| = n$, $p \leq n/(\tau_n \log n)$ where $\tau_n$ was defined in Lemma 7 $\delta, d_1, d_2$ as defined in Lemma 7, $h$ a member of $R_S = R_S(n^{1-\delta}, p, d_1, d_2)$ that follows the requirements of Lemma 7 and $t$ any number such that $t > 0$. Then there is a function $\rho(n, t, \tau_n) = o(1)$ such that the probability that for some $j \in < p >$ we have $b_j^h \geq (1+\rho)\, n/p$ is at most $p/n^{t+1}$.*

**Proof:** The proof is based on that of Theorem 4.6 of [15]. We use Proposition 17 to derive these new bounds.

Let $r = n^{1-\delta}$, $\delta$ appropriately chosen as in the Proof of Lemma 1, so that $n^{\delta} < p$, $p \leq n/(\tau_n \log n)$. Fix $f \in H_r^{d_1}$, $g \in H_p^{d_2}$ so that $b_i^f \leq 2 \cdot n^{\delta}$, and $g$ is $d_2$-perfect for $B_i^f$ for all $i \in < r >$. Following the terminology of [15] let us fix $j$ and define the random variable $X_i = \mid \{x \in B_i^f \mid g(x) + a_i \equiv j \pmod{s}\} \mid$, for $i \in < r >$. Then $b_j^h = \sum_{i=1}^{r} X_i$. It is also the case that $0 \leq X_i \leq d_2$ since $g$ is $d_2$-perfect on $B_i^f$. The $X_i$ are also independent. We have that

$$\mathcal{E}[X_i] = \frac{\left|B_i^f\right|}{p},$$

and

$$\mu = \mathcal{E}\left[\frac{1}{r}\sum_{i=1}^{r} X_i\right] = \frac{1}{r}\sum_{i=1}^{r}\mathcal{E}\left[X_i\right] = \frac{n}{r\cdot p}.$$

We then apply Proposition 17 and get

$$\begin{aligned}
Prob(b_j^h \geq (1+\rho)\frac{n}{p}) &= Prob(\sum_i X_i \geq (1+\rho)\frac{n}{p}) \\
&= Prob(\frac{1}{r}\sum_i X_i \geq (1+\rho)\frac{n}{p\cdot r}) \\
&= Prob(\frac{1}{r}\sum_i X_i \geq (1+\rho)\mu) \\
&\leq \exp(-\frac{1}{3\cdot d_2}\cdot\rho^2\cdot r\cdot\mu) \\
&= \exp(-\frac{1}{3\cdot d_2}\cdot\rho^2\cdot r\cdot\frac{n}{r\cdot p}) \\
&= \exp(-\frac{1}{3\cdot d_2}\cdot\rho^2\cdot\frac{n}{p})
\end{aligned}$$

Since $n/p \geq \log n \cdot \tau_n$, where $\tau_n$ is a function tending to infinity for $n \to \infty$, we get

$$Prob(b_j^h \geq (1+\rho)\frac{n}{p}) \leq n^{-\frac{1}{3\cdot d_2}\cdot\rho^2\cdot\tau_n}$$

Thus for any $t > 0$ and $\rho = \sqrt{\frac{3\cdot(t+1)\cdot d_2}{\tau_n}}$ we get

$$Prob(b_j^h \geq (1+\rho)\ \frac{n}{p}) \leq n^{-(t+1)}.$$

There are $p$ such buckets and $p \leq n$. Thus the probability that any of them is of size at least $(1+\rho)\ \frac{n}{p}$ is at most $p/n^{t+1}$, as desired. ■

We can also strengthen the way we interpret the simulation results as follows.

If the size of the memory used by a simulation is polynomial in $n$ say, $n^{t+1}$, then one step of the simulation will require at most $(1+\rho)\ n/p$ global memory accesses (per processor). In case more accesses are required by some processor, reorganization of the memory (rehashing) occurs that is, at most $n^{t+1}/p$ global communication accesses per processor will be required. Since the failure probability of Theorem 31 is at most $p/n^{t+1}$, the expected time per simulation step, when global communication accesses are only considered, is $(1+\rho)\ n/p+1 = (1+\rho')n/p = (1+o(1))n/p$, for $\rho' = \rho + p/n$.

Thus, on the average, under the assumption that the memory requirements of a problem and thus of the memory that needs to be rehashed are polynomial in the number of processors of the simulated machine reorganization is not prohibitively expensive that is, one-optimality in communication can still be claimed.

### 5.13.4 Concluding Remarks

For a hash function randomly chosen from $R_S$, the results of the previous section imply that that Property 1 holds with high probability.

These results can be summarized in the following two Remarks. When we define a function $w_p$ to be "slowly growing" we mean it is a polylogarithmic function of $n$. The universal class of hash functions addressed in these Remarks refers to the one of Definition 13 which was also claimed in the previous section.

**Remark 3** *Let $n = w_p p \log p$, for some slowly growing $w_p \to \infty$ as $n \to \infty$. Then with probability at least $1 - 1/n^t$, for some $t > 0$, any hash function $h$ chosen from some universal class of hash functions will map no more than $(1+\rho)n/p$, for some $\rho = o(1)$, distinct memory accesses to any one of $p$ memory components.*

**Proof:** For such $n$, let in Lemma 7, $\delta$ be a constant strictly less than one, and let $k$ be chosen such that $k > t + 1$. Then the probability of choosing an $h$ as given by Lemma 7 is asymptotically $1 - O(1/n^k) > 1 - 1/(2n^t)$.

The probability indicated in Theorem 31 of failing to have a mapping of the accesses as indicated in the statement of Remark 3 is thus, asymptotically for large $n$, at most $p/n^{t+1} < 1/(2n^t)$ and thus the remark follows by subadditivity. ■

**Remark 4** *Let $n = w_p p^2 \log p$, for some slowly growing $w_p \to \infty$ as $n \to \infty$. Then with probability at least $1 - 1/n^t$, for some $t > 0$, any hash function $h$ chosen from some universal class of hash functions will map no more than $(1+\rho)n/p$, for some $\rho = o(1)$, distinct memory accesses to any one of $p$ memory components.*

**Proof:** For such $n$, let in Lemma 7, $\delta$ be a constant strictly less than 1/2 so that $\log(p^\delta/n)/\log n$ is bounded below by a positive constant, and let $k > t+1$. Then the probability of choosing an $h$ as given by Lemma 7 is asymptotically $1 - O(1/n^k) > 1 - 1/(2n^t)$.

The probability indicated in Theorem 31 of failing to have a mapping of the accesses as indicated in the statement of Remark 4 is thus, asymptotically for large $n$, at most $p/n^{t+1} < 1/(2n^t)$ and thus the remark follows by subadditivity. ■

In both cases the setup of the hash function requires time proportional to $n^{1-\delta}$. For both Remarks this is $\omega(n/p)$. Thus the claims of optimality of the simulations ignore the setup time of the hash function. We can alternately remove this assumption for simulations requiring $\omega(p/n^\delta)$ steps on the simulated machine. For such cases, the set-up of the hash function requires time $o(1)$ of the total simulation time.

We presented in this section an application of the class of hash functions of [15] to the simulations given earlier in this chapter. In real-life applications the set-up of the hash function can be incorporated in the compiled code which will be the result of the translation of say a *CPRAM* program to the *BSP*. If this is the case, special pipelining hardware may also be used to efficiently hash memory addresses. Since the hashing polynomials are of constant degree, pipelining may be advantageous even for a few evaluations of the hashing polynomials per simulation step.

## 5.14 Conclusions

We examined in this chapter the bulk-synchronous parallel model. We suggested some simulations of PRAMs on it and on a model of optical computing. It is interesting to find simulations more efficient than the ones presented, especially in computation time and in the complexity of the hash functions. With respect to communication time, only the simulation of the $EPRAM$ on the $S^*PRAM$ leaves room for improvements.

We also presented direct BSP algorithms for important computational problems. An interesting question is how well these algorithms perform in practice for the slack claimed for each such algorithm. Another question is whether the slack claimed for each problem in order to get an optimal algorithm can be improved.

# Bibliography


[1] Miklós Ajtai, János Komlós, and Endre Szemerédi. The longest path in a random graph. *Combinatorica*, 1:1–12, 1981.

[2] R. Aleliunas. Randomized parallel computation. In *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 60–72, August 1982.

[3] Richard J. Anderson and Gary L. Miller. Optical communication for pointer based algorithms. *Technical Report CRI 88-14*, 1988.

[4] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *J. Computer and System Sciences*, 18:155–193, 1979.

[5] Reinhold Baer. Null systems in projective spaces. *Bulletin of the American Mathematical Society*, pages 903–906, 1945.

[6] Reinhold Baer. Polarities in finite projective planes. *Bulletin of the American Mathematical Society*, pages 77–93, February 1946.

[7] O. E. Barndorff-Nielsen and D. R. Cox. *Asymptotic Techniques for Use in Statistics.* Chapman and Hall, 1989.

[8] C. Berge. *Graphs and Hypergraphs.* North Holland, 1973.

[9] P. Billingsley. *Probability and Measure.* Wiley, New York, 1986.

[10] Béla Bollobás. *Random graphs.* Academic Press, New York 1985.

[11] A. Borodin and I. Munro. *The Computational Complexity of Algebraic and Numerical Problems.* American Elsevier, New York, 1975.

[12] S. C. Chen. *Speedup of iterative programs in multiprocessor systems.* Rep. No 75-694. PhD Thesis, Department of Computer Science, Univeristy of Illinois, Urbana, 1975.

[13] Colin McDiarmid. On the method of bounded differences. *Surveys in Combinatorics, 1989*, 141:148–188, 1989.

[14] Martin Dietzfelbinger, A. Karlin, K. Mehlhorn, Friedhelm Meyer auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *I.E.E.E. Symposium on the Foundations of Computer Science*, pages 524–531, 1988.

[15] Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. A new universal class of hash functions and dynamic hashing in real time. *International Colloqium on Automata, Languages and Programming*, 443:6–19, Springer Verlag, 1990.

[16] P. Erdős, A. Rényi, and V. T. Sós. On a problem of graph theory. *Studia Scientarium Mathematicarum Hungarica*, 1, 1966.

[17] William Feller. *An Introduction to Probability Theory and its Applications*, Volume 2. Wiley, New York, 1971.

[18] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker. *Solving Problems on Concurrent Processors, Vol 1: General Techniques and Regular Problems.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[19] W. D. Frazer and A. C. McKellar. Samplesort: A sampling approach to minimal storage tree sorting. *Journal of the ACM*, 17:3:496–507, 1970.

[20] A. V. Gerbessiotis. A graph-theoretic result for a model of neural computation. *Submitted for publication*, June 1991.

[21] A. V. Gerbessiotis. Broadcasting in random graphs. Discrete Applied Mathematics (to appear).

[22] A. V. Gerbessiotis. Close-to-Optimal and Near-Optimal Broadcasting in random graphs. *To be submitted for publication*, May 1992.

[23] A. V. Gerbessiotis. Connectivity properties of $G_{n,1/(kn)^{1/2}}$ random graphs related to a model of Neural Computation. *To be Submitted for publication*, June 1992.

[24] A. V. Gerbessiotis. Iterative properties of $G_{n,1/(kn)^{1/2}}$ random graphs related to a model of Neural Computation. *To be Submitted for publication*, September 1992.

[25] A. V. Gerbessiotis. New results for the Bulk-Synchronous Parallel Model of computation. *To be Submitted for publication*, November 1992.

[26] A. V. Gerbessiotis and L. G. Valiant. *Direct bulk-synchronous parallel algorithms.* Technical Report, Aiken Computation Laboratory, Harvard University, 1992. Also in *Proceedings of Third Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science, Volume 621, July 1992, Springer-Verlag.

[27] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The John-Hopkins University Press, Baltimore, MD, 1989.

[28] Marshall Hall Jr. *Combinatorial Theory.* Wiley, New York, 1986.

[29] Alfred Hartmann and Steve Redfield. Design sketches for optical crossbar switches intended for large scale parallel processing applications. *Optical Engineering*, 29:3:315–327, April 1989.

[30] L. Hyafil and H. T. Kung. The complexity of parallel evaluation of linear recurrence. *Proceedings 7th ACM Symposium on Theory of Computing*, New Mexico, 1975.

[31] C. A. R. Hoare. Quicksort. *The Computer Journal*, 5:10–15, 1962.

[32] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statist. Association*, 58, 13-30, 1963.

[33] J. S. Huang and Y. C. Chow. Parallel sorting and data partitioning by sampling. *IEEE Computer Society's Seventh International Computer Software and Applications Conference*, pages 627–631, November 1983.

[34] Daniel R. Hughes and Fred C. Piper. *Projective Planes*, Volume 6. Springer-Verlag, 1973.

[35] D. E. Knuth. *The Art of Computer Programming: Fundamental Algorithms*. Addison-Wesley Publishing Company, New York, 1973.

[36] V. F. Kolchin, B. A. Sevastyanov, and V. P. Chistyakov. *Random Allocations*. Translation ed. A.V. Balakrishnan, Halsted Press, 1978.

[37] Clyde P. Kruskal, Larry Rudolph, and Marc Snir. A complexity theory of efficient parallel algorithms. *Theoretical Computer Science*, 71:95–132, 1990.

[38] D. J. Kuck. Parallel Processing of Ordinary Programs. In *Advances in Computers*, Vol. 15, pp119-179, 1976.

[39] H. T. Kung. New Algorithms and Lower Bounds for the Parallel Evaluation of Certain Rational Expressions and Recurrences. *Journal of the ACM*, Vol. 23, pp252-261, 1976.

[40] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*. Morgan Kaufmann Publishers, San Mateo, California, 1992.

[41] Richard A. Linke. Power distribution in a planar-waveguide based broadcast star network. *IEEE Photonics Technology Letters*, 3:9:850–852, September 1991.

[42] Eric S. Maniloff, Kristina M. Johnson, and J. H. Reif. Holographic routing network for parallel processing machines. In *Holographic Optics II: Principles and Applications, Proc. SPIE* (G. M. Morris, ed.), 1136:283–289, 1989.

[43] D. Nassimi and S. Sahni. Parallel permutation and sorting algorithms and a new generalized connection network. *Journal of the ACM*, 29:3:642:667, July 1982.

[44] Alfred Rényi. *Probability Theory*. Akademiai Kiado, Budapest.

[45] J. H. Reif and L. G. Valiant. A logarithmic time sort for linear size networks. *Journal of the ACM*, 34:60–76, January 1987.

[46] Edward R. Scheinerman and John C. Wierman. Optimal and near-optimal broadcast in random graphs. *Discrete Applied Mathematics*, 25:289–297, 1989.

[47] H. S. Stone. An efficient parallel algorithm for the solution of a tridiagonal linear system of equations. *Journal of ACM*, 20:27–38, 1973.

[48] E. Upfal. Efficient schemes for parallel communication. In *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 55–59, August 1982.

[49] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[50] L. G. Valiant. Functionality in neural nets. *Proceedings of the American Association for Artificial Intelligence*, Volume 2, 1988. Also in monograph *Neuroids*, manuscript, 1992.

[51] L. G. Valiant. General purpose parallel architectures. In *Handbook of Theoretical Computer Science* (J. van Leeuwen, ed.), North Holland, pages 945-971, 1990.

[52] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33:103–111, August 1990.

[53] M. H. van Emden. Increasing the efficiency of quicksort. *Communications of the ACM*, 13:9:563–567, 1970.