Alcom-FT Technical Report Series
ALCOMFT-TR-01-66

Chapter 1

RANDOMIZED TECHNIQUES FOR MODELLING FAULTS AND ACHIEVING ROBUST COMPUTING

Sotiris E. Nikoletseas
Computer Technology Institute and
Patras University, Greece
nikole@cti.gr

Paul G. Spirakis
Computer Technology Institute and
Patras University, Greece
spirakis@cti.gr

Abstract Computing in the presence of faults is a challenging problem. Modelling of faults in communication networks leads to graph models where some edges (or vertices) may be missing with a certain probability. Faults in computations can be modeled by stochastic patterns of unavailability of some processing power. In this chapter we discuss various structural and simulation results of the area of robust computations, which make heavy use of randomization techniques.

Introduction

Graphs are simple models of communication networks. Each edge (that is a communication line) may have a certain probability of failing. In the simplest case it is assumed that each communication line transmits information both ways and that the lines fail independently and with a common probability. Thus one is lead to various random graph models. For example, the $G_{n,p}$ model $(0 < p < 1)$ consists of all graphs of n vertices in which the edges are chosen independently and with probability p. Another useful model for structured networks is the G_n^r, i.e. the class of all regular graphs of n vertices and degree r $(0 < r < n)$. We can turn this into a probability space by giving all the elements of the class the same probability. If, in addition, we remove each edge of a random element of G_n^r with probability $1-p$, then we have the model of random regular graphs with edge faults, denoted by $G_{n,p}^r$. For an excellent summary of work on these models see the book "Random Graphs" of B. Bollobás ([10]).

Perhaps the most celebrated theorem in Random Graph Theory is that of the Giant Component: Erdös and Renyi ([26]) first showed that $G_{n,p}$ has a giant (i.e. $\Theta(n)$-sized) connected component when, roughly, $p > \frac{1}{n}$. In fact, many of the results in Random Graphs were inspired by the work of Uncle Paul Erdös.

Advances in computer technology made parallel and distributed machines a reality. It is a challenge to design such systems, or protocols for such systems, that compute correctly and efficiently in the presence of faults. The difficulty associated with combining fault-tolerance with efficiency is that the two have conflicting means, as observed by Kanellakis and Shvartsman in [44]: fault-tolerance is achieved by introducing redundancy, while efficiency is achieved by removing redundancy. It is interesting that randomized techniques can help to achieve both means (or to sharply express their tradeoff).

In this chapter, we take a partial view of the models of random faults and randomized robustness techniques. In the first half, we examine structural properties of random graphs and random regular graphs with edge faults, such as connectivity, expansion, dominance and trade-offs between density and robustness. In the second half, we survey recent work on robust parallel and distributed computing and present an interesting case study. The writing is biased by the work of the authors and aims at motivating further work in this fascinating field where randomization techniques are essential for both modeling and analysis.

1. BASIC PROBABILISTIC TECHNIQUES

1.1 INTRODUCTION

The common, unifying concept of the various techniques comprising the Probabilistic Method is the non-constructive proof of the existence of combinatorial structures satisfying certain desired properties.

The basic probabilistic technique, which we call the method of positive probability, can be described in the following way: in order to prove the existence of a combinatorial structure satisfying certain desired properties, we first construct (by an abstract random experiment) an appropriate probability space, whose sample points correspond to the combinatorial structures under examination. Then we show that the desired property is satisfied in this space with positive (non-zero) probability. This implies, by the axiomatic definition of the concept of probability, the existence of at least one point in the space (that is, a corresponding combinatorial structure) with the desired property.

In this section, we will present some basic probabilistic techniques. In each case, we will also provide a characteristic example from recent research. More specifically:

1. we will present the use of Markov's inequality in proving non-existence results. As an example, we will show that in dense random graphs there do not exist dominating sets of size less than $\ln n$, almost certainly.

2. we will then employ the famous second moment method in order to prove that dominating sets of size $\lceil \log n \rceil$ exist in dense random graphs, with high probability.

3. we will present Chernoff bounds, a strong probability-theoretic tool for bounding the large deviation probability of a sum of independent random variables from its mean, widely used in the analysis of randomized algorithms. As an example, we provide an efficient randomized algorithm for finding near-optimal dominating sets in dense random graphs.

4. we also present the notion of random configurations and use it in order to prove various connectivity properties (and also, by a BFS approach, to prove the existence of a giant component) in random regular graphs with edge faults.

1.2 FIRST MOMENT ARGUMENTS FOR NON-EXISTENCE PROOFS

We will use the following important inequality:

Theorem 1 (Markov's inequality) *Let X any non-negative random variable. For every $t > 0$,*

$$\Pr\{X \geq t\} \leq \frac{E(X)}{t}$$

$\square$

As an immediate consequence of the above theorem we get that:

Corollary 1 *For any non-negative, integer-valued random variable $X(n)$,*

$$\lim_{n\to\infty} E[X(n)] = 0 \quad \Rightarrow \quad \lim_{n\to\infty} \Pr\{X(n) = 0\} = 1$$

$\square$

Thus, in order to show the almost certain non-existence of desired combinatorial structures, it suffices to prove that the mean of a random variable X counting their number tends to 0.

1.3 LOWER BOUNDS FOR DOMINATING SETS IN DENSE RANDOM GRAPHS

The dominating set problem is the following: given a graph $G(V, E)$ and a positive integer $k < |V|$, is there a subset $V' \subseteq V$ (called a dominating set) with $|V'| \leq k$ such that for all u in $V - V'$ there is a v in V' for which $\{u, v\} \in E$? The problem is NP-Complete (see [34]), even for planar graphs with maximum degree 3. In [60], Nikoletseas and Spirakis have studied the "average case" complexity of the problem, following the line of research of [25]. More specifically, they have examined the existence and efficient finding of small dominating sets in dense random graphs. They show, for the model $G_{n,p}$ with $p = 1/2$, that:

1. The probability of existence of dominating sets of size less than $\ln n$ tends to zero as n tends to infinity.

2. Dominating sets of size $\lceil \log n \rceil$ exist almost surely.

3. They provide two algorithms which construct small dominating sets in $G_{n,1/2}$ and run in $O(n \log n)$ time (on the average and also with high probability). These algorithms almost surely construct a dominating set of size at most $(1 + \epsilon) \ln n$, for any fixed $\epsilon > 0$.

These results extend easily to the case $G_{n,p}$ with p fixed to any constant < 1.

Recently, Gkantsidis, Hatzis, Nikoletseas and Spirakis ([39]) have studied the same problem in the (technically much more difficult) case of sparse random graphs, providing both lower bounds and algorithms.

In this section we prove the following result.

Theorem 2 *For any $k < \ln n$ the probability of existence of a dominating set of size k in $G_{n,1/2}$ tends to 0 as n tends to infinity.*

Proof: Let us consider a $G_{n,1/2}$ graph and let S be any fixed set of k vertices of this graph. Because of the stochastic independence characterizing the $G_{n,p}$ model, the mean of the random indicator variable

$$X_S = \begin{cases} 1 & \text{if } S \text{ is dominating} \\ 0 & \text{otherwise} \end{cases}$$

is equal to

$$E(X_S) = \left(1 - \frac{1}{2^k}\right)^{n-k}$$

since the probability that a certain vertice out of S connects to at least one vertice in it is $\left(1 - \frac{1}{2^k}\right)$ and this must happen for all $n - k$ vertices out of S. The random variable

$$X = \sum_{S, |S|=k} X_S$$

counts the number of dominating sets of size k and, because of linearity of expectation, its mean is

$$E(X) = \sum_{S,|S|=k} E(X_S) = \binom{n}{k} \left(1 - \frac{1}{2^k}\right)^{n-k}$$

But $\binom{n}{k} \leq n^k$ and $1 - \frac{1}{2^k} \leq e^{-\frac{1}{2^k}}$ so that

$$E(X) \leq e^{\frac{k}{2^k}} \left(e^{k \ln n - \frac{n}{2^k}}\right)$$

We remark that

$$k = \frac{\ln n}{c} \text{ where } c > 1 \Rightarrow F = k \ln n - \frac{n}{2^k} \to -\infty$$

that is the exponent F tends to infinity when $k < \ln n$ and finally

$$k < \ln n \;\Rightarrow\; E(X) \to 0$$

In other words, the expected value of the number of the dominating sets of size less than $\ln n$ tends to zero, thus there do not exist such dominating sets, almost certainly. □

1.4 EXISTENCE RESULTS WITH THE SECOND MOMENT METHOD

The second moment method is a powerful and quite general technique widely used in proving the almost certain existence of combinatorial structures satisfying desired properties. For a complete presentation of the second moment method we suggest the excellent books of J. Spencer ([69]) and N. Alon, J. Spencer ([1]). These references also contain characteristic applications of this method, such as Paul Turán's proof on the prime factors of a number, results on distinct sums of sets of positive integers, concentration results on the clique number of dense random graphs and also thresholds for the existence of balanced subgraphs in $G_{n,p}$ random graphs.

From the probability-theoretic point of view, the second moment method is based on Chebyshev's inequality for bounding the deviations (in terms of the variance) of the values of a random variable from its mean.

Theorem 3 (Chebyshev's inequality) *For any random variable X with mean μ and variance σ^2, it is*

$$\Pr\{|X - \mu| \geq \lambda\sigma\} \leq \frac{1}{\lambda^2}$$

where $\lambda > 0$ any constant. □

Chebyshev's inequality is much stronger compared to Markov's inequality and best possible when only the mean and the variance of the random variable are known. This fact is the probabilistic-theoretic reason for why second moment arguments are much stronger than first moment ones (though technically more difficult to apply, since we must calculate the variance, whose linearity needs stochastic independence). Still, we note that there are techniques and inequalities (e.g. the Local Lemma, Janson's inequality, the Martingale technique) achieving (in the case when, roughly speaking, the stochastic dependencies are small) exponentially small bounds (compared to the only polynomial ones provided by Chebyshev's inequality) on this large deviation probability, and thus are much stronger (though less general) than the second moment method.

We now give the technical theorem the second moment method is based on:

Theorem 4 (Second Moment Method) *For any random variable X if*

$$E(X) \to \infty \quad \textit{and} \quad Var(X) = o(E^2(X))$$

then

$$\Pr\{X = 0\} \to 0$$

(*the limits taken with respect to an independent variable n, that is $X = X(n)$*).
□

Thus, in order to deliver existence proofs using the second moment method, we must not only show that the expected value of the random variable X counting the number of the combinatorial structures under examination tends to infinity, but also that its variance grows asymptotically slower than the square of its mean.

The random variable X can be usually expressed as a sum of random indicator variables, which are in most cases not independent. We employ the notion of the covariance of two random variables, in order to capture these stochastic dependencies:

Definition 1 *The covariance of two random variables X, Y is given by*

$$Cov(X, Y) = E(XY) - E(X)\,E(Y)$$

From the above definition we remark that the covariance of two variables provides a measure of their stochastic dependencies, taking value zero in the case of independent variables.

We can now express the variance of the sum X of the random indicator variables mentioned earlier in terms of the covariances of the (ordered) pairs of these variables.

Theorem 5 *Let* $X = X_1 + \cdots + X_n$*, where the variances* $Var(X_i)$ *of the random indicator variables* $X_i, 1 \leq i \leq n$ *exist and are finite. Then*

$$Var(X) = \sum_{1 \leq i,j \leq n} Cov(X_i, X_j) \tag{1.1}$$

□

As an immediate consequence, we get the following theorem, providing an upper bound for the variance in terms of the expected value.

Theorem 6 *If* $X = \sum_i X_i$*, where* $X_i, 1 \leq i \leq n$*, are random indicator variables, then*

$$\begin{aligned} Var(X) &= \sum_{i=1}^{n} Var(X_i) + \sum_{1 \leq i \neq j \leq n} Cov(X_i, X_j) \\ &\leq E(X) + \sum_{1 \leq i \neq j \leq n} Cov(X_i, X_j) \end{aligned} \tag{1.2}$$

□

We provide here two techniques (presented by N. Alon and J. Spencer in [1, 69]) in order to further bound the variance from above):

- using quantities Δ or Δ^*
- using quantity $f(S, T)$

These techniques are essentially the same and only suggest different ways to carry out calculations. Still, the first one might be considered as more general, since it deals with quantities Δ, Δ^* that are also used in other probabilistic techniques (e.g. Janson inequality).

Let us now define the following important quantity Δ: Let A_i, A_j the events corresponding to the random indicator variables X_i, X_j and the relation $i \sim j$, holding when $i \neq j$ and the events A_i, A_j are stochastically dependent. Then

$$\Delta = \sum_{i \sim j} \Pr\{A_i \wedge A_j\}$$

Since

$$Cov(X_i, X_j) \leq E(X_i X_j) = \Pr\{A_i \wedge A_j\}$$

we get, from inequality (1.2), that

$$Var(X) \leq E(X) + \Delta$$

and finally the basic theorem of the second moment method becomes:

Theorem 7

$$E(X) \to \infty \quad \textit{and} \quad \Delta = o(E^2(X)) \quad \Rightarrow \quad \Pr\{X = 0\} \to 0$$

$\square$

We also note the following useful variation of the above theorem: we call the events A_i, A_j symmetric, if

$$\Pr\{X_j | X_i = 1\} = \Pr\{X_i | X_j = 1\}$$

Note that in most cases the above condition is satisfied, especially when studying graph-theoretic functions where the events A_i, A_j concern properties of subgraphs where "symmetry" appears naturally, in the sense that order does not matter and we can, without loss of generality, assume any of the two events as the event which we condition on.

In the case of symmetric events, let

$$\Delta^* = \sum_{j \sim i} \Pr\{A_j | A_i\}$$

Then, by using the definition of conditional probability, we get

$$\Delta = \Delta^* \, E(X)$$

and finally

Theorem 8

$$E(X) \to \infty \quad \textit{and} \quad \Delta^* = o(E(X)) \quad \Rightarrow \quad \Pr\{X = 0\} \to 0$$

$\square$

We now provide the second alternative way to carry out technical calculations in order to bound the variance from above (see also J. Spencer's presentation in [69]).

Let X be expressed as a sum of random indicator variables $X_1, X_2, \ldots, X_m$, each having mean μ. According to equation (1.1), we get

$$Var(X) = \sum_{S,T} Cov(X_S, X_T)$$

where S, T are the sets corresponding to i, j, respectively. Clearly

$$Cov(X_S, X_T) = \mu^2 f(S, T)$$

where

$$f(S, T) = \frac{E(X_T | X_S = 1)}{E(X_T)} - 1$$

is a quantity measuring the dependency of X_T from X_S (as an example of this, remark that it takes value 0 if X_S, X_T are independent).

Since there are m^2 ways to select ordered pairs from the m variables, we get

$$Var(X) = m^2 \mu^2 E_{S,T}\left[f(S,T)\right]$$

(where $E_{S,T}\left[f(S,T)\right]$ is the expected value of $f(S,T)$ when set S is fixed and set T is randomly chosen among all possible sets) and because $E(X) = m\mu$ we get

$$\frac{Var(X)}{E^2(X)} = E_{S,T}\left[f(S,T)\right]$$

Hence, the basic theorem of this variation of the second moment method becomes:

Theorem 9 *Let X a random variable expressed as a sum of indicator variables. We have*

$$E(X) \to \infty \quad \textit{and} \quad E_{S,T}\left[f(S,T)\right] = o(1) \quad \Rightarrow \quad \Pr\{X = 0\} \to 0$$

□

We note that $E_{S,T}\left[f(S,T)\right]$ depends on the size of intersection $S \cap T$ and is actually a sum of the products of the probabilities of each intersection size times the corresponding $f(S,T)$.

In the next section we present, as an example of the use of the second moment method (and its $f(S,T)$ variation), a result of Nikoletseas and Spirakis ([60]), proving the almost certain existence of dominating sets of size $\lceil \log n \rceil$ in $G_{n,\frac{1}{2}}$ dense random graphs.

1.5 LOGARITHMIC DOMINATING SETS IN DENSE RANDOM GRAPHS

We first remark (see also section 1.3) that the expectation of the number $X^{(k)}$ of dominating sets of size k is:

$$E\left(X^{(k)}\right) = \binom{n}{k}\left(1 - \frac{1}{2^k}\right)^{n-k}$$

Next we easily get the following theorem:

Theorem 10

$$E\left(X^{(\log n)}\right) \to +\infty$$

□

Thus, it is enough to show that $Var\left(X^{(k)}\right) = o\left(E^2\left(X^{(k)}\right)\right)$ when $k = \log n$. By fixing a set S of size $\log n$ and by randomly choosing sets T of the same size, we get for $k = \log n$:

$$\frac{Var\left(X^{(k)}\right)}{E^2\left(X^{(k)}\right)} = \sum_{i=0}^{k} a(i)\, f_i(S,T) \tag{1.3}$$

where:

$$a(i) = \Pr\{|S \cap T| = i\}$$

and

$$f_i(S,T) = f(S,T) = \frac{E(X_T|X_S = 1)}{E(X_T)} - 1$$

when $|S \cap T| = i$ and X_T, X_S are indicator variables for the sets T, S with respect to being dominating sets.

We will now bound appropriately from above the terms of the products in the sum. Clearly:

$$\alpha(i) = \frac{\binom{k}{i}\binom{n-k}{k-i}}{\binom{n}{k}}$$

Lemma 1 *There exists a constant γ and a positive integer n_0 : $\forall n \geq n_0$ and $\forall i = 1, \ldots, \log n$:*

$$f_i(S,T) \leq \gamma$$

Proof: Since X_T is an indicator variable, we get $E(X_T|X_S = 1) \leq 1$. Thus

$$f_i(S,T) \leq \frac{1}{E(X_T)} - 1$$

But

$$E(X_T) = \left(1 - \frac{1}{2^{\log n}}\right)^{n-\log n} \geq \left(1 - \frac{1}{n}\right)^n$$

But $\forall n \geq 2$: $\left(1 - \frac{1}{n}\right)^n \geq 1/4$, thus $E(X_T) \geq 1/4$, hence $f_i(S,T) \leq 3$ and choose $\gamma = 3$. □

Lemma 2 *$\alpha(i)$ is monotone decreasing on i for $i = 1, \ldots, \log n$.*

Proof: Simply remark that

$$\Lambda = \frac{\alpha(i+1)}{\alpha(i)} = \frac{k-i}{i+1}\,\frac{k-i}{n-2k+i+1} \leq \frac{k^2}{n-2k+1}$$

for $i \geq 0$. For $k = \log n$, we get

$$\Lambda \leq \frac{(\log n)^2}{n - 2\log n + 1} < 1 \quad \text{for all} \quad n > 37$$

We note here, since we are investigating asymptotic results for large graphs ($n \to \infty$), that small finite graphs of a few nodes are not of interest and can be in any case processed by brute force. □

Lemma 3 *For* $i = 2, 3, \dots, \lceil \log n \rceil$ *we have*

$$\alpha(i) \leq \frac{4}{n}$$

Proof: From previous lemma we have $\alpha(i) \leq \alpha(2)$. But

$$\alpha(2) = \frac{\binom{k}{2}\binom{n-k}{k-2}}{\binom{n}{k}} \leq k^4 \ \frac{1}{n-k+1} \ \frac{1}{n-k+2}$$

i.e. for $k = \log n$ and sufficiently large n

$$\alpha(2) \leq \frac{(\log n)^4}{(n - \log n + 1)^2} \leq \frac{4}{n}$$

□

We also bound the first term of sequence $a(i)$ from above:

Lemma 4

$$\alpha(1) \leq \frac{(\log n)^2}{n - \log n + 1}$$

Proof:

$$\alpha(1) = \frac{\binom{k}{1}\binom{n-k}{k-1}}{\binom{n}{k}} \leq \frac{k^2}{n-k+1}$$

□

We know get:

Theorem 11 *As* $n \to +\infty$

$$\frac{Var\left(X^{(\log n)}\right)}{E^2\left(X^{(\log n)}\right)} \to 0$$

Proof: By equation (1.3) and previous lemmas we get

$$\begin{aligned}
\frac{Var\left(X^{(\log n)}\right)}{E^2\left(X^{(\log n)}\right)} &\leq \gamma\alpha(1) + \gamma \sum_{i=2}^{\log n} \alpha(i) \\
&\leq \gamma \ \frac{(\log n)^2}{n - \log n + 1} + \gamma \ \frac{4 \log n}{n} \\
&\leq \frac{18 \ (\log n)^2}{n} \to 0
\end{aligned}$$

□

Thus we finally get:

Theorem 12 *There exists a dominating set of size* $\lceil \log n \rceil$ *in* $G_{n,1/2}$ *with probability at least* $1 - 18\ \frac{(\log n)^2}{n}$ *i.e. almost surely.* □

1.6 CHERNOFF BOUNDS

Chernoff bounds ([15]) are strong probability-theoretic inequalities bounding from above the large deviation probability of a sum of independent random variables from its mean, thus proving sharp concentration of the values of such sums around their expected value. These bounds are widely used in the analysis of randomized algorithms, where the random variables of such sums are indicator variables corresponding to Poisson (or Bernoulli, in the case of same probability) trials representing the repetitive execution of independent experiments with two possible outcomes (success, failure). These outcomes correspond to the realization or not of certain events during the repetitive random choices which the algorithm's evolution is based on.

For a detailed presentation of Chernoff bounds see the excellent book of R. Motwani and P. Raghavan on Randomized Algorithms ([58]) and also the detailed survey in [40]. We here provide the following general form of the Chernoff bound for the tail probability of exceeding the mean.

Theorem 13 *Let* $X_1, \ldots, X_n$ *independent random variables corresponding to Poisson trials with success probabilities* $\Pr\{X_i\} = p_i$, $1 \leq i \leq n$. *The probability that their sum* $X = \sum_{i=1}^n X_i$ *exceeds its expected value* $\mu = \sum_{i=1}^n p_i$ *is bounded from above according to the following inequality*

$$\Pr\{X \geq (1+\beta)\mu\} \leq \left(\frac{e^\beta}{(1+\beta)^{1+\beta}} \right)^\mu$$

where $\beta > 0$ *a constant.* □

We also present the following (less general but easier to use and very popular) form, when $\beta \in (0,1)$:

Theorem 14 *For any* $\beta \in [0,1]$ *we have*

$$\Pr\{X \geq (1+\beta)\mu\} \leq e^{-\beta^2\mu/3}$$

□

Next we provide a similar theorem for the tail probability of values below the mean.

Theorem 15 *Let $X_1, \ldots, X_n$ independent variables corresponding to Poisson trials with success probabilities $\Pr\{X_i\} = p_i$, $1 \leq i \leq n$. The probability that their sum $X = \sum_{i=1}^{n} X_i$ takes values below its mean $\mu = \sum_{i=1}^{n} p_i$ is bounded from above according to the following inequality*

$$\Pr\{X \leq (1 - \beta)\mu\} \leq e^{-\beta^2 \mu / 2}$$

where $\beta \in [0, 1]$ a constant. □

Combining the above theorems, we easily get the following bound for the tail probabilities of large deviations from the mean.

Theorem 16 (Chernoff bound) *Let $X_1, X_2, \ldots, X_n$ be independent random variables corresponding to Poisson trials with success probabilities $\Pr\{X_i\} = p_i$, $1 \leq i \leq n$. The probability that their sum $X = \sum_{i=1}^{n} X_i$ is concentrated around its mean $\mu = \sum_{i=1}^{n} p_i$ is at least*

$$\Pr\{X \in (1 \pm \beta)\mu\} \geq 1 - 2e^{-\beta^2 \mu / 3}$$

where $\beta \in [0, 1]$ a constant. □

In the next section, we give an example of the use of Chernoff bounds, presenting the analysis of a fast algorithm for finding near-optimal dominating sets in dense random graphs, with high probability and also on the average.

1.7 A GREEDY ALGORITHM FOR NEAR-OPTIMAL DOMINATING SETS IN DENSE RANDOM GRAPHS

The greedy approach each time chooses a vertex to be put in a dominating set under construction and then it deletes all its neighbors (and the corresponding edges) from the graph, since these vertices are already covered by the set. Note that, because of deleting these inspected edges, the remaining graph is each time still random, since its edges are not exposed. The method achieves dominating sets of size $(1 + \epsilon) \log n$ almost surely and runs in $O(n \log n)$ expected time.

In the following technical description of the algorithm, D is the dominating set under construction, while V_i, E_i are each time the remaining sets of vertices, edges respectively. We also each time note by N_i the neighboring vertices of selected vertex u_i in the graph $G(V_i, E_i)$ and we use set V' to temporarily store the remaining vertex set $V_i - N_i$.

ALGORITHM "GREEDY"

Input: A random graph $G(V,E)$ of $G_{n,1/2}$ and a constant $\epsilon > 0$
(1) $i \leftarrow 0$; $V_i \leftarrow V$; $D \leftarrow \emptyset$
(2) **until** $|V_i| \leq \epsilon \log n$ **do**
  begin
    select a vertex $u_i \in V_i$
    $V' \leftarrow V_i - N_i$
    $D \leftarrow D \cup \{u_i\}$
    $i \leftarrow i+1$
    $V_i \leftarrow V'$
  end
(3) $D \leftarrow D \cup V_i$
(4) **return** D

The basic idea of the Chernoff bound analysis of the algorithm above is the following: while the number of vertices $|V_i|$ in the remaining graph is still at least logarithmic, Chernoff bounds prove that with high probability the selected vertex each time connects with about half of the remaining, and thus the number of remaining vertices is divided by two each time. Hence this greedy process performs well for a logarithmic number of repetitions, after which the number $|V_i|$ of remaining vertices falls below $\epsilon \log n$ and Chernoff bounds do not provide satisfactory probability anymore. We then finish, by explicitly including the remaining vertices in the dominating set under construction D.

We now provide technical lemmas for the analysis of the algorithm. By using Theorem 15, we get for the $|V_i| - 1$ Bernoulli trials with success probability $\frac{1}{2}$ corresponding to set N_i that:

Lemma 5 *Let $\beta \in [0,1]$ a constant. For any, arbitrarily small, constant $\epsilon > 0$ and given $|V_i| \geq \epsilon \log n$ we have*

$$\Pr\left\{|N_i| \geq (1-\beta)\frac{|V_i|-1}{2}\right\} \geq 1 - \alpha\, n^{-\frac{\beta^2}{4}\epsilon}$$

where $\alpha = 1.28$. □

We now define the event the above lemma is conditioned on.

Definition 2 *Let ε_i = "during the i-th execution of the loop of algorithm GREEDY, $|N_i| \geq (1-\beta)\frac{|V_i|-1}{2}$"*

Definition 3 *Let $\varepsilon = \varepsilon_1 \cap \varepsilon_2 \cap \cdots \cap \varepsilon_t$ until $|V_t| < \epsilon \log n$*

We next provide the following theorem bounding the number of repetitions of the loop of the algorithm, and thus the number of vertices put in the dominating set during phase (2).

Lemma 6 *Conditioning on event* ε*, the number* t *of repetitions of the loop of the algorithm is at most* $(1+\epsilon')\log n$*, where* $\epsilon' > 0$ *a constant.*

Proof: Clearly $|V_{i+1}| \leq (1-\gamma)|V_i|$ after i-th repetition, where $\gamma = \frac{1-\beta}{2}$, and cumulatively $|V_t| \leq (1-\gamma)^t\, n$. Since $(1-\gamma)^t\, n \leq \epsilon \log n$ we get $t \geq \frac{\log n}{\log(\frac{1}{1-\gamma})} - \Theta(\log\log n)$. We note that $\frac{1}{1-\gamma} = \frac{2}{1+\beta}$. By choosing, for any $\epsilon' > 0$,

$$\beta = 2^{\frac{\epsilon'}{1+\epsilon'}} - 1$$

we finally get $t \leq (1+\epsilon')\log n$.

□

We now prove high probability for the event ε.

Lemma 7 *Let any constants* $\beta \in [0,1], \epsilon > 0$ *and* $\alpha = 1.28.$ *We have*

$$\Pr\{\varepsilon\} \geq 1 - \alpha n^{-\frac{\beta^2}{8}\epsilon}$$

Proof: By lemmas 5, 6 we get

$$\Pr\{\overline{\varepsilon}\} \leq \sum_{j=1}^{t} \Pr\{\overline{\varepsilon_j}\} \leq \alpha\, t\, n^{-\frac{\beta^2}{4}\epsilon}$$

hence

$$\Pr\{\overline{\varepsilon}\} \leq \alpha(1+\epsilon')(\log n)\, n^{-\frac{\beta^2}{4}\epsilon} \leq \alpha\, n^{-\frac{\beta^2}{8}\epsilon}$$

for any constants $\beta \in [0,1], \epsilon > 0, \epsilon' > 0.$

□

Combining the above results we get the following final theorem:

Theorem 17 *Algorithm "GREEDY" constructs a dominating set of size* $(1+\epsilon'+\epsilon)\log n$ *in time* $O((1+\epsilon')n\log n)$ *with probability at least* $1 - n^{-\frac{\beta^2}{8}\epsilon}$*.*
□

We remark that algorithm "GREEDY" constructs a near-optimal dominating set, in small polynomial time, both with high probability and on the average.

However, we remark that the probability of success of algorithm "GREEDY", though tending to 1 with n, is not polynomially big, since in $1-n^{-\frac{\beta^2}{8}\epsilon}$ constant $\frac{\beta^2}{8}\epsilon$ is just positive.

In [60], Nikoletseas and Spirakis provide also a better algorithm for the same problem (algorithm "REPEATED TRIALS"), with success probability $1-n^{-\alpha}$, where $\alpha > 1$ a constant. This algorithm is based on the fact that any fixed vertex set of size $(1+\epsilon)\log n$ ($\epsilon > 0$ constant) is a dominating set with high probability.

Although this abundance of near-optimal dominating sets implies that by choosing such a set we may succeed in finding it to be dominating with high probability, the repetition of this experiment provides trials that are not independent since the edges between previously and currently tried sets have been exposed already.

The main idea of this algorithm is to exploit the abundance of dominating sets of size $(1 + \epsilon) \log n$ by repetitively trying new vertex sets in such a way that the independence of the trials is ensured, while "backward" dominance is guaranteed by construction.

1.8 RANDOM REGULAR GRAPHS WITH EDGE FAULTS AND RANDOM CONFIGURATIONS

We note that it is not as easy (from the technical point of view) as in the $G_{n,p}$ case to argue about random regular graphs, because of the stochastic dependencies on the existence of the edges due to regularity. The notion of *configurations*, which we present in the next subsection, was introduced by B. Bollobás ([11, 10]) to translate statements for random regular graphs to statements for the corresponding configurations which avoid the edge dependencies due to regularity and thus are much easier to deal with.

To model edge faults we use the notion of type$-H$ random graphs. Let H be an undirected graph. A *random graph of type*$-H$ is obtained by selecting edges of H independently and with probability p. Let $f = 1 - p$ be called the *link failure probability*.

We study here a simple but fundamental type of H: H being a *random* member of the class of all regular graphs of n nodes and degree r, leading to *random regular graphs with edge faults*, a new class which we call $G^r_{n,p}$. In the model $G^r_{n,p}$, the random selection of a regular graph is capturing the *random allocation of a subnetwork* to a computation (e.g. by the operating system) and the edge failure probability $f = 1 - p$ is capturing the link faults that may happen.

An explicit definition of the new probability space $G^r_{n,p}$ follows.

Definition 4 (the $G^r_{n,p}$ probability space) *Let G^r_n be the probability space of all random regular graphs with n vertices where the degree of each vertex is r and each such graph is selected equiprobably. The probability space $G^r_{n,p}$ of random regular graphs with edge faults is constructed by the following two subsequent random experiments: first, we choose a random regular graph from the space G^r_n and, second, we randomly and independently delete each edge from this graph with probability $f = 1 - p$.*

A preliminary presentation ([62]) of this new model has already inspired relevant research: Nikoletseas et al investigate in [62, 63] important connectivity properties of $G^r_{n,p}$ graphs by estimating the ranges of r, p for which, with high probability, $G^r_{n,p}$ graphs a) are highly connected b) become disconnected and c) admit a giant connected component of small diameter. Also, recently Andreas Goerdt ([35]) continued the work presented in [62] and estimated the threshold probability for the existence of a linear sized component in the faulty version of almost all random regular graphs.

In order to deal with this new model $G^r_{n,p}$, we extend the notion of configurations and the translation lemma between configurations and random regular graphs provided by B. Bollobás ([11, 10]), by introducing here the concept of *random configurations* to account for edge faults, and by also providing an *extended translation lemma* between random configurations and random regular graphs with edge faults.

1.8.1 Bollobás translation lemma. The notion of *configurations* was introduced by B. Bollobás ([11], [10]) to translate statements for random regular graphs to statements for the corresponding configurations which avoid the edge dependencies due to regularity and thus are much easier to deal with:

Definition 5 (configurations, Bollobás, [11]) *Let G be a graph with degrees $d_i, i = 1, \ldots, n$. Let $w = \cup_{j=1}^n w_j$ be a fixed set of $2m = \sum_{j=1}^n d_j$ labeled vertices where $|w_j| = d_j$. A configuration F is a partition of w into m pairs of vertices, called edges of F.*

Given a configuration F, let $\theta(F)$ be the (multi)graph with vertex set V in which (i, j) is an edge if and only if F has a pair (edge) with one element in w_i and the other in w_j.

Bollobás notes that every regular graph $G \in G^r_n$ is of the form $G = \theta(F)$ for exactly $(r!)^n$ configurations. However not every configuration F with $d_j = r$ for all j corresponds to a $G = \theta(F) \in G^r_n$ since F may have an edge entirely in some w_j or parallel edges joining w_i and w_j.

Let ϕ be the set of all configurations F and let G^r_n be the set of all regular graphs. Given a property (set) $Q \subseteq G^r_n$ let $Q^* \subseteq \phi$ such that $Q^* \cap \theta^{-1}(G^r_n) = \theta^{-1}(Q)$.

In order to identify the set $\{\theta^{-1}(G) : G \in G^r_n\}$, Bollobás considers possible cycles of length one (self-loops) and two (loops) among pairs w_i, w_j of a configuration. Let $X_1(F), X_2(F)$ be the numbers of self-loops, loops in a configuration F, respectively.

Clearly, $\theta^{-1}(G^r_n) = \{F : X_1 = X_2 = 0\}$. Thus, if we turn θ into a probability space by giving all configurations the same probability, we can estimate the probability of a property Q, from the corresponding set of configurations Q^* such that $Q^* \cap \theta^{-1}(G^r_n) = \theta^{-1}(Q)$, by using the trivial fact that:

$$\Pr\{Q\} = \Pr\{Q^* | X_1 = X_2 = 0\} \geq 1 - \frac{1 - \Pr\{Q^*\}}{\Pr\{X_1 = X_2 = 0\}} \tag{1.4}$$

Bollobás proves, in an ingenious way, by examining joint factorial moments, that the distributions of X_1, X_2 are asymptotically independent Poisson distributions, with mean $E(X_i) = \frac{\lambda^i}{2i}$, $i = 1, 2$ where

$$\lambda = \frac{1}{m} \sum_{i=1}^{n} \binom{d_i}{2} = r - 1$$

thus

$$\Pr\{X_1 = X_2 = 0\} \sim e^{-\frac{\lambda}{2} - \frac{\lambda^2}{4}}$$

So, Bollobás shows the following important lemma:

Lemma 8 (Bollobás, [10]) *If $r \geq 2$ is fixed and Q^* holds for a.e. configuration, then Q holds for a.e. $r-$regular graph.* □

The main importance of the above lemma is that when we study random regular graphs whose degree does not grow too fast with n, instead of considering the set of all random regular graphs, we can study the (much more easier to deal with) set of configurations.

1.8.2 An extended translation lemma. In order to deal with edge failures, we present here the following extension of the notion of configurations:

Definition 6 (random configurations, Nikoletseas et al, [62]) *Let G a graph with degrees $d_i, i = 1, \ldots, n$. Let $w = \cup_{j=1}^{n} w_j$ be a fixed set of $2m = \sum_{j=1}^{n} d_j$ labeled "vertices" where $|w_j| = d_j$. Let F be any configuration of the set ϕ. For each edge of F, remove it with probability $1 - p$, independently. Let $\hat{\phi}$ be the new set of objects and $\hat{F}$ the outcome of the experiment. $\hat{F}$ is called a* random configuration.

Now let $\bar{Q}$ be a property for $G^r_{n,p}$ graphs and $\hat{Q}$ the corresponding property for random configurations.

By introducing probability p in every edge, direct translation of the proof of Lemma 8 leads (since in both $\bar{Q}$ and $\hat{Q}$ each edge has the same probability and independence to be deleted, thus the modified spaces follow the properties of Q and Q^*) to the following extension to random configurations.

Lemma 9 (extended translation lemma) *Let $r \geq 2$ fixed and $\bar{Q}$ be a property for $G^r_{n,p}$ graphs. If $\hat{Q}$ holds for a.e. random configuration, then the corresponding property $\bar{Q}$ holds for a.e. graph in $G^r_{n,p}$.* □

Important Remark: We note that, *for monotone increasing properties*, results proved using the above extended translation lemma for $r \geq 2$ fixed, *obviously extend to greater (unbounded) r*, since (because of the monotonicity) increasing r can only strengthen the probability of such properties. In this work, we use this remark for the monotone properties a) of graph connectivity and b) of the existence of a giant connected component in a graph.

In the next two sections, we present the work of Nikoletseas et al ([62, 63]) concentrating on the investigation of connectivity properties of $G^r_{n,p}$. We show that:

1. $G^r_{n,p}$ is $r-$connected, except for $O(1)$ vertices, with high probability, for all failure probabilities $f \leq n^{-\epsilon}$ ($\epsilon > 0$ fixed).

2. $G^r_{n,p}$ is disconnected, almost certainly, for constant f and any $r \leq \frac{1}{2}\sqrt{\log n}$, but is highly connected, with high probability, when $r \geq \alpha \log n$, where $\alpha > 0$ a constant.

3. Even when $G^r_{n,p}$ becomes disconnected, it still has a giant connected component of small diameter, even when $r = O(1)$.

1.9 CONNECTIVITY PROPERTIES IN RANDOM REGULAR GRAPHS WITH EDGE FAULTS

Let G^r_n be the class of all regular graphs on n vertices of degree r. It is known (see e.g. [10]) that G^r_n is almost certainly $r-$connected for $r \geq 3$. In the next section we show the impressive effect of constant faults in connectedness, since we prove that the corresponding graphs with faults become, in the case of small degrees, disconnected, with high probability.

1.9.1 $G^r_{n,p}$ becomes disconnected in the case of constant edge failure probability and small degrees . Using the notion of random configurations presented in the previous section, we can now prove the following theorem:

Theorem 18 *When $2 \leq r \leq \frac{\sqrt{\log n}}{2}$ and $p = \Theta(1)$ then $G^r_{n,p}$ has at least one isolated node with probability at least $1 - n^{-k}, k \geq 2$.*

Proof: Consider the random configuration $\hat{F}$ corresponding to $G^r_{n,p}$. In the configuration F (out of which $\hat{F}$ was produced) there exist $n/(r+1)$ "vertices" that do not share an "edge". The probability that each particular one of them (say v) is isolated in $\hat{F}$ is:

$$\Pr\left\{v \text{ isolated in } \hat{F}\right\} = (1-p)^r$$

With respect to the above event, the $n/(r+1)$ vertices define $n/(r+1)$ *independent* experiments. Let $\hat{E}$ the event "there exists at least one isolated vertex in $\hat{F}$ out of the $\frac{n}{r+1}$ vertices". Thus we have:

$$\Pr\left\{\hat{E}\right\} = 1 - (1-(1-p)^r)^{\frac{n}{r+1}} \geq 1 - \exp\left(-\frac{n}{r+1}(1-p)^r\right)$$

But for $r \leq \frac{\sqrt{\log n}}{2}$ and by setting $f = 1 - p$, we get:

$$\Pr\left\{\hat{E}\right\} \geq 1 - \exp\left(-\frac{n^{1-\frac{\log(1/f)}{2\sqrt{\log n}}}}{0.5\sqrt{\log n}+1}\right) \geq 1 - n^{-c}$$

for any constant $c > 0$. Thus, by the remark above and lemma 6, if $\bar{E}$ is the event that $G^r_{n,p}$ has an isolated point, then:

$$\Pr\{\bar{E}\} \geq 1 - \frac{1 - \Pr\{\hat{E}\}}{\Pr\{\hat{X}_1 = \hat{X}_2 = 0\}} \geq 1 - \frac{n^{-c}}{\Pr\{X_1 = X_2 = 0\}}$$

But $\Pr\{X_1 = X_2 = 0\} \sim e^{-\frac{\lambda}{2}-\frac{\lambda^2}{4}}$. Since $\lambda = r - 1$, we get $e^{-\frac{\lambda}{2}-\frac{\lambda^2}{4}} \sim n^{-\frac{1}{16}}$. Thus, $\Pr\{\bar{E}\} \geq 1 - n^{-k}$, for $k \geq c - \frac{1}{16}$. The proof is completed by choosing $c \geq 2 + \frac{1}{16}$. □

In the next section we study constant link failure probability f, which represents a worst case for the connectivity preservation. Still, we are able to show that logarithmic degrees suffice to guarantee that $G^r_{n,p}$ remains almost certainly highly connected, despite these constant edge failures.

1.9.2 Large degree random regular graphs are robust even for constant failure probability. Let $r \geq \alpha \log n$, for α being a suitably chosen constant, and let the probability of edge failures f be constant (w.l.o.g. let $f = 1/2$).

Then, for any particular vertex v, whose degree after the failures is denoted by $deg(v)$, we have:

$$\Pr\left\{(1-\beta)rp \leq deg(v) \leq (1+\beta)rp\right\} \geq 1 - 2e^{-\frac{\beta^2}{2}rp} \geq 1 - 2n^{-\gamma}$$

where $\gamma \geq 2$ a constant, by Chernoff bounds, and by choosing $\alpha \geq 2\gamma/(\beta^2 p)$. Clearly then:

$$\Pr\left\{\text{for all } v, (1-\beta)rp \leq deg(v) \leq (1+\beta)rp\right\} \geq 1 - 2n^{-(\gamma-1)}$$

We note that, interestingly, this property is shared by the graphs $G_{n,p'}$ for $p' \geq (\alpha p \log n)/n$, by the same technical argument. Indeed, remark that in $G^r_{n,p}$ graphs, the neighbors of any vertex are determined by first selecting r edges at random among all $n-1$ possible, and by then sampling them randomly and independently with the same probability p. In $G_{n,p'}$ graphs, all $n-1$ possible adjacent edges of a vertex are sampled independently and with the same probability p'. In either case, we end up, with high probability, with almost the *same number* of edges adjacent to any vertex, which are *randomly selected*. We note that it is true that the spaces $G^r_{n,p}$ and $G_{n,p'}$ are different, but the difference includes only the graphs where although p' is small, the degree of a vertex may exceed r, and this event is of negligible probability compared to the rest, from Chernoff bounds. In fact, by choosing (for an $\beta \in (0,1)$) $np' = \frac{rp}{1+\beta}$, the model $G_{n,p'}$ is weaker that the $G^r_{n,p}$. That is, in this case the "randomly truncated" random regular graph $G^r_{n,p}$ and $G_{n,p'}$ *produce almost the same graph outcomes and have the same properties*, with extremely high probability. The minimum necessary probability p_c so that $G_{n,p}$ graphs are a.c. k-connected when $p \geq p_c$, is given in [10] (p. 152, lemma 5). By calculating the maximum value of k such that

$$p' = \frac{\alpha \log n}{2(1+\beta)n} \geq p_c = \frac{\log n + (k-1)\log\log n - \omega(n)}{n}$$

where $\omega(n) \to \infty$, $\omega(n) \leq \log\log\log n$, we get the following theorem.

Theorem 19 *Let G be an instance of $G^r_{n,p}$ where $p = \Theta(1)$ and $r \geq \alpha \log n$, where $\alpha > 0$ an appropriate constant. Then G is almost certainly k-connected, where*

$$k = O\left(\frac{\log n}{\log\log n}\right)$$

□

We remark here that the constant link failure probability dramatically alters the connectivity structure of the regular graph. While G^r_n is a.c. $r-$connected for any $r \geq 3$, we get that $G^r_{n,p}$ is *disconnected* w.h.p. for all $r \leq \frac{\sqrt{\log n}}{2}$ and that large degrees $r \geq \alpha \log n$ are needed for $G^r_{n,p}$ to be a.c. connected. The gap $\left(\frac{1}{2}\sqrt{\log n}, \alpha \log n\right)$ is due to the translation theorem from random configurations to $G^r_{n,p}$ (we recall that the extended translation lemma is valid for $r \geq 2$ fixed). In order to handle this gap, we must avoid the translation lemma. In fact, first consider a graph $G \in G^r_n$. By Turán's celebrated theorem ([72]), it contains an independent set S with cardinality $|S| \geq \frac{n}{r+1}$. Now delete edges from the graph with probability p, and let X_v denote the event that in the new graph G_p obtained in this way a vertex v is isolated. Then, for each pair v, w of vertices of S the events X_v and X_w are independent. Now simple calculations and Chebyshev's inequality reveal that w.h.p. the graph G_p contains an isolated vertex. Then we can repeat the calculations that we did in the proof of Theorem 1 (exactly the same argument but not in the configuration graph, so we avoid the translation lemma in this case) and get that $G^r_{n,p}$ is disconnected a.c. for any $r = o(\log n)$ and constant f.

1.9.3 $G^r_{n,p}$ is $r-$connected for $f = O(n^{-\epsilon})$ except for $O(1)$ vertices.

We now consider the (more practical) case in which $f = 1 - p = o(1)$ and we prove that the nice properties of random regular graphs are preserved despite link failures.

The technical lemmas related to random configurations are still valid. By a similar result in [10] (p. 174, Theorem 32) on G_n^r and by the fact that a $G_{n,p}^r$ graph can not have more edges than the corresponding G_n^r graph, the following also holds:

Lemma 10 *There is a constant $c = c(r)$ such that $G_{n,p}^r$ graphs do not have a subgraph with $k < c \log n$ vertices and at least $k+1$ edges; this holds with probability at least $1 - e^{-n}$.* $\square$

Consider now in $G_{n,p}^r$ a partition of $V : V = A \cup S \cup B$, where $s_1 = |S| \geq (r-2)\alpha_0, \alpha_0 \geq 3$. Let $\alpha_1 = 4(r-2)\alpha_0$ and consider all partitions in which $|A| = \alpha \geq \alpha_1$. Consider the event: $\varepsilon_1(\alpha) =$ "There is a partition $V = A \cup S \cup B$ as above with no $A-B$ edges, i.e. with S a separating set and $|A| = \alpha$ ".

Theorem 20 *Almost every $G_{n,p}^r$ graph is such that if a smaller than cardinality r set separates it, then only $O(1)$ vertices break away and the rest of the graph (of $O(n)$ size) is $r-$connected.*

Proof: It suffices to show that $\sum_\alpha Pr\{\varepsilon_1(\alpha)\} \to 0$. By Lemma 10, $G_{n,p}^r$ is such that w.h.p. every set of $k < c(r)\log n$ vertices spans at most k edges.

We have $\binom{n}{s_1}$ choices for S and $\binom{n-s_1}{\alpha}$ choices for A. By using random configurations we see that a configuration edge incident with an element of $\hat{A}$ joins this element to $\hat{B}$ with probability at least $(\frac{n-s_1-\alpha}{n})p$.

Let $S(\alpha) = Pr\{\varepsilon_1(\alpha)\}$. By pairing off elements of $\hat{A}$ one by one and by conditioning on the events of Lemma 10 we get:

$$S(\alpha) \leq \binom{n}{s_1}\binom{n-s_1}{\alpha}\left(1 - \frac{n-s_1-\alpha}{n}\ p\right)^{\frac{\alpha r}{2}}$$

Now, we use $f = 1-p$ to get :

$$S(\alpha) \leq S'(\alpha) = n^{s_1}\binom{n-s_1}{\alpha}\left(\frac{s_1+\alpha}{n}(1-f)+f\right)^{\frac{\alpha r}{2}}$$

We will study only the worst, with respect to connectivity issues, case where $r = O\left(\sqrt{\log n}\right)$. For bigger r, things are even better.

Clearly, the hardest case for connectivity is when f is large. So, let us assume that $f = \Omega\left(\frac{\sqrt{\log n}}{n}\right)$. Then $S'(\alpha) \leq n^{s_1+\alpha}((1+\gamma)f)^{\alpha r}$ for some $\gamma > 0$. We remark that if $f \leq \frac{1}{1+\gamma}\ n^{-\frac{2s_1+\alpha}{\alpha r}}$ then $S'(\alpha) \leq n^{-s_1}$.

Let $\alpha_2 = \lfloor\frac{n-\alpha_1}{2}\rfloor$. Clearly

$$\sum_\alpha \Pr\{\varepsilon_1(\alpha)\} \leq \sum_{\alpha=\alpha_1}^{\alpha_2} S'(\alpha) \tag{1.5}$$

Since we assumed $f = \Omega\left(\frac{\sqrt{\log n}}{n}\right)$, we have that

$$\frac{\sqrt{\log n}}{n} < f \leq \frac{1}{1+\gamma}\ n^{-\frac{2s_1+\alpha}{\alpha r}}$$

Note that: $\frac{S'(\alpha+1)}{S'(\alpha)} \sim nf^r$ i.e. $\frac{S'(\alpha+1)}{S'(\alpha)} \leq n(\frac{1}{1+\gamma})^r\, n^{-\frac{2s_1+\alpha}{\alpha}} < \frac{1}{\sqrt{n}}$, since $4s_1 \geq \alpha$. Thus, from inequality (1.5) above, we easily get

$$\sum_{\alpha=\alpha_1}^{\alpha_2} S'(\alpha) \to 0$$

as $n \to +\infty$. □

From the above theorem we get:

Theorem 21 *Let $r \geq 3$ and $f = 1 - p = O(n^{-\epsilon})$ for $\epsilon \geq \frac{3}{2r}$. Then $G^r_{n,p}$ is $r-$connected, except for $O(1)$ vertices, with probability tending to 1 as n tends to $+\infty$.* □

1.10 GIANT COMPONENTS IN RANDOM REGULAR GRAPHS WITH EDGE FAULTS

Since $G^r_{n,p}$ is a.c. disconnected for $r = o(\log n)$ and $1 - p = f = \Theta(1)$, we would like to know whether at least a large part of the network represented by $G^r_{n,p}$ is still connected, i.e. whether the biggest connected component of $G^r_{n,p}$ is large. Perhaps the most celebrated theorem in Random Graph Theory is that of the Giant Component: $G_{n,p}$ has a giant (i.e. $\Theta(n)$-sized) connected component when, roughly, $p > \frac{1}{n}$. This was first remarked by Erdös and Renyi ([26]). We will here show that:

Theorem 22 *When $f < 1 - \frac{32}{r}$ then $G^r_{n,p}$ admits a giant (i.e. $\Theta(n)$-sized) connected component with probability at least $1 - O\left(\frac{\log^2 n}{n^{\alpha/3}}\right)$, where $\alpha > 0$ a constant that can be selected.*

Proof: We will prove the Theorem for r fixed by using the extended translation lemma. The result then obviously extends to greater (unbounded) r by the monotonicity of the giant connected component property.

We will consider the random configuration $\hat{F}$ corresponding to $G^r_{n,p}$. We first select each existing edge of $\hat{F}$ with probability 1/2, independently. This partitions the edges of $\hat{F}$ into two sets of edges (E_1 and E_2).

Since $f < 1 - \frac{32}{r}$ we have $p > \frac{32}{r}$ i.e. $pr > 32$. In the graph G^r_n we have a total number of edges $\frac{rn}{2}$ and after the experiment of *independently* deleting each edge with probability f and the experiment of the partition, the remaining number of edges in E_1, E_2 satisfies the Bernoulli $B\left(\frac{nr}{2}, \frac{p}{2}\right)$ i.e. by Chernoff bounds we get that $|E_1|, |E_2| \geq (1-\beta)\frac{nrp}{4}$, for any $\beta \in (0,1)$ w.h.p. and since $pr > 32$ we get $|E_1|, |E_2| \geq (1-\beta)8n$ with probability $\geq 1 - \exp\left(-\frac{\beta^2}{2}8n\right)$. We now condition, for E_1, on the event $A_1 =$ "$|E_1| \geq (1-\beta)8n''$.

Let $\hat{F}(E_i)$ the random configuration corresponding to the edge set E_i, $i = 1, 2$. The following technical claim will be needed here:

Claim 1 *$\hat{F}(E_1)$ contains a "path" (i.e. a sequence of sets w_i, of r vertices, each connected to the next by an edge) of length at least $\alpha \log n$ ($\alpha > 0$ an appropriate constant that can be selected) with probability at least $1 - n^{-\lambda}, \lambda \geq 2$.*

Proof: Consider now the set E_1 partitioned into g groups $g_1, g_2, \ldots$ of $\alpha \log n$ ($\alpha > 0$ an appropriate constant) edges in each group. For each group g_i let q_i be the probability that it will form a path in $\hat{F}(E_1)$ if thrown alone in the $\binom{rn}{2}$ possible "places". Let a random indicator variable $X_i = 1$ with probability q_i and $X_i = 0$ else. Clearly the random variables X_i are *not independent*. The number of ways for g_i to form a path is calculated as follows:

(a1) Choose the supervertices of the path in $\binom{n}{\alpha \log n}$ ways

(a2) Order them in the path in $(\alpha \log n)!$ ways

(a3) In this order $(r^2)^{\alpha \log n - 1}$ paths are defined, because any two specific supervertices can be joined by an edge in r^2 ways.

But there are

$$\binom{\binom{rn}{2}}{\alpha \log n}$$

places for the edges of g_i. Thus

$$q_i = \frac{\binom{n}{\alpha \log n}(\alpha \log n)!\ r^{2\alpha \log n - 2}}{\binom{\binom{rn}{2}}{\alpha \log n}}$$

The denominator is at most $(rn)^{2\alpha \log n} = r^{2\alpha \log n} n^{2\alpha \log n}$, thus

$$q_i \geq \frac{\frac{n!}{(\alpha \log n)!(n - \alpha \log n)!}(\alpha \log n)!}{r^2\ n^{2\alpha \log n}}$$

i.e.

$$q_i \geq \frac{n(n-1)\cdots(n - \alpha \log n + 1)}{r^2 n^{2\alpha \log n}}$$

and since $r \leq n - 1$ we get

$$q_i \geq \frac{(n-2)\cdots(n - \alpha \log n + 1)}{n^{2\alpha \log n}} \geq \frac{(\frac{n}{2})^{\alpha \log n - 2}}{n^{2\alpha \log n}} \geq \frac{4}{n^{\alpha \log n + 2 + \alpha}}$$

But there are $N \geq \binom{4n}{\alpha \log n}$ groups g_i i.e.

$$N \geq \frac{(4n)!}{(\alpha \log n)!(4n - \alpha \log n)!}$$

i.e. (crudely, and for sufficiently large n)

$$N \geq \frac{(3.9n)^{\alpha \log n}}{(\alpha \log n)!} \geq (3n)^{\alpha \log n}$$

Notice that forall g_i

$$N \cdot q_i \geq 4(1.5)^{\alpha \log n}$$

i.e.

$$N \cdot q_i \geq 4 \left(\frac{3}{2}\right)^{\alpha \log n} = 4n^{\alpha \log\left(\frac{3}{2}\right)}$$

As remarked in [24], "a frequently occurring scenario underlying the analysis of many randomized algorithms and processes involves random variables that are, intuitively, dependent in the following *negative* way: if one subset of the variables is "high", then a disjoint subset of the variables is "low". One specific paradigm involving negative dependency is the classical ball and bins experiment".

Definition 7 (Negative Association, [24]) *Let $X := (X_1, \ldots, X_n)$ be a vector of random variables. The random variables X_i are negatively associated if for every two disjoint index sets $I, J \subseteq [n]$,*

$$E[f(X_i, i \in I)g(X_j, j \in J)] \leq E[f(X_i, i \in I)]E[g(X_j, j \in J)]$$

for all functions $f : R^{|I|} \to R$ and $g : R^{|J|} \to R$ that are both non-decreasing or both non-increasing.

Now we notice that in our case the random indicator variables X_i and X_j (showing whether a path is formed or not in group g_i, g_j) are negatively associated for all i, j, because if $X_j = 1$ (high) then X_i is lower since some places are already occupied. That is

$$E(X_i | X_j) \leq E(X_i)$$

i.e.

$$E(X_i X_j) \leq E(X_i)E(X_j)$$

(negative covariance).

We are interested in the sum $X = \sum_i X_i$ i.e. in the number of paths of length $\alpha \log n$ in $\hat{F}(E_1)$. From [24] we have Theorem 23. For its proof we will use the following Lemma.

Lemma 11 (Dubhashi and Ranjan, [24]) *Let $X_1, \ldots, X_n$ random variables satisfying the negative association condition. Then for any non-decreasing functions $f_i, i \in [n]$,*

$$E\left(\prod_{i\in[n]} f_i(X_i)\right) \leq \prod_{i\in[n]} E(f_i(X_i))$$

Proof: Take the non-decreasing functions $f(X_i, i < n) := \prod_{i<n} f_i(x_i)$ and $g(x_n) := f_n(x_n)$ to deduce that

$$E\left(\prod_{i\in[n]} f_i(X_i)\right) \leq E\left(\prod_{i<n} f_i(X_i)E(f_n(X_n))\right)$$

and now use induction.
(end of proof of Lemma 11).

Theorem 23 (Dubhashi and Ranjan, [24]) *The Chernoff-Hoeffding bounds are applicable to sums of random variables that satisfy the negative association condition, as in definition 7.*

Proof: Let $X_1, \ldots, X_n$ be negatively associated (and bounded) variables. To show that the Chernoff-Hoeffding bounds apply to the sum $X = X_1 + \cdots + X_n$, we use the standard proof of the Chernoff-Hoeffding bound, see for example [57]. The only change needed is in a crucial step, where one uses the fact for *independent* variables, $E(e^{tX}) = E(\prod_i e^{tX_i}) = \prod_i E(e^{tX_i})$. For negatively associated variables, we have, for $t > 0$, $E(e^{tX}) = E(\prod_i e^{tX_i}) \leq \prod_i E(e^{tX_i})$, by Lemma 11 applied with each $f_i(x) := e^{tx}$. The rest of the proof is unchanged, and gives the upper tail bound. For the lower tail, we apply the same argument to the variables $b_i - X_i$, where b_i is an upper bound on the variable X_i. Note that if the X_i variables are negatively associated, then so are the variables $b_i - X_i$. (In fact, because of the monotonicity of the exponential function, the negative covariance relation among the variables suffices to prove the theorem).
(end of proof of Theorem 23)

Thus, the Chernoff-Hoeffding bounds are applicable in this case and we have that, for any $\gamma \in (0, 1)$,

$$\Pr\{X \geq (1-\gamma)Nq_i\} \geq 1 - e^{-\frac{\gamma^2}{2}Nq_i}$$

i.e. the number of paths of length $\alpha \log n$ in $\hat{F}(E_1)$ is at least $4n^{\alpha\log\left(\frac{3}{2}\right)}$ with probability at least

$$1 - e^{-\frac{\gamma^2}{2}4n^{\alpha\log\left(\frac{3}{2}\right)}}$$

As an easy conclusion, $\hat{F}(E_1)$ contains at least one such path of length $\alpha \log n$ with probability at least

$$\left(1 - e^{-\frac{\gamma^2}{2}4n^{\alpha \log\left(\frac{3}{2}\right)}}\right)\left(1 - e^{-\frac{\beta^2}{2}8n}\right) \geq 1 - n^{-\lambda}$$

for any $\lambda \geq 1$.

(end of proof of Claim 1)

Let us now condition the rest of the proof of the theorem on the existence of such a path $\Pi = (P_1, P_2, \ldots, P_{\alpha \log n})$, where each "supervertex" P_i (of r vertices) is connected to P_{i+1} by an edge of E_1. We now define a Breadth-First-Search Expansion Process in $\hat{F}(E_2)$ as follows:

BFS Expansion Process

1. Let $B_0 = \{P_1, P_2, \ldots, P_{\alpha \log n}\}; i = 0$.
2. Let $i \leftarrow i + 1$.
3. Let B_i be the set of supervertices not in S_{i-1} that are connected to at least one supervertex in S_{i-1} by an edge.
4. If $B_i \neq \emptyset$ then go to 2.

For each set of supervertices B_i, define:

Definition 8 *Let $S_i = \cup_{j \leq i} B_j$.*

Definition 9 *Let Active(B_i) be the set of those supervertices in B_i each of which has only at most $r/2$ edges connected to S_i.*

Definition 10 *Let Out(B_i) be the set of edges of B_i connecting to vertices which do not belong to S_i.*

Definition 11 *Let R_i be the event $|Active(B_i)| \geq \frac{|B_i|}{2}$.*

Note that the edges we use in the expansion process are independent of the edges we used to get the existence of the path Π. Let $i \leq \log n - o(\log n)$. Choose $\alpha \geq \frac{6 \cdot 32}{rp}$. Assume inductively that

$$\Pr\{R_i\} \geq \left(1 - \frac{1}{n^{\alpha/3}}\right)^i$$

For $i = 0$, $\Pr\{R_i\} = 1$ (basis). By the induction hypothesis and conditioning on R_i, we get that at least $\frac{r}{2}\frac{|Bi|}{2}$ edges of S_i go out of S_i. Each of them will exist with probability p. Thus, by the Bernoulli $B\left(\frac{r}{2}\frac{|Bi|}{2}, p\right)$, and because $rp > 32$, the event $\epsilon_1(i) =$ "at least $(1-\beta_1)8|B_i|$ edges exist and go out of S_i departing from S_i" has (by Chernoff bounds) probability at least $1 - \exp\left(-\frac{\beta_1^2}{2}8|B_i|\right)$ and since $|B_i| \geq \alpha \log n$ we get that

$$\Pr\{\epsilon_1(i)\} \geq 1 - e^{-\frac{\beta_1^2}{2}8\alpha\log n} \geq 1 - n^{-\beta_1^2 4\alpha}$$

for any $\beta_1 \in (0,1)$.

Now, condition also on $\epsilon_1(i)$. Then let us see how many new supervertices these edges hit. We can safely assume that during the process, $|S_i|$ (i.e. the number of supervertices in the component in the i-th stage of the BFS) is less than $\frac{n}{8}$ (else we have a giant component already). But then consider the edges out of S_i in the order $e_1, e_2, \ldots, e_x$ ($x \geq (1-\beta_1)8|B_i|$). Let the set $\{e_1, e_2, \ldots, e_\kappa\}$ have hit m_1 new supervertices. We may again assume $m_1 < \frac{n}{8}$ (since else we are done). Consider $e_{\kappa+1}$. The probability that it hits another *new* supervertex is at least $\frac{n-\frac{n}{8}-\frac{n}{8}}{\frac{7n}{8}} \geq \frac{6}{7}$ and this holds for all $\kappa \leq x$. Consider the Bernoulli $B\left(x, \frac{6}{7}\right)$. By Chernoff bounds, the number of edges that each hits a supervertex separate from those hit before is $(1-\beta_2)x\,\frac{6}{7}$ thus at least $(1-\beta_2)(1-\beta_1)8|B_i|\frac{6}{7}$ i.e. at least $(1-\beta_3)6|B_i|$ with probability at least

$$1 - e^{-\frac{\beta_2^2}{2}6|B_i|} \geq 1 - n^{-3\alpha\beta_2^2}$$

since $|B_i| \geq \alpha \log n$.

Note that by appropriately choosing β_1, β_2 we get that the number of edges out of S_i that each hits a new supervertex separate from before, is at least $6|B_i|$ with probability at least $1 - n^{-3\alpha\frac{\beta_2^2}{2}}$. But each of these new supervertices (they belong to B_{i+1}) is *active by construction*, because each has only one edge into S_i, thus $r-1$ edges free to expand. So, given that R_i holds, the conditional probability $\Pr\{R_{i+1}|R_i\}$ is at least

$$\Pr\{R_{i+1}|R_i\} \geq \Pr\{\epsilon_1(i)\}\left(1 - n^{-3\alpha\beta_2^2}\right) \geq 1 - n^{-\frac{\alpha}{3}}$$

Thus

$$\Pr\{R_{i+1}\} \geq \Pr\{R_i\}\Pr\{R_{i+1}|R_i\} \geq \left(1 - n^{-\frac{\alpha}{3}}\right)^{i+1}$$

(end of proof of induction)

So, in each stage we get an expansion of the number of connected new supervertices by 6, provided $i \leq \gamma \log n - o(\log n)$ ($\gamma \geq 1$). Thus, at the end we get a giant component of size $\Theta(n)$, *provided all events* R_i *hold.*

The probability of the event "$\exists i : \bar{R}_i$" is the probability of failure of the process. This is at most

$$\Pr\left\{\exists i : \bar{R}_i\right\} \leq \sum_{i=0}^{\log n} \Pr\{\bar{R}_i\}$$

But

$$\Pr\{R_i\} \geq \left(1 - \frac{1}{n^{\frac{\alpha}{3}}}\right)^i \geq 1 - \frac{i}{n^{\frac{\alpha}{3}}}$$

thus

$$\Pr\{\bar{R}_i\} \leq \frac{i}{n^{\frac{\alpha}{3}}} \leq \frac{\log n}{n^{\frac{\alpha}{3}}}$$

and finally

$$\sum_{i=0}^{\log n} \Pr\{\bar{R}_i\} \leq \frac{\log^2 n}{n^{\frac{\alpha}{3}}}$$

This ends the proof of Theorem 22.

Recently, Andreas Goerdt ([35]) continued the work presented in a preliminary version of this paper ([62]) and showed the following results: if the degree r is fixed then $p = \frac{1}{r-1}$ is a threshold probability for the existence of a linear sized component in the faulty version of almost all random regular graphs. In fact, he further shows that if each edge of an *arbitrary* graph G with maximum degree bounded above by r is present with probability $p = \frac{\lambda}{r-1}$, when $\lambda < 1$, then the faulty version of G has only components whose size is at most logarithmic in the number of nodes, with high probability. His result implies some kind of optimality of random regular graphs with edge faults.

2. TRADE-OFFS BETWEEN DENSITY AND ROBUSTNESS IN RANDOM INTERCONNECTION GRAPHS

In this section, we discuss the work of Ph. Flajolet, K. Hatzis, S. Nikoletseas and P. Spirakis ([27]) applying combinatorial and symbolic-analytic techniques in order to characterize the interplay between two parameters of a random graph: its density (number of edges) and its robustness to link failures, where robustness means multiple connectivity by short paths. A triple (G, s, t), where G is a graph and s, t are designated vertices, is called $\ell - robust$ if s and t are connected via at least two edge-disjoint paths of length at most ℓ. Flajolet et al determine in [27] the expected number of ways to get from s to t via two edge-disjoint paths in the random graph model $G_{n,p}$ and also derive bounds on related threshold functions.

2.1 INTRODUCTION

Given a triple (G, s, t), where G is a $G_{n,p}$ random graph and s, t are two of its nodes, a natural notion of robustness is to require at least two edge disjoint paths of short length (say, exactly ℓ or at most ℓ) between s and t, so that connectivity by short paths survives, even in the event of a link failure. We next give the following formal definition of $\ell - robustness$.

Definition 12 (ℓ-robustness) *A random graph G of the model $G_{n,p}$ is $\ell - robust$ for two vertices s, t when there are, with high probability, two edge-disjoint paths of length at most ℓ between s, t in G.*

In this section, we investigate the expected number of such paths between two vertices of the random graph, as well as bounds for the threshold probability $p_{n,\ell}$ (as functions of ℓ and n) for the existence of such paths in the $G_{n,p}$ random graph G.

It turns out that even the enumeration of paths among the vertices 1 and n that avoid all edges of the graph $(1, 2 \ldots, n)$ but pass through its vertices, is a non-trivial task. In fact, such an enumeration is a special case of enumerating permutations $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of $(1, 2, \ldots, n)$ where certain gaps $\sigma_{i+1} - \sigma_i$ are forbidden. In our case, $\sigma_{i+1} - \sigma_i$ must not be in the set $\{-1, 1\}$.

In this section, we provide a precise estimate of the expected number of unordered pairs of paths in a random graph that connect a common source to a common destination, and have no edge in common, though they may share some nodes. Thus, for any given set of values of n, p, ℓ (where ℓ represents the length) we estimate precisely the mean number of avoiding pairs in graphs of a given size. This leads to a tight bound for the probability of non-existence of such paths between any fixed pair of nodes and also to a bound for the probability of the existence of many pairs of edge disjoint paths of length ℓ.

In order to achieve this, Flajolet et al devise a finite state mechanism that describes classes of permutations with free places and exceptions. The finite-state description allows for a direct construction of a multivariate generating function. The generating function is then subjected to an integral transform that implements an inclusion-exclusion argument and an explicit enumeration result; see Theorems 24 and 25. This enables them to quantify the trade-off between ℓ-robustness (as defined above) and the density of the graph (i.e. the number of its edges). The originality of our approach consists in introducing in this range of problems methods of analytic combinatorics and recent research in automatic analysis (based on symbolic computation). For context, see [28], [29], [19]. Additional threshold estimates regarding properties of multiple source-destination pairs are discussed in the last two sections of the paper.

From earlier known results ([10], [62]) and the results in this section, a picture of robustness under the $G_{n,p}$ model emerges. (As usually in random graph theory, various regimes for $p = p(n)$ are considered). Start with an initially totally disconnected graph, corresponding to $p = 0$. As p increases, the graph becomes connected near the connectivity threshold $p_c(n) \simeq (\log n)/n$. All the following results hold for any integer ℓ: $6 \leq \ell < n$. Any fixed s, t pair (or equivalently a random s, t pair, given the invariance properties of $G_{n,p}$) *is "likely" to be ℓ-robust* when p crosses the value

$$p_{n,\ell}(n) = 2^{\frac{1}{2\ell}}\ n^{-1+\frac{1}{\ell}}$$

(see Theorem 26 and Equation 1.11). (Here "likely" means that the expected number of edge-disjoint pairs is at least 1 when $n \to \infty$). As long as $p \leq p_L(n, \ell)$, where

$$p_L(n,\ell) = n^{-1+\frac{1}{\ell}} \left(\log \frac{n^2}{\log n}\right)^{\frac{1}{\ell}}$$

we know, w.h.p., the *existence* of s, t pairs that are *not* connected by short (of length at most ℓ) paths; see Theorem 27. (The function $p_L(n)$ is in fact a threshold for diameter). However, one only needs a tiny bit more edges, namely $p \geq p_U$:

$$p_U(n,\ell) = n^{-1+\frac{1}{\ell}}\ 2\left(\log\left(n^2 \log n\right)\right)^{\frac{1}{\ell}}$$

to ensure that *almost all* s, t-pairs are ℓ-robust; see Theorem 28.

2.2 THE ENUMERATION OF "AVOIDING" CONFIGURATIONS

The problem at hand is that of estimating the expected number of "avoiding pairs" of length ℓ between a random source and a random destination in a $G_{n,p}$ random graph G. (An avoiding pair of length ℓ means an unordered pair of paths, each of length ℓ, that connect a common source to a common destination, and have no edge in common though they may share some nodes). This problem first necessitates the solution of enumeration problems that involve two major steps:

— Enumerate simple paths called "avoiding permutations" of length ℓ that can be viewed as hamiltonian paths on the set of nodes $[1, \ldots, \ell + 1]$, connecting the source 1 and the destination $\ell + 1$, and having no edge of type $[i, i + 1]$ or $[i, i - 1]$.

— Enumerate so-called "avoiding paths", that are simple paths allowed to contain outer nodes taken from outside the segment $[1, \ldots, \ell + 1]$. This situation is closer to the random graph problem since it allows nodes taken from the pool of vertices available in the graph $G \in G_{n,p}$.

The first problem is of independent combinatorial interest as it is equivalent to counting special permutations with restrictions on adjacent values. It also serves as a way to introduce the methods needed for the complete random graph problem. Both problems rely on the inclusion-exclusion principle that is familiar from combinatorial analysis and counting by generating functions (GF's). Applications of these results to the $G_{n,p}$ model are treated in the next section.

2.2.1 Symbolic Enumeration Methods. Flajolet et al ([27]) use a symbolic approach to combinatorial enumeration, according to which many general set-theoretic constructions have direct translations over generating functions. A specification language for elementary combinatorial objects is defined for this purpose. The problem of enumerating a class of combinatorial structures then simply reduces to finding a proper specification, a sort of a formal grammar, for the class in terms of basic constructions. (This general method has been carefully explained in the fundamental work of F. Chyzak, Ph. Flajolet and B. Salvy ([5])).

In this framework, classes of combinatorial structures are defined either iteratively or recursively in terms of simpler classes by means of a collection of elementary combinatorial constructions. The approach followed resembles the description of formal languages by means of context-free grammars, as well as the construction of structured data types in classical programming languages.

The approach developed here is direct, more "symbolic", as it relies on a specification language for combinatorial structures. It is based on so-called admissible constructions that have the important feature of admitting direct translations into generating functions. We specifically examine constructions whose natural translation is in terms of ordinary generating functions.

The ordinary generating function (OGF) of a sequence $\{A_n\}$ is, we recall, $A(z) = \sum_{n=0}^{\infty} A_n \cdot z^n$.

Definition 13 (Admissible Constructions) *Assume that Φ is a binary construction that associates to two classes of combinatorial structures B and C a new class $A = \Phi(B, C)$ in a finite way (each A_n depends on finitely many of the B_n and C_n). The Φ is admissible iff the counting sequence $\{A_n\}$ of A is a function of the counting sequences $\{B_n\}$ and $\{C_n\}$ of B and C only: $\{A_n\} = \Xi[\{B_n\}, \{C_n\}]$. In that case, there exists a well defined operator Ψ relating the corresponding ordinary generating functions: $A(z) = \Psi[B(z), C(z)]$.*

In this work, we will basically use three important constructions: union, product and sequence, which we describe below.

Definition 14 (Union Construction) *The disjoint union A of two classes B, C is the union (in the standard set-theoretic sense) of two disjoint copies, B^o and C^o, of B and C. Formally, we introduce two distinct "markers" ϵ_1 and ϵ_2, each of size zero, and define the (disjoint) union $A = B + C$ of B, C by $B + C = (\{\epsilon_1\} \times B) \cup (\{\epsilon_2\} \times C)$. The ordinary generating function is clearly $A(z) = B(z) + C(z)$. We represent the disjoint union construction by* Union.

Definition 15 (Product Construction) *If construction A is the cartesian product of classes B and C ($A = B \times C$), then, considering all possibilities, the counting sequences corresponding to A, B, C are related by the convolution relation: $A_n = \sum_{k=0}^{n} B_k \cdot C_{n-k}$ and the ordinary generating function is clearly $A(z) = B(z) \cdot C(z)$. We represent the product construction by* Prod.

Definition 16 (Sequence Construction) *If C is a class of combinatorial structures then the sequence class $\mathcal{G}\{\mathcal{C}\}$ is defined as the infinite sum $\mathcal{G}\{\mathcal{C}\} = \{\epsilon\} + \mathcal{C} + (\mathcal{C} \times \mathcal{C}) + \cdots$ with ϵ being a "null structure", meaning a structure of size 0. The null structure plays a role similar to that of the empty word in formal language theory and the sequence construction is analogous to the Kleene star operation (C^*). The ordinary generating function is clearly given by $A(z) = 1 + B(z) + B^2(z) + \cdots = \frac{1}{1-B(z)}$ where the geometric sum converges in the sense of formal power series since $[z^0]B(z) = 0$. We represent the sequence construction by* Sequence.

2.2.2 Avoiding Permutations. An avoiding permutation of length ℓ is a sequence $\tau = [\tau_1, \tau_2, \ldots, \tau_\ell, \tau_{\ell+1}]$ that is a permutation of $[1, \ldots, \ell + 1]$ and that satisfies the following conditions: $\tau_1 = 1$, $\tau_{\ell+1} = \ell + 1$, and $\tau_{i+1} - \tau_i \neq \pm 1$ for all i such that $1 \leq i \leq \ell$. Clearly, such a permutation encodes a path from 1 to $\ell + 1$ that has no edge in common with the graph $(1, 2, \ldots, \ell + 1)$. The parameter $\ell + 1$ is referred to as the size. There is no avoiding permutation for sizes $2, 3, 4, 5$. Surprisingly, the first nontrivial configurations occur at size 6, where the 2 possibilities are $[1,4,2,5,3,6]$ and $[1,3,5,2,4,6]$, while for size 7, there are 10 possibilities

$$[1,3,6,4,2,5,7],[1,3,5,2,6,4,7],[1,4,6,2,5,3,7],[1,4,2,6,3,5,7],[1,5,3,6,4,2,7],$$
$$[1,5,3,6,2,4,7],[1,5,2,4,6,3,7],[1,4,6,3,5,2,7],[1,6,3,5,2,4,7],[1,6,4,2,5,3,7].$$

The goal in this subsection is to determine the number Q_n of avoiding permutations of size n (that is, of length $n-1$). We prove:

Theorem 24 *Avoiding permutations have ordinary generating function*

$$Q(z) := \sum_n Q_n z^n = \frac{\left(z - 1 + e^{\frac{1+z}{z(z-1)}}\, \mathit{Ei}\left(1, \frac{1+z}{z(z-1)}\right)\right) z}{(z-1)(1+z)} =$$
$$\frac{z\left(z + z\, \mathit{hypergeom}\left([1,1],[\,],-\frac{z(z-1)}{1+z}\right) + 1\right)}{(1+z)^2}$$

where *Ei* is the exponential integral and *hypergeom* represents the hypergeometric series. Equivalently, Q_n is expressible as a double binomial sum:

$$Q_{n+2} = (-1)^{n-1} + \sum_{k_2=0}^{n} \sum_{k_1=0}^{n-k_2} (-1)^{k_1+k_2} (n-k_1-k_2)! \binom{n-k_1-k_2}{k_1} \binom{n+1-k_1}{k_2}$$

Proof: By the inclusion-exclusion principle, we need to determine the number $F_{n,j}$ of permutations $[\tau_1 = 1, \tau_2, \ldots, \tau_{n-1}, \tau_n = n]$, with at least j "exceptions", among which j distinguished, that are successions of values of the form $\tau_j - \tau_{j-1} = \pm 1$. The number of permutations with no exception is then:

$$Q_n = \sum_{j=0}^{n-1} (-1)^j F_{n,j} \tag{1.6}$$

A permutation with exceptions includes a subcollection of "exceptional" edges that belong to the graph with edges $(1,2),(2,3),\ldots,(n-1,n)$. If we scan from left to right and group such exceptions by blocks, we get a *template*; a template thus represents a possible pattern of exceptional edges.

A template can be defined directly as made of blocks that are either: (i) isolated points; (ii) contiguous unit intervals oriented left to right (LR); (iii) contiguous unit intervals oriented right to left (RL). There is the additional constraint that the first and last blocks cannot be of type RL. For instance, for $n = 13$, the template $[\,[1,2,3],\ [4],\ [5,6],\ [7],\ [8],\ [11,10,9],\ [12,13]\,]$ will correspond to any permutation that has successions of values (in the cycle traversal) $1,2\,;\ 2,3\,;\ 5,6\,;\ 11,10\,;\ 10,9\,;\ 12,13$ as distinguished exceptions to the basic constraint of avoiding permutations.

We next provide the combinatorial specification for avoiding permutations. Let $\{a, b\}$ be a binary alphabet. We now describe the grammar of templates. The collection of strings beginning and ending with a letter a is described by the following rule:

sp0 := *S* = Prod(Sequence(Prod(a, Sequence(b))), a)

(It suffices to decompose according to each occurrence of the letter a). Now, the thre types of blocks in a template are described by the following rules:

sp1 := Prod(*begin_blockP, Z, end_blockP*)
sp2 := Prod(*begin_blockLR, Z,* Sequence(Prod(*mu_length, Z*), $1 \leq$ *card*), *end_blockLR*)
sp3 := Prod(*begin_blockRL,* Sequence(Prod(*mu_length, Z*), $1 \leq$ *card*), *Z, end_blockRL*)

Clearly, $sp2$ and $sp3$ are combinatorially isomorphic. For reasons related to the application of the inclusion-exclusion argument, we keep track of the size (number of nodes $l = 1$ of the basic interval graph, denoted by *card*) as well as of the length of blocks and of their *LR* or *RL* character, denoted by *mu_length*. Then, the grammar of templates is completed by substituting into $sp0$

$$a = \text{Union}(sp1, sp2) \quad \text{and} \quad b = sp3$$

Let $F_{n,k,j}$ be the number of templates with size n, k blocks and j exceptional edges. Then, counting the number of ways of linking blocks together, yields:

$$F_{n,j} = \sum_k F_{n,k,j}\, \phi(k) \tag{1.7}$$

where $\phi(k)$ is the Gamma integral:

$$\phi(k) \equiv (k-2)! = \int_0^\infty e^{-u}\, u^k\, \frac{du}{u^2}$$

for $k \geq 2$, and $\phi(1) = 1$ (since any such linking is determined by an arbitrary permutation of the $k-2$ intermediate blocks). Observe that the extension of ϕ by linearity to an arbitrary series $\mathrm{h}(u)$ in u is given by

$$\phi(\mathrm{h}(u)) = \int_0^\infty \frac{e^{-u}\left(\mathrm{h}(u) - (u-u^2)\left(\frac{\partial}{\partial u}\mathrm{h}(u)\right)_{u=0}\right)}{u^2}\, du$$

That is to say, we just replace in expansions $u \to u^2$ and apply the Euler integral

$$\int_0^\infty e^{-u} u^k du = k!$$

Thus, with $\mathrm{F}(z, u, v) = \sum F_{n,k,l}\, z^n u^k v^l$,the OGF $\mathrm{Q}(z) = \sum Q_n z^n$ satisfies

$$\mathrm{Q}(z) = \phi(\mathrm{F}(z, u, -1))$$

Thus, from (1.6) and (1.7) above, everything boils down to obtaining the $F_{n,k,j}$.

Template enumeration. The approach to determining the sequence $F_{n,k,j}$ consists in introducing the trivariate GF, which immediately results from the above combinatorial specification.

$$F(z,u,v) = \sum_{n,k,\ell} F_{n,k,\ell}\; z^n u^k v^l$$

There, z records size, u records the total number of blocks (needed for subsequent permutation enumerations since blocks should be chained to each other), and v records the total length of *LR* or *RL* blocks (the number of distinguished exceptions needed for inclusion-exclusion).

We now carefully employ the generating functions for the union, product and sequence constructions in the grammar rules of the combinatorial specification for avoiding permutations defined above.

The set of words made of a's and b's that start and end with an a is described symbolically by

$$W = \frac{1}{1-\frac{a}{1-b}} \cdot a \tag{1.8}$$

This is because $(1-f)^{-1} = 1 + f + f^2 + f^3 + \cdots$ generates symbolically all sequences of objects of type f. Thus, W represents a sequence of objects of type $\frac{a}{1-b}$ that start with an a. On the other hand, $\frac{a}{1-b}$ represents a sequence of objects of type b that end with an a. Take now the three types of blocks: isolated, LR, and RL. The GF's are, respectively, z, $LR(z) = z^2/(1-z)$, $RL(z) = z^2/(1-z)$. This is because isolated points are always of size 1, while LR and RL objects must be of size at least 2 (we have thus to multiply with z^2). Since the first and the last blocks can only be isolated points or LR blocks, the univariate GF for blocks is obtained by substituting a by $z + LR$ (isolated point or LR block) and b by RL in W. Thus we get the following trivariate GF:

$$\begin{aligned} F(z,u,v) &= \left(1 - \frac{u\left(z+\frac{z^2v}{1-vz}\right)}{1-\frac{uz^2v}{1-vz}}\right)^{-1} \cdot u\left(z + \frac{vz^2}{1-vz}\right) = \\ &= -\frac{uz(-1+vz+uz^2v)}{1-2vz-uz+v^2z^2+v^2z^3u} \end{aligned}$$

Path counting. For the inclusion-exclusion argument, it is easy to observe that the desired sum $\sum_{n,k,\ell} F_{n,k,\ell}\; z^n u^k (-1)^\ell$ corresponds to the specialization $F(z,u,-1)$. This yields:

$$F(z,u,-1) = -\frac{uz(-1-z-uz^2)}{1+2z-uz+z^2+z^3u} \tag{1.9}$$

Application of the ϕ-transformation (that counts the number of ways to connect the blocks) requires the modified form and so we get:

$$F(z,u,-1) = \frac{zu^2(uz^2+2z-uz+1)}{(1+z)(uz^2+z-uz+1)}$$

The corresponding ordinary generating function is

$$Q(z) = \int_0^\infty \frac{z(uz^2+2z-uz+1)}{(1+z)(uz^2+z-uz+1)} \cdot e^{-u}\,du$$

The quantity $Q(z)$ can be expressed in terms of the exponential integral

$$\int_0^\infty e^{-u}\,u^k\,du = k!$$

in the following closed form

$$Q(z) := \frac{\left(z-1+e^{\frac{1+z}{z(z-1)}}\,\mathrm{Ei}\left(1,\frac{1+z}{z(z-1)}\right)\right)z}{(z-1)(1+z)}$$

Since one deals with ordinary generating functions, this is to be taken as a formal (asymptotic) series. Note also that the exponential integral (Ei) involves the divergent series of factorials $\sum_{n=0}^{\infty} n!(-y)^{(-n-1)}$ which is also a hypergeometric series. This gives rise to a general conversion procedure from exponential integrals to hypergeometric forms. Hence, another closed form for the OGF of the Q_n is

$$Q(z) := \frac{z\left(z+z\,\text{hypergeom}\left([1,1],[\,],-\frac{z(z-1)}{1+z}\right)+1\right)}{(1+z)^2}$$

Thus, with $\mathrm{F}(z,u,v) = \sum F_{n,k,l}\,z^n u^k v^l$ and for OGF $\mathrm{Q}(z) = \sum Q_n z^n$, by recalling that $\mathrm{Q}(z) = \phi(\mathrm{F}(z,u,-1))$ we get the expression for $Q(z)$ as stated in the Theorem. The expression can then be expanded using the binomial theorem, and double combinatorial sums result for coefficients. □

Though they have no direct bearing on the graph problem at hand, we mention two interesting consequences of this theorem.

Corollary 2 *The quantities Q_n satisfy the recurrence*

$$(n+1)Q_n + Q_{n+1} - 2nQ_{n+2} + 4Q_{n+3} + (n+3)Q_{n+4} - Q_{n+5} = 0,$$

where $Q(0) = 0$, $Q(1) = 1$, $Q(2) = Q(3) = Q(4) = Q(5) = 0$ and the asymptotic estimate

$$\frac{Q_n}{(n-2)!} = e^{-2}\left(1+O\left(\frac{1}{n}\right)\right)$$

Proof: To get the recurrence relation, we use the following holonomic descriptions (introduced by Zeilberger), that is sequences that satisfy linear recurrences with polynomial coefficients:

$$\left(z^4 + z^5 + 4z^3 - 1 - z + 4z^2\right)\mathrm{Y}(z) + \left(-2z^4 + z^2 + z^6\right)\left(\frac{\partial}{\partial z}\mathrm{Y}(z)\right) -$$

$$-z^4 - 2z^3\_C_0 - 2z^3 - z^4\_C_0 + \_C_0 z^2 - z^5 + z - z^2 = 0$$

where $Y(z) = Q(z)$. By putting $C_0 = 1$, we get:

$$(z^4+z^5+4z^3-1-z+4z^2)\mathrm{Y}(z)+(-2z^4+z^2+z^6)\left(\frac{\partial}{\partial z}\mathrm{Y}(z)\right)-2z^4-4z^3-z^5+z=0$$

We can now get (by elementary properties of the z-transform) the following transformation to a linear recurrence:

$$u(0)=0,\ u(1)=1,\ u(2)=0,\ u(3)=0,\ u(4)=0,\ u(5)=0,$$

$$(n+1)u(n)+u(n+1)-2n\,u(n+2)+4u(n+3)+(n+3)u(n+4)-u(n+5)=0$$

We note that this provides an algorithm that uses a linear number of arithmetic operations to determine the quantities Q_n. By using the following principle based on the generating function method:

$$coeff_{z^n}\text{hypergeom}\left([1,1],[\,],z+dz^2+\mathrm{O}\left(z^3\right)\right) = n!\,e^d(1+\mathrm{o}(1))$$

provided that the argument of the hypergeometric is a function that is analytic at the origin, we have proved

$$Q(z) = \frac{z\left(z + z\,\text{hypergeom}\left([1,1],[\,],-\frac{z(z-1)}{1+z}\right)+1\right)}{(1+z)^2}$$

and since

$$-\frac{z(z-1)}{1+z} = z - 2z^2 + 2z^3 - 2z^4 + 2z^5 + \mathrm{O}(z^6)$$

we have proved that the asymptotic proportion of legal permutations is *exactly* equal to e^{-2}. $\Box$

The recurrence above implies the non-obvious fact that the number of avoiding permutations Q_n are computable in linear time. The asymptotic estimate extends properties known for permutations with excluded patterns (e.g., derangements have asymptotic density e^{-1}). Consequently, a nonzero proportion (about 13.53%) of all permutations that start with 1 and end with n are avoiding.

2.2.3 Avoiding Paths. We consider now the problem of counting the number $Q_{n,j}$ of avoiding paths of type (n, j), where n is the size (the number of nodes) and j is the number of "outer nodes". Such avoiding paths are defined by the fact that they satisfy the basic constraints of avoiding permutations regarding the base line $(1, 2, \ldots, n)$, but contain in addition j outer nodes taken to be indistinguishable (unlabelled) and conventionally represented by the symbol '$\star$'. For instance, for types $(n, j) = (3, 1), (4, 1), (4, 2)$, the listings are respectively

$$\{[1, \star, 3]\} \qquad \{[1, 3, \star, 4], [1, \star, 2, 4]\} \qquad \{[1, \star, \star, 4]\}$$

Theorem 25 *The number of avoiding paths is expressible as*

$$Q_{n+2,j} = \sum_{k_2=0}^{n-j} \sum_{k_1=0}^{n-j-k_2} (-1)^{k_1+k_2} (n - k_1 - k_2)! \binom{n-j-k_1-k_2}{k_1} \cdot \\ \cdot \binom{n-j+1-k_1}{k_2} \binom{n-k_1-k_2}{j}^2$$

(where $j \geq 0$)

Proof: We first define templates on which an inclusion-exclusion argument is applied. The specifications are a simple modification of the templates associated to avoiding permutations.

Let $\{a, b, x\}$ be a ternary alphabet. We now define the grammar of templates.

The collection of strings beginning with a and containing only one x that occurs at the end is described by the rule:

sp0 := *S* = Prod(Sequence(Prod(a, Sequence(b))), x)

(It suffices to decompose according to each occurrence of the letter a). We first need so-called "outer points" that are taken from outer space:

Outerpoints := Sequence(Prod(*Z, mu_outerpoint*))

We also need "inner points":

Innerpoints := Sequence(Prod(*Z, mu_innerpoint*))

Size is defined as the cumulative number of points in the pair of paths that underlies an avoiding path in the sense above: it is thus equal to the length of the avoiding path plus the number of $\star$ symbols corresponding to the outer nodes. We thus introduce a special notation for nodes of the integer line that are shared by the two paths:

$Z2$:= Prod(Z, Z)

Now, the three types of blocks are described by the following rules:

$sp1$:= Prod(*mu_block, Z2, Outerpoints, Innerpoints*)
$sp2$:= Prod(*mu_block, Z2,* Sequence(Prod(*mu_length, Z2*), card $\geq$ 1),
Outerpoints, Innerpoints)
$sp3$:= Prod(Sequence(Prod(*mu_length, Z2*), card $\geq$ 1), *Z2, Outerpoints,*
Innerpoints, mu_block)

(Clearly, $sp2$ and $sp3$ are combinatorially isomorphic). The blocks that can occur at the end are of type x and can only be of type $sp1$ or $sp2$ but without outer points nor inner points.

$sp1x$:= Prod(*mu_block, Z2*)
$sp2x$:= Prod(*mu_block, Z2,* Sequence(Prod(*mu_length, Z2*), $1 \leq card$))

The above grammar is completed (to give S) by substituting into $sp0$

$$\begin{aligned} a &= \text{Union}(sp1, sp2) \\ b &= sp3 \quad \text{and} \\ x &= \text{Union}(sp1x, sp2x) \end{aligned}$$

The 5-variate GF immediately results from the above specification:

$$\mathrm{F}(z,u,v,w_1,w_2) := \frac{-u\,z^2(-1+z\,w_2+z\,w_1-z^2\,w_1\,w_2+v\,z^2-v\,z^3\,w_2-v\,z^3\,w_1+v\,z^4\,w_1\,w_2+u\,z^4\,v)}{1-z(w_2+w_1)+z^2(w_1\,w_2-u)+v\,z^2(-2+2z\,w_1-2\,z^2\,w_1\,w_2+v\,z^2(1-z\,w_2-z\,w_1+z^2\,w_1\,w_2+z^2\,u))}$$

where u, v, w_1, w_2 represent the blocks, the length, the outer nodes and the inner nodes, respectively.

For inclusion-exclusion, we set $v = -1$. Application of the ϕ-transformation (that counts the number of ways to connect the blocks) requires the modified form

$$F(z,u,-1,w_1,w_2) := \frac{u^2\,z^2(1+2z^2-z^3\,w_1+u\,z^4+z^4\,w_1\,w_2-z^3\,w_2-u\,z^2-z\,w_2-z\,w_1+z^2\,w_1\,w_2)}{(1+z^2)(z^4\,w_1\,w_2+u\,z^4-z^3\,w_2-z^3\,w_1+z^2\,w_1\,w_2+z^2-u\,z^2-z\,w_2-z\,w_1+1)}$$

The ordinary generating function is here

$$Q(z) := \int_0^\infty \frac{z^2(1+2z^2-z^3\,w_1+u\,z^4+z^4\,w_1\,w_2-z^3\,w_2-u\,z^2-z\,w_2-z\,w_1+z^2\,w_1\,w_2)\,e^{-u}}{(1+z^2)(z^4\,w_1\,w_2+u\,z^4-z^3\,w_2-z^3\,w_1+z^2\,w_1\,w_2+z^2-u\,z^2-z\,w_2-z\,w_1+1)}\,du$$

And this can be expressed in terms of the exponential integral as follows:

$$Q(z) := z^2 \cdot e^{-\frac{(1+z^2)(w_2+w_1)}{z(z-1)(1+z)}} \cdot$$

$$\cdot \frac{\mathrm{Ei}\left(1, \frac{z^4\,w_1\,w_2-z^3\,w_2-z^3\,w_1+z^2\,w_1\,w_2+z^2-z\,w_2-z\,w_1+1}{z^2(z^2-1)}\right) e^{\frac{(1+z^2)(z^2\,w_1\,w_2+1)}{z^2(1-z^2)}}}{(z^2-1)(1+z^2)} +$$

$$+z^2 \cdot e^{-\frac{(1+z^2)(w_2+w_1)}{z(z-1)(1+z)}} \cdot \frac{z^2 e^{\frac{(1+z^2)(w_2+w_1)}{z(1-z^2)}} - e^{\frac{(1+z^2)(w_2+w_1)}{z(1-z^2)}}}{(z^2-1)(1+z^2)}$$

Again, there is an "explicit form" of the OGF of the problem

$$Q(z) := z^2 \cdot$$

$$\cdot \frac{z^3(z\,w_1\,w_2 - w_2 - w_1) + z^2 \text{ hypergeom}\left([1,1],[\,],-\frac{z^2\,(z-1)(1+z)}{(1+z^2)(z\,w_2-1)(z\,w_1-1)}\right)}{(1+z^2)^2(z\,w_2-1)(z\,w_1-1)} +$$

$$+z^2 \cdot \frac{z^2(1+w_1\,w_2) - z(w_2+w_1)+1}{(1+z^2)^2(z\,w_2-1)(z\,w_1-1)}$$

and also

$$Q(z) := \frac{z^4 \text{ hypergeom}\left([1,1],[\,],-\frac{z^2\,(z-1)(1+z)}{(1+z^2)(z\,w_2-1)(z\,w_1-1)}\right)}{(1+z^2)^2(z\,w_2-1)(z\,w_1-1)} + \frac{z^2}{1+z^2}$$

The coefficient $c(n,j,k)$ of $z^n\,{w_1}^j\,{w_2}^k$ is obtained by straight expansion and avoiding paths are then enumerated by $C(n,j) = c(n,j,j)$. The corresponding formulae of the Theorem statement are obtained directly by symbolic expansions.

The computations are rather intensive and, for instance, the 4-variable GF that "lifts" $F(z,u,-1)$ is found to be

$$\frac{u\,z^2\left(1 - z\,w_2 - z\,w_1 + z^2\,w_1\,w_2 + z^2 - z^3\,w_2 - z^3\,w_1 + z^4\,w_1\,w_2 + u\,z^4\right)}{(1+z^2)\left(z^4\,w_1\,w_2 + u\,z^4 - z^3\,w_2 - z^3\,w_1 + z^2\,w_1\,w_2 + z^2 - u\,z^2 - z\,w_2 - z\,w_1 + 1\right)} \tag{1.10}$$

It is to be noted that computations have been performed with the help of the computer algebra packages `Combstruct` and `Gfun` that are dedicated to automating computations in combinatorial analysis and have been developed in the Maple system for symbolic computation. □

2.3 AVERAGE-CASE ANALYSIS FOR THE RANDOM GRAPH MODEL

We show now how to estimate the robustness to link failures in a random graph that obeys the $G_{n,p}$ model. An *avoiding pair* of length ℓ in a graph is an *unordered* pair of paths, each of length ℓ, with a common source and a common destination, that may share some nodes, but are edge disjoint. We have:

Theorem 26 *The mean number of avoiding pairs of length ℓ between a random source and a random destination in a random graph obeying the $G_{n,p}$ model is*

$$N_\ell(n,p) := \frac{p^{2\ell}}{2n(n-1)} \sum_{j=0}^{\ell} Q_{\ell+1,j} \binom{n}{l+1+j} (l+1+j)!$$

where the coefficients $Q_{n,j}$ are given by Theorem 25.

Proof: The coefficient $1/2$ corresponds to the fact that one takes unordered pairs of paths; the coefficient $1/(n(n-1))$ averages over all possible sources and destinations; the factor $p^{2\ell}$ provides the edge weighting corresponding to $G_{n,p}$; the arrangement numbers account for the number of ways to embed an avoiding path into a graph by choosing certain nodes and assigning them in some order to an avoiding path; the coefficients $Q_{\ell+1,j}$ provide the basic counting of avoiding paths that build up avoiding pairs. $\square$

Note: Since the $G_{n,p}$ model implies isotropy, the quantity $N_\ell(n,p)$ is also the mean number of avoiding pairs between *any fixed* source and destination s,t.

Robustness.. A short table of initial values of $N_\ell(n,p)$ follows:

$$N_2 = \tfrac{1}{2}(n-2)(n-3)p^4 \qquad N_3 = \tfrac{1}{2}(n-2)(n-3)^2(n-4)p^6$$
$$N_4 = \tfrac{1}{2}(n-1)(n-2)(n-3)(n-4)(n-5)^2p^8$$
$$N_5 = \tfrac{1}{2}(n-2)(n-3)(n-4)(n-5)^2(n^3-11n^2+25n+32)p^{10}$$

From developments in the previous section, the formulæ are computable in low polynomial time (as a function of ℓ). They make it possible to estimate the mean number of avoiding pairs in graphs of a given size for all reasonable values of n,p,ℓ. Take for instance a graph with $n=10^5$ nodes and an edge probability $p=5\cdot 10^{-5}$. This corresponds to a mean node degree that is extremely close to 5, so that, on average, each node has 5 neighbors. Then the mean values are

$$N_2 = 3.1\cdot 10^{-6},\ N_3 = 7.8\cdot 10^{-5},\ N_4 = 1.9\cdot 10^{-5},\ N_5 = 4.8\cdot 10^{-4},\ N_6 = 1.2\cdot 10^{-2}$$
$$N_7 = 0.30,\ N_8 = 7.6,\ N_9 = 190,\ N_{10} = 4763,\ N_{11} = 119062,\ N_{12} = 2.9\cdot 10^7$$

Thus, in this example, one expects to have short and multiple connections between source and destination provided paths of length 8 are allowed. This numerical example also shows that there are rather sharp transitions. The formula of Theorem 26, that entails the following rough approximation

$$N_\ell(n,p) \approx \frac{1}{2}\ n^{2\ell-2}p^{2\ell} \tag{1.11}$$

precisely accounts for such a sharpness phenomenon.

In the introduction, we have defined ℓ-robustness as multiple connectivity by edge-disjoint paths of length *at most* ℓ. In fact, Equation 1.10 leads to explicit expressions for generalized avoiding pairs of type (ℓ_1,ℓ_2) that are made of two paths, of lengths ℓ_1, ℓ_2. It can then be seen that the bottleneck for existence of pairs (ℓ_1,ℓ_2) with $\ell_1,\ell_2 \le \ell$ is in fact the case (ℓ,ℓ). Thus, since $N_\ell(n,p)\to 0$ when $\frac{p}{p_r(n,\ell)\to 0,}$ the function

$$p_r(n,\ell) = 2^{\frac{1}{2\ell}} n^{-1+\frac{1}{\ell}}$$

is a cut-off point for ℓ-robustness and an $(\leq \ell, \leq \ell)$-avoiding pair is expected or not depending on whether p/p_r tends to 0 or to ∞.

Corollary 3 *Any fixed pair in a $G_{n,p}$ graph is almost certainly not $\ell - robust$ if $p/p_r(n) \to 0$.*

Proof: When $\frac{p}{p_r(n,\ell)} \to 0$, then the expected number $N_\ell(n,p)$ of the desired pairs of paths tends to 0 and so does the probability of existence of at least one such pair of paths (since this probability, by Markov Inequality, is bounded from above by the expectation). Thus, with probability tending to 1, there is no pair of edge disjoint paths between the two vertices and these two vertices are, almost certainly, not $\ell - robust$. □

2.4 THRESHOLDS IN THE RANDOM GRAPH MODEL

In this section we provide bounds for the probability (and thus the threshold, if it exists) of the existence, between any fixed pair of vertices, of two edge-disjoint paths of length at most ℓ, by proving the following:

- We give an estimation of the value $p_L \equiv p_L(n,\ell)$ such that $G_{n,p}$ graphs with $p \leq p_L$ do not satisfy the desired property of the existence, between any fixed pair of nodes, of two edge-disjoint paths between some pair of vertices, with probability tending to 1 as n goes to infinity.
- We present a value $p_U \equiv p_U(n,\ell)$ such that almost every $G_{n,p}$ graph with $p \geq p_U$ has almost all its source-destination pairs of vertices connected by at least two edge-disjoint paths of length at most ℓ.

Theorem 27 *Define*

$$P_L(n,\ell) = \left(\log \frac{n^2}{\log n}\right)^{\frac{1}{\ell}} n^{-1+\frac{1}{\ell}}$$

Then, for $p \leq P_L(n,\ell)$, almost surely, there exists a pair of vertices in the $G_{n,p}$ graph that does not have the $\ell - robustness$ property.

Proof: By using the threshold function for diameter. □

Theorem 28 *Define*

$$P_U(n,\ell) = 2\left(\log\left(n^2 \log n\right)\right)^{\frac{1}{\ell}} n^{-1+\frac{1}{\ell}}$$

Then, for $p \geq P_U(n,\ell)$, almost surely, almost all pairs of vertices of a $G_{n,p}$ graph have the $\ell - robustness$ property.

Proof: Consider two independent distributions G_{n,p_1} and G_{n,p_2} on the same set of vertices. Let $E_i (i = 1, 2)$ be the events "G_{n,p_i} has diameter ℓ".

Consider the graph $\tilde{G}$ obtained when we superimpose an instance $G' \in G_{n,p_1}$ and an instance $G'' \in G_{n,p_2}$ and OR them (i.e., $\tilde{G}$ has an edge joining u, v iff at least one of G', G'' has). Clearly $\tilde{G} \in G_{n,p}$ with

$$p = p_1(1 - p_2) + p_2(1 - p_1) + p_1 p_2 = p_1 + p_2 - p_1 p_2$$

In fact, if u, v are joined in G' by a path P_1 and in G'' by a path P_2, then these two paths both exist in $\tilde{G}$. For p around the threshold for diameter ℓ of $G_{n,p}$ and $\ell = o(n)$, the number of pairs u, v of $\tilde{G}$ for which the paths of G', G'' overlap is $o(n^2)$, thus the vast majority of pairs of vertices ($n^2 - o(n^2)$ of them) in $\tilde{G}$ is connected via two edge disjoint paths of length $\leq \ell$.

This gives approximately a $p_U \leq p_1 + p_2 - p_1 p_2$ and if $p_1 = p_2 = p_0^{(\ell)}$ (p_0 a threshold for diameter ℓ or $\ell + 1$) then

$$p_U \leq 2p_0^{(\ell)} - \left(p_0^{(\ell)}\right)^2 \leq 2(2 \log n - \log c)^{\frac{1}{\ell}} n^{\frac{1}{\ell} - 1}$$

from [10], where c can be adjusted so that the diameter is almost surely ℓ (see [10], Corollary 12, p. 237). $\square$

3. RANDOM WALKS AND EXPANDER PROPERTIES IN RANDOM REGULAR GRAPHS WITH EDGE FAULTS

3.1 INTRODUCTION

The study of random walks has many important applications to the design and analysis of randomized algorithms. The main reason is that random walks can be used to simulate the evolution of randomized algorithms, where certain typical questions about the walk (such as the expected number of steps to go from a given vertex to another or the expected number of steps to visit all vertices) are directly related to the expected time complexity of the simulated algorithm.

In order to facilitate the systematic study of random walks, we also employ an abstract representation of them, by using the powerful tool of Markov chains. Random walks and corresponding Markov chains are widely used in Computer Science. There is recently great interest in Markov Chains that are "rapidly mixing", that is they get close (in terms of the variance distance) to the limit distribution after a polylogarithmic number of steps, as a function of the total number of states of the chain (see e.g. [58, 67]). It is known that if G is an expander then the Markov chain associated with P is rapidly mixing. One important application of the rapid mixing property is in the almost uniform generation and approximate counting problems (see [67]).

We also note the close relation of certain important graph properties (such as connectedness, regularity, expansion) to the eigenvalues of the adjacency matrix of the graph. It is also well known that random regular graphs of the model G_n^d are almost surely efficient certifiable expanders. In this section, we investigate whether their partially destroyed due to edge faults counterparts (of the model $G_{n,p}^d$, see also section 1.8) have still expander properties that can be efficiently verified, by studying the second largest eigenvalue of their adjacency matrix. Recall (see also section 1.10) that $G_{n,p}^d$ has a giant component of small diameter, even when in the (worst) case of constant edge failures and constant degrees $d = O(1)$. Here we wish to determine the minimum necessary value of p for which the giant component of $G_{n,p}^d$ remains a certifiable expander with high probability. In the next section we show that the second eigenvalue of the adjacency matrix of the giant component of $G_{n,p}^d$ is concentrated in an interval of small width around its mean, and that its mean is $O((dp)^{1/2})$, provided that $dp > 33$ (in fact $dp > 31^{\frac{1}{1-\epsilon}}$, where $\epsilon > 0$ a very small constant. By appropriate tuning of parameters, we get $\epsilon = 0.0166$ and $dp > 32.86$). Thus, the giant component of a random member of $G_{n,p}^d$ remains, with high probability, a certifiable efficient expander, despite the link faults, provided that $dp > 33$. Recently, A. Goerdt ([35]) has continued the work of Nikoletseas and

Spirakis ([64]), using their model of random configurations and strengthening their results.

3.2 DEFINITIONS AND STATE OF THE ART

Let $G(V, E)$ be a graph of n vertices. We allow self-loops and multiple edges. Let A be its adjacency matrix. Its entries, $a_{i,j}$, are positive integers and count the number of edges from i to j (self-loops are counted twice). A is a $n \times n$ symmetric matrix. The graph G is $d-$regular if $\sum_j a_{i,j} = d$, for $1 \leq i \leq n$.

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ the eigenvalues of the graph $G(V, E)$. For any $d-$regular graph, $\lambda_1 = d$. The subdominant eigenvalues λ_2 and λ_n provide important information about expansion properties of G. For $d-$regular graphs, one wants the size of $\rho = max\{d^{-1}|\lambda_2|, d^{-1}|\lambda_n|\}$ to be as small as possible.

For the graph G it is sometimes useful to consider the matrix P with entries $p_{i,j} = \frac{1}{d_i} a_{i,j}$ where d_i is the degree of vertex i. The matrix P can be viewed as the transition matrix of the Markov chain of n states, associated with a random walk on the graph G.

If G is not connected, then the Markov Chain is separated into a number of aperiodic and irreducible chains, one per connected component.

There is considerable recent interest in Markov chains that are "rapidly mixing" i.e. they get close (in terms of the variance distance) to the limit distribution after a polylogarithmic number of steps, as a function of the total number of states of the chain (see e.g. [67]). It is known that if G is an expander then the Markov chain associated with P is rapidly mixing.

Definition 17 *A graph $G(V, E)$ is a $c-$expander if for every $X \subseteq V, |X| \leq n/2$ implies that $|N(X) - X| \geq c|X|$ where $N(X) = \{y|(x, y) \in E$ and $x \in X\}$. The constant c is called the expansion constant.*

Expanders are the building blocks of optimal networks and algorithms for many purposes such as sorting, routing, superconcentrators etc (see [65]). For $d-$regular graphs it is true that $c \geq \alpha(1 - \lambda_2/d)$ where $\alpha > 0$ is a constant.

Random $d-$regular graphs were shown to be almost surely very powerful expanders. Broder and Shamir ([14]) have shown that for most $d-$regular graphs, $|\lambda| \leq cd^{3/4}$ where $\lambda = max\{|\lambda_2|, |\lambda_n|\}$ and $c > 0$ a constant. Slightly better estimates were produced by ([30]). The above result leads to a quick randomized algorithm for generating certified efficient expanders:

Repeat
Construct a random member G of G_n^d ;
Evaluate its λ_2 ;
until a G is found with $\lambda_2 \leq cd^{3/4}$

By the result of ([14]), the expected number of repetitions is bounded.

Let now G be a random member of $G^d_{n,p}$. Let now λ_i be the eigenvalues of the adjacency matrix of the giant component of G. How does p affect λ_2? In fact, if the giant component of $G^d_{n,p}$ remains a certifiable efficient expander, despite edge faults, then this would lead to an efficient construction of robust fat-trees and other universal networks (see e.g. [55]). Here we prove that for almost any member of $G^d_{n,p}$, $|\lambda| \leq c(dp)^{1/2}$ for a constant $c > 0$. In fact $c^2 \leq \sqrt{33}$, provided $dp > 33$. Thus the efficient certifiable construction of robust expanders despite faults is indeed possible, when p is such that $dp > 33$.

3.3 THE MODEL AND THE GENERAL FRAMEWORK

The work of Nikoletseas and Spirakis ([64]) closely follows, but judiciously modifies and extends, the framework of Broder and Shamir ([14]).

We construct an undirected random $2d-$regular graph G^{2d}_n by choosing independently d permutations uniformly randomly among all possible permutations of the n vertices. For each of the chosen permutations π and for each vertex i ($1 \leq i \leq n$) we add to the graph the edge $(i, \pi(i))$ with probability p, independently. Then an outcome of $G^{2d}_{n,p}$ is produced where each vertex i has a degree $d_i \leq 2d$.

We recall (see also sections 1.9 and 1.10) the following results:

Fact 1 *$G^d_{n,p}$ is $d-$connected except for $O(1)$ vertices, with high probability, for all failure probabilities $f = 1 - p \leq n^{-\epsilon}$ ($\epsilon > 0$ fixed).*

Fact 2 *When $G^d_{n,p}$ is disconnected, it still has a giant (i.e. $\Theta(n)-$sized) connected component of $\Theta(\log n)$ diameter for any $f < 1 - \frac{32}{d}$, with high probability.*

These facts are proved for the (different) $d-$regular graph model of ([10]) by first showing the results in the configuration space with edge faults. A configuration H is a collection of n labeled groups (supervertices) each group having d labeled (small) vertices and a partition of the set of nd vertices into pairs called edges. We then remove each edge independently with probability p.

One can translate the proofs of Facts 1, 2 to the permutation model by appropriately replacing $k-$th group of n edges of the partition by $k-$th permutation. Furthermore, all the theorems proved in this paper for the permutation space hold also for the configuration space with the same mapping working in the opposite direction.

Let A be the incidence matrix of the giant component, C, of G (an outcome of $G^{2d}_{n,p}$) and P its Markov Chain matrix (irreducible, aperiodic) as before. Let

C have $n' = \Theta(n)$ vertices. In the case where the graph G is connected, we simply take as C the graph G itself and $n' = n$.

Let the eigenvalues of P be $\rho_1, \ldots, \rho_{n'}$ where we consider them sorted: $\rho_1 = 1 \geq \rho_2 \geq \cdots \geq \rho_{n'}$. Let $\rho = max\{\rho_2, |\rho_{n'}|\}$. P has real eigenvalues, because the Markov Chain it defines is reversible (see [67]).

Lemma 12 *The Markov Chain defined by P (i.e. a random walk on the giant component C of G) is (time) reversible.*

Proof : It is enough to show that for all $i, j \in \{1, \ldots, n'\}$, $p_{ij}\pi_i = p_{ji}\pi_j$ where $\{1, \ldots, n'\}$ are the vertices of C, π_i is the (steady state) probability of state i and $p_{i,j} = \frac{\alpha_{ij}}{d_i}$ is the probability of walking in one step from state i to state j. It is well known (see [73]) that $\pi_i = \frac{d_i}{2e}$ where e is the number of edges of C. Thus $p_{ij}\pi_i = \frac{\alpha_{ij}}{2e} = \frac{\alpha_{ji}}{2e} = p_{ji}\pi_j$ (because C is undirected and A is symmetric). $\square$

Thus, the eigenvalues of P are also those of a similar symmetric matrix (see [67]) and so are all real.

Since

$$trace\left(P^{2k}\right) = \sum_i \rho_i^{2k}$$

and since P has real eigenvalues we get:

$$\rho^{2k} \leq trace\left(P^{2k}\right) - 1$$

for $k \geq 1$. Thus, by taking expectations:

$$E(\rho) \leq \left(E\left(\rho^{2k}\right)\right)^{1/2k} \leq \left[E\left(trace(P^{2k})\right) - 1\right]^{1/2k} \tag{1.12}$$

by Jensen's inequality.

Let $P_{ii}^{(2k)}$ be, for the graph C, the probability that a particle starting at vertex i and moving by following the random walk defined by P, is again at vertex i after $2k$ steps. Note that the probability $P_{ii}^{(2k)}$ is a random variable, since it depends on the (randomly produced) graph. Clearly $E(P_{ii}^{(2k)})$ does not depend on i when C is taken over $G_{n,p}^{2d}$, due to uniformity. Thus

$$E\left(trace\left(P^{(2k)}\right)\right) = n'E\left(P_{11}^{(2k)}\right) = \Theta(n)\,E\left(P_{11}^{(2k)}\right) \tag{1.13}$$

where the expectation is over all graphs $G \in G_{n,p}^{2d}$, and for their giant components C.

We view G as produced in two steps: First we choose a random member $G_{reg} \in G_n^{2d}$. Then we construct G by selecting each edge of G_{reg} with probability p. Fix a G_{reg}.

In order to study P_{11}, first we formally represent the particle motion on the C of G produced from the G_{reg} by a sequence of moves $S = \sigma_1\sigma_2\cdots\sigma_{2k}$.

Each S is a word of length $2k$ in the free monoid M generated by the alphabet $A = \{f_1, f_1^{-1}, \ldots, f_d, f_d^{-1}\}$. Here the mapping f_i is "what remains" from $\pi(i)$ (out of which G_{reg} was constructed) after applying edge faults to edges of G_{reg} to get G. In fact f_i are "partial permutations" since some edges have been removed. Second, we assign to each f_i a partial permutation by choosing, for each vertex v in G, equiprobably one of its (remaining) neighbors. The sequence S then determines, for each vertex x, a trajectory starting at x: $T = (x, x_1, x_2, \ldots, x_{2k})$ where $x_1 = f_1(x), \ldots, x_i = f_i(x_{i-1})$.
In the free group F generated by the letters $\{f_1, \ldots, f_d\}$ S is just an element. Let 1_F (e.g. equal to $f_1 f_1^{-1}$) be the identity element of F.

Definition 18 *S is an identity sequence if $S \equiv 1_F$.*

Definition 19 *Given S, let $R(S)$ be the unique sequence of minimum length equivalent to S obtained by removing from S all identity subsequences. $R(S)$ is called the reduction of S.*

Definition 20 *Given S, if $S = R(S)$ then S is irreducible. If S^2 is irreducible, we call S strongly irreducible.*

Then for G:

Lemma 13 *$P_{11}^{(2k)}$ is the probability that the particle starting at vertex 1 either returns to 1 after $2k$ steps because S is an identity sequence or it does so because S reduces to some non-empty irreducible sequence S' having a trajectory $T(S')$ with $x'_{2s} = 1$, where $2s$ is the length of S').* □

We note that the above events are exclusive (the second event implies that S is not an identity sequence), thus $P_{11}^{(2k)}$ is the sum of their corresponding probabilities. We now bound from above these probabilities.

3.4 ESTIMATION OF $E(P_{11}^{(2K)})$ AND OF THE MEAN OF THE SECOND LARGEST EIGENVALUE

In the sequel, we consider vertex 1 belonging to the giant component C of G.

Lemma 14 *Let S be a random word of length $2k$ in M. Then, assuming $k = \Theta(\log n)$, $\Pr\{S$ is an identity sequence$\} \leq \left(\frac{31}{(dp)^{1-\epsilon}}\right)^{k/2}$.*

Proof: A balanced string of parentheses is equivalent to an identity sequence (just assign σ to every left parenthesis and σ^{-1} to the corresponding right and vice-versa). The number of balanced strings of parentheses is the Catalan number: $\frac{1}{2k+1}\binom{2k+1}{k}$. Let d_i be the degree of vertex x_i in G, for each vertex

x_i in the trajectory $T(S)$. Consider first G_{reg} fixed and let G be an outcome of the experiment of sampling edges out of G_{reg}.

Let $X_v(k)$ be the number of paths of length k starting from vertex v in the giant component C of G. The fraction, y, of the number of all identity sequences of length $2k$ over the number of all paths of length $2k$ in G, will be bounded by

$$y \leq \frac{1}{2k+1}\binom{2k+1}{k}\frac{X_1(k)}{X_1(2k)}$$

Let $c_1(k)$ be the number of all cycles of length k starting (and returning) to vertex 1 of C. Then, clearly

$$X_1(2k) \geq c_1(k)\, X_1(k)$$

Thus

$$y \leq \frac{1}{2k+1}\binom{2k+1}{k}\frac{1}{c_1(k)}$$

In order to estimate a lower bound on $c_1(k)$, we must consider the degrees of the vertices of C. Let $d_1, d_2, \ldots, d_k$ the degrees in a cycle including vertex 1. From the experiment in G_{reg} we have $E(d_i) = 2dp$ for each vertex separately, because the edges are sampled in independently (of course there are dependencies among neighbors). For a particular vertex i, then

$$\Pr\{d_i > 2dp(1-\beta)\} \geq 1 - \exp\left(-\frac{\beta^2}{3}2dp\right)$$

by Chernoff bounds in the Bernoulli of $2d$ experiments with success probability p. Fix a $\beta \in (0,1)$ and let $\gamma = 1 - \exp\left(-\frac{\beta^2}{3}2dp\right)$ ($0 < \gamma < 1$). We shall call a "success" the outcome "$d_i > 2dp(1-\beta)$" for the corresponding vertex i. The probability of success is γ.

Fix a cycle in C of length k and let $k = 2\alpha\log n$ ($\alpha > 0$ a constant to be determined). Consider on the cycle the $\frac{k}{2} = \alpha\log n$ vertices which are not neighbors. Then $c_1(k)$ is at least the number of cycles in the subgraph of C where all vertices have degree at least 2 except for the vertices i where a success happened, and they have degree at least $2dp(1-\beta)$. The number of such successes in a single cycle is at least $g(k) \geq \frac{k}{2}\gamma\,(1-\beta')$ (where $\beta' \in (0,1)$) with probability at least $1 - \exp\left(-\frac{\beta'^2}{3}\gamma\frac{k}{2}\right)$ by Chernoff bounds.

By choosing appropriate values for the constants β, β', α, we get high probability for having a number of successes which is at least $\frac{k}{2}$ minus a very small constant.

Lemma 15 *Let $\mathcal{E}$ be the event of "at least $\frac{k}{2}(1-\epsilon)$ successes", where $\epsilon = 1-\gamma(1-\beta') > 0$ a constant. By choosing appropriate values for the constants β, β', α, we can make ϵ as small as desired and get*

$$\Pr\{\mathcal{E}\} \geq 1 - \frac{1}{n}$$

(As an example, when $\beta = 0.5, \beta' = 0.01, \alpha \geq 30000$, then $\epsilon = 0.0166$).

(end of proof of lemma 15) $\square$

Thus, conditioning on event $\mathcal{E}$, we get, per cycle, at least $g(k) = \frac{k}{2}(1-\epsilon)$ vertices of degree $d_i \geq dp$ with high probability. So, $c_1(k) \geq [2dp(1-\beta)]^{g(k)}$ gives $c_1(k) \geq (dp)^{\frac{k}{2}(1-\epsilon)}$. We then have, by using also $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$, that

$$\Pr\{S \equiv 1_F|\mathcal{E}\} \leq \frac{1}{2k+1}\binom{2k+1}{k}\frac{1}{(dp)^{k/2(1-\epsilon)}} \leq \frac{5.5^k}{(dp)^{k/2(1-\epsilon)}}$$

Thus

$$\Pr[S \equiv 1_F|\mathcal{E}] \leq \left(\frac{5.5^2}{(dp)^{1-\epsilon}}\right)^{k/2}$$

When $(dp)^{1-\epsilon} > 31$ (in fact, $\frac{31}{(dp)^{1-\epsilon}} = 1-\epsilon'$, we get that

$$\Pr\{S \equiv 1_F|\mathcal{E}\} \leq n^{-x}$$

where $x > 0$ a constant. In fact $x = \alpha \log\left(\frac{1}{1-\epsilon'}\right)$.

We get the high probability result by removing the conditioning. Clearly,

$$\Pr\left\{S \equiv 1_F\right\} = \Pr\left\{S \equiv 1_F|\mathcal{E}\right\}\Pr\{\mathcal{E}\} + \Pr\left\{S \equiv 1_F|\bar{\mathcal{E}}\right\}\Pr\{\bar{\mathcal{E}}\} \leq \Pr\left\{S \equiv 1_F|\mathcal{E}\right\}\left(1-\frac{1}{n}\right)$$

for each G_{reg}, and by summing over all the G_{reg} (their total probability is 1) we get

$$\Pr[S \equiv 1_F] \leq \left(\frac{31}{(dp)^{1-\epsilon}}\right)^{k/2}$$

(end of proof of lemma 14) $\square$

Let S be a fixed sequence and let its trajectory be $T(S) = (1, x_1, \ldots, x_s)$. Clearly $x_1, \ldots, x_s$ are random variables depending on G and the assignment made to $f_1, \ldots, f_d$. We can view $x_1, \ldots, x_s$ constructed as follows: $\sigma_1(1)$ is chosen uniformly randomly in $\{1, \ldots, n\}$. (This does not depend on d or p since, if vertex 1 is connected to some vertex this could be any vertex in $\{1, \ldots, n\}$ equiprobably. Indeed, the regularity constraint on G_{reg} fixes for each vertex only the number of its neighbors. These neighbors are randomly selected and they clearly remain random after the faults, because faults sample edges randomly). For $i \geq 2$, if $\sigma_i(x_{i-1})$ is already fixed in S then $x_i \leftarrow \sigma_i(x_{i-1})$ else x_i is randomly chosen, etc. We call the former situation a forced choice, else a free choice. Because of the fact that these probabilities do not depend on either d or p, we can easily extend the corresponding lemmas proved in [14]:

Lemma 16 (extension of [14], Broder and Shamir) *Given any sequence S of length s, the probability that its $T(S)$ induces a subgraph of more than one loop is $O\left(\frac{s^4}{n^2}\right)$, provided $dp \geq 2$.* □

By the same argument we get:

Lemma 17 (extension of [14], Broder and Shamir) *For any irreducible sequence S of length s, the probability that its trajectory starting at 1 returns to 1 by traversing a single loop once is $\frac{1}{n} + O\left(\frac{s}{n^2}\right)$.* □

Next we prove the following upper bound on the probability of reductions of a given size.

Lemma 18 (extension of [14], Broder and Shamir) *The probability that induced loops are traversed at least twice (given that $R(S)$ is an identity) cannot be more than $\frac{k}{n} + O\left(\frac{k^2}{n^2}\right)$.* □

We also get that:

Lemma 19 *Let S be a sequence of length $2k$ chosen uniformly and randomly. The probability that S:*

1. *is not an identity and*
2. *the trajectory of S, $T(S)$ induces a subgraph of exactly one loop and*
3. *$T(S)$ returns to 1 by traversing the loop at least twice is*

$$O\left(2\,\frac{k}{n}\left(\frac{31}{(dp)^{1-\epsilon}}\right)^{k/2}\right)$$

□

By adding up we get the following basic theorem:

Theorem 29 *If $k = \alpha \log n$, $\alpha \geq 30000$ then*

$$E\left(P_{11}^{(2k)}\right) \leq \left(\frac{31}{(dp)^{1-\epsilon}}\right)^{k/2} + \frac{1}{n} + O\left(\frac{k^4}{n^2}\right) + O\left(2\,\frac{k}{n}\left(\frac{31}{(dp)^{1-\epsilon}}\right)^{k/2}\right)$$

□

Now, from relations 1.12 and 1.13 in previous section, we get

$$E(\rho) \leq \left[n \frac{31}{(dp)^{1-\epsilon}} \right)^{k/2} + \frac{1}{n} + O\left(\frac{k^4}{n^2}\right) + O\left(2\,\frac{k}{n}\left(\frac{31}{(dp)^{1-\epsilon}}\right)^{k/2}\right]^{1/(2k)}$$

$$\leq \left[n \left(\frac{31}{(dp)^{1-\epsilon}}\right)^{k/2} (1 + o(1)) \right]^{1/(2k)}$$

and by using a suitable k we have:

Corollary 4 *Let P be the Markov chain probability matrix of a random member of $G^{2d}_{n,p}$. Then the second largest eigenvalue of the adjacency matrix of the giant component satisfies*

$$E(\rho) \leq \left(\frac{31}{(dp)^{1-\epsilon}}\right)^{1/2} (1 + o(1))$$

□

Corollary 5 *The second largest eigenvalue, λ, of the adjacency matrix of $G^{2d}_{n,p}$ satisfies:*

$$E(|\lambda|) \leq (31)^{1/2} (dp)^{(1/2)(1-\epsilon)} (1 + o(1))$$

provided that $(dp)^{1-\epsilon} > 31$ □

Note that in the proof of Lemma 14 we required paths of length $2k$, $k \geq 30000 \log n$. Since $k < n$, our result is asymptotic and holds for $n > k \geq 30000 \log n$ i.e. $\frac{n}{\log n} \geq 30000$.

By using a martingale argument (for the technique of martingale sequences see [1]) we can also show sharp concentration of the eigenvalue around its mean, thus proving that the eigenvalue is bounded away from 1.

4. ROBUST DECENTRALIZED COMPUTATIONS THROUGH RANDOMIZATION[1]

4.1 INTRODUCTION

The issue of fault tolerance in the framework of Parallel and Distributed Computations, tries to capture phenomena where some nodes (or communication links) of a target parallel machine corrupt (become unavailable or stall) during the execution of a parallel algorithm. The issue has become very intriguing in last years, due to the demand for execution of parallel algorithms over arbitrary sets of machines that work as a whole. In addition, the necessity of exploiting off-the-self computational power has lead to the consideration of arbitrary environments of Parallel and Distributed Computing that may vary with time, according to the availability of their building blocks. Thus, it would be very interesting to devise techniques that use an environment which is prone to failures, for the emulation of a similar environment which is guaranteed to be fault-free during the execution of a parallel algorithm. Fault tolerance in the concept of parallel and distributed computations can be provided at various levels of such a decentralized computing environment:

- At machine level, where the underlying environment is actually a fixed network of processing elements that tries to overcome the corruption of a specific node (or edge). Such works are usually based on two major techniques: the first is the technique of embedding of an ideal parallel machine into a fixed, fault-tolerant underlying network of processing elements which can be fault-tolerant with only constant slowdown (e.g. in [54] it is showed that an N-node butterfly or shuffle-exchange network can emulate a fault free network of the same type and size, with only constant slowdown). The second technique for providing fault tolerance at machine level, is the technique of redundant computations. Albeit it seems that redundancy in parallel computations is rather a waste of computational power, this technique is proved as powerful as randomization in some cases, especially when we have to deal with static faults (see in subsection 4.1.1 the categorization of the fault occurrences).

- At cost model level, where the abstract machine model that is considered by the programmer, is itself fault-tolerant. In particular, the cost model tries to exploit the underlying realistic machine in such a way that the overall execution of the input algorithm will not be affected by the corruption of arbitrary processing elements at runtime. The major difference between this and the previous category of fault tolerance is that, in the latter category, the augmented cost models may become the middle-tier technologies between the actual (arbitrary) environments that provide parallel computation capabilities and the programming environments that need to consider general-purpose parallel architectures in order to implement parallel algorithms transferable from parallel machine to parallel machine. Most works of this kind focus on the PRAM cost model, which used to be the most popular model of Parallel Computations until the early nineties (e.g. [45, 46, 48, 49]). Recently some new works on fault tolerant versions of more realistic cost models than PRAM (such as BSP) have come up, which seem to provide general solutions for fault tolerant versions of these realistic cost models (see for example [52, 53]).

- At programming environment level, in which the programming environment itself takes over the responsibility to provide fault-tolerant primitives to the designers of parallel algorithms (e.g. synchronization operations, end-to-end safe communication, robustness of the storage scheme, agreement protocols among the live processors, etc.). Some of these works are [32, 22, 21].

4.1.1 Categorization of faults. The major distinction of fault-tolerance problems is based on the kind of the fault occurrences they consider. In the area of Parallel Computing, the prevailing model of faults is the fail-stop model, introduced by Kanellakis and Shvartsman ([46]), according to which whenever a processor dies it is excluded from the remaining simulation process. In [45] the processing elements are allowed to restart at arbitrary times (this is the so-called restartable fail-stop model). Of course, in that case, serious problems with the coordination of work might arise, which are usually dealt with by definite synchronization operations, or timestamping techniques.

Similarly, in the area of Distributed Computing we may have to come up with a large range of faults, from crash faults (which are equivalent to the fail-stop model), to omission faults, or even malicious[1] faults, where the faulty processors join forces to affect the rest of the simulation process). In most cases the crash faults case is considered, while the malicious faults case has to do mainly with issues such as virus attacks in parallel or distributed settings, secure storage in distributed environments, etc.

Additionally the faults may be classified as static (i.e. known at the beginning of the simulation process), or dynamic (i.e. they may occur at arbitrary points of the simulation process). Both these cases are quite interesting. More specifically, the static case reflects the adaptation of a certain cost model or parallel computing environment, over an unknown (but fixed from that point on) working environment. On the other hand, the problem of tolerating dynamic processor faults in pragmatic settings, can be seen as a {Safe Storage & Checkpoint & (dynamic) Scheduling} problem. More specifically, the issue of the fault tolerance in such environments can be modeled as a problem of scheduling sequential sets of independent jobs, on a dynamically changing working environment. Hence, our task is to reassign the amount of pending work because of new processor (or communication links) corruptions on a dynamically decreasing set of live processors, in such a way, that as long as there is at least a fraction of the initial processors still live, the total amount of work will be successfully executed. The challenge lies in proposing an efficient strategy that will achieve an almost work-preserving, robust execution of the input algorithm, and will also assure a balanced split of the work load among the operational processors. This strategy will also have to exploit a robust storage scheme that will tolerate arbitrary processor failures, and provide a periodic Checkpoint procedure to commit the work of the simulation process at runtime. Typical examples of this approach are [53, 21].

4.1.2 Fault tolerance on PRAM machines. There have been many works in the area of fault tolerance on the PRAM cost model, especially in early nineties ([49, 48, 45, 46, 18]). In these works the fail-stop model is adopted. The reason was that the PRAM model considers that the input algorithms are executed in a lock-step fashion, and the processing elements are totally synchronized by a global clock. Thus, there is no chance of dealyed action of a processing element, that might completely mislead the whole computation. On the other hand, the malicious (or byzantine) faults necessitate the coordination of work among the live processors via the robust implementation of agreement protocols, which are unnecessary in the PRAM case, due to the complete synchrony of the processing elements, and the shared-memory feature.

In [48] a general strategy for simulating arbitrary CRCW PRAM steps[2] over a setting that allows dynamic processor faults is provided, by solving a core problem for this cost model, the `Certified Write All` problem. In [49] the same problem is dealt with over the restartable fail-stop model, by using a combination of tentative computations (i.e. computations that are most likely to be correct) and definite computations (guaranteed computations against any sequence of fault occurrences). This approach achieves constant amortized slowdown per CRCW PRAM step, for many reasonable fault distributions. This is done by having the processors tentatively simulate the fault-free execution of the input algorithm, while a definite auditing procedure monitors the simulation process at specific points. In [46, 45] some strategies are provided for dealing with the `Write All` problem in the fail-stop, non-restartable and restartable cases, where the faults may occur dynamically during the execution of an input CRCW PRAM algorithm.

The reader is referred to the monograph of Paris Kanellakis and Alex Shvartsman ([44]) for an overview of the most important simulation strategies that deal with the issue of fault-tolerance on PRAM machines, and an excellent classification of the instances of fault tolerance on PRAM machines in the fail-stop model.

4.1.3 Fault tolerance on arbitrary machines. In the case of arbitrary computing environments that consist of processing elements communicating via an underlying network infrastructure, there are several crucial parameters other than the computational power provided by the machine, that affect the performance of fault-tolerant strategies. Such parameters are the latency of the communication infrastructure, the bandwidth per processor or the synchronization cost, that are not accounted for in the PRAM cost model. Thus, proprietary solutions for providing fault tolerance in such settings should be provided, or more appropriate models should be chosen, that better estimate the actual overhead of a simulation strategy.

In [22] the case of dynamic processor faults is considered over an arbitrary message-passing underlying computing environment of synchronous machines. In this setting, an optimal strategy is provided for executing a set of independent tasks. In this work it is stated that the core of any simulation strategy over a synchronous message-passing environment, is a CHECKPOINT routine of the remaining live processors, along with a BALANCED ALLOCATION strategy.

In [32], another primitive operation of distributed computing is considered, namely the BYZANTINE AGREEMENT. In this work the BYZANTINE AGREEMENT is provided against crash failures, within optimal message complexity. Then this primitive operation is used for the provision of a family of early stopping agreement protocols with improved message complexity, and a new solution to the CHECKPOINTING procedure that was provided in [22].

Another work that deals with matters of synchronization over computing environments of limited asynchrony is [33], where a general strategy for simulating a completely synchronous network by a network of limited asynchrony. Despite the fact that this work considers a totally reliable underlying network, it is interesting that it uses the notion of tentative computations and definite auditing (or checkpointing) procedures for the safe progress of the simulation process, which is a strategy that was exploited in many works of fault tolerance (e.g., [49, 53]).

Although such proprietary approaches achieve great efficiency (and in some cases optimality) in the general case of an arbitrary synchronous message-passing computing environment, they cannot exploit the feature of bulk synchrony provided by some new cost models (e.g. BSP, QSM) that seem to prevail in the area of Parallel Computation the last few years. It should also be noted that bulk synchrony (or limited asynchrony) is inherent in the parallel algorithms themselves in many cases, and this gives raise to the provision of bulk synchronous, fault-tolerant environments, that focus their power on other features than the continuous synchronization and agreement protocols, such as Adaptive Load Balancing and Robust Storage Schemes (e.g. see [53]).

4.1.4 Fault tolerance on BSP machines. The BSP cost model focuses mainly on the {computation & communication & bulk synchrony} cost during the execution of an algorithm, rather than on the continuous synchronization procedure of a completely synchronous setting, as in the cases of [32, 22]. Thus new strategies are necessitated that exploit this special characteristic of bulk synchrony and will provide fault tolerance on BSP machines.

In [52], the issue of fault tolerance over BSP machines has been addressed. Simulations for two different cases were considered. In the static case, the faulty or unavailable processors are already known at the beginning of the computation and no processor changes its status afterwards. On the other hand, in the (semi) dynamic case, each processor may fail or become unavailable with a fixed probability during the computation and remains so until the end of the computation; however, some critical periods during the computation where no processor was allowed to fail, could not be avoided. In this work some Monte Carlo constructions based on embedding of the Virtual BSP machine on the operational subset of the Real BSP machine (for the static case) and of work redundancy (for the semi-dynamic case) assure efficient executions of BSP algorithms over fault-prone BSP machines.

In [53], fault tolerant BSP computations under "fully dynamic processor faults" are considered. Namely, the faults may happen online at any point of the computation. To tackle the problem, the issue of the fault tolerance on BSP is modeled as an independent job scheduling problem, on a dynamically changing working environment. The goal is to choose an efficient strategy that will achieve an almost work preserving, robust execution of the BSP algorithm, and will also assure a balanced split of the work load among the operational processors. A modular and efficient simulation scheme is proposed, which, compared to an optimal offline adversarial computation under the same sequence of fault occurrences, achieves an $\mathcal{O}((\log n \cdot \log\log n)^2)$-factor times the optimal work. This scheme combines an ADAPTIVE LOAD BALANCING scheme with a MIXED STORAGE scheme (based on Rabin's Information Dispersal Algorithm, [66]) and a CHECKPOINTING procedure (that exploits a BSPAGREEMENT Protocol for periodic synchronization among the processing elements).

4.1.5 Fault tolerance on NETWORKS OF WORKSTATIONS. The NETWORKS OF WORKSTATIONS ([21, 4]) platform tries to satisfy the demand for construction of parallel systems using off-the-self workstations that actually deliver and even surpass the power and reliability of many large scale machines. It actually represents an inherently asynchronous (or bulk-synchronous) environment for the execution of parallel algorithms. It seems that a cost model such as QSM or BSP would be very easily applicable to this parallel setting, because it consists of processing elements communicating via a specific communication infrastructure, and operate in an asynchronous or bulk-synchronous mode. In the work of [21] the NETWORK OF WORKSTATIONS is modeled as a completely asynchronous multiprocessor, shared-memory system, augmented with a fault-tolerant mechanism that treats even the slow workstations as failed ones. The architecture of this system is centralized, in the sense that there are specialized processors that perform specific operations (i.e. there is one task manager that schedules the pending work, some processors that deposit the necessary information, and some workers that actually execute the tasks that are assigned to them). Nevertheless, as stated in this work, an optimal fault-tolerant strategy should minimize job migration and should be integrated into the parallel system itself. This work actually tries to exploit the techniques that have appeared in the literature for providing fault tolerance over PRAM, combined with robust storage schemes based on information dispersal techniques (see [66]). Of course fault tolerant strategies on more relevant cost models will be much more realistic.

In the sequel, we present as a case study a simulation strategy for handling processor failures on the BSP cost model, which is a bulk-synchronous model of parallel computations. This strategy has to face fully dynamic processor faults and a more complex approach is adopted, that combines a balancing scheme, a storage scheme and a checkpointing procedure.

4.2 CASE STUDY: ROBUST BSP COMPUTATIONS UNDER DYNAMIC PROCESSOR FAULTS

In this section we present as a case study the work of Kontogiannis, Pantziou, Spirakis and Yung for providing BSP simulations over faulty BSP computing environments ([53]).

4.2.1 The BSP cost model. The Bulk Synchronous Model (BSP) was introduced by L. Valiant ([73]) as a bridging model that tries to close the gap between the domains of decentralized (i.e. parallel or distributed, according to the terminology of [44]) architectures and parallel algorithms.

The applicability of the BSP cost model lies in the fact that, apart from the cost of the parallelism that is accounted by the traditional PRAM cost model, it also considers the communication and synchronization limitations that are imposed by the realistic decentralized architectures. Yet, it does not limit the interoperability of the model among different decentralized computing environments, by abstracting away from the designers detailed architectural features such as the topology of the processing elements, or the synchronization procedures. Thus, the objective of this model is to allow the design of parallel algorithms that can be efficiently executed on a variety of decentralized architectures, at a predictable cost, w.r.t. some architectural parameters that reflect the capabilities of the underlying decentalized machine.

A BSP algorithm $\mathcal{A}$ consists of a sequence of supersteps that are separated by Bulk Synchronization operations (SYNC in short). Each superstep consists of a Local Computation phase and a Communication Phase (LC-phase and Comm-Phase respectively). During an LC-phase each processor performs a sequence of operations on data held in its local memory at the beginning of the superstep, while Comm-phase takes over the transmission of the outgoing messages of each processing element to their destinations, through the underlying communication infrastructure. At the end of the superstep a SYNC operation indicates the end of the current superstep. A BSP machine consists of the following components:

- A collection of n identical processor/memory elements which are distinguished by their unique identification numbers.
- A communication infrastructure that takes over the point-to-point communication process. This infrastructure is characterized by the bandwidth parameter g and the latency parameter L (which are explained in the next paragraph).
- A barrier synchronization mechanism among the n processing elements.

The two major parameters of the decentralized architecture are the bandwidth g, i.e. the (per-processor) ratio of the total throughput of the whole system in terms of local computation operations, to the throughput of the underlying communication network in terms of words of information delivered, and the latency L, which is the minimum time interval between two consecutive SYNC operations.

Thus, the running time of a single superstep on the BSP cost model is characterized by the parameters $n, g,$ and L, and is given by the following formula:

$$T_{\text{superstep}} = \max\{L, T_{\text{LC}} + T_{\text{Comm}}\}, \tag{1.14}$$

where T_{LC} is the maximum (among the processing elements) cost for local computations, and T_{Comm} is the maximum time needed for transmitting all the outgoing messages to their target processors. If we consider that during the Comm-phase each processor sends and receives at most h $1-$word messages (i.e. an $h-$relation has to be implemented during the Comm-phase) then $T_{\text{Comm}} = g \cdot h$, according to W.F. McColl ([56]). In that case, we have:

$$T_{\text{superstep}} = \max\{L, T_{\text{LC}} + g \cdot h\} \tag{1.15}$$

Remark: In some cases the underlying machine charges the implementation of an $h-$relation as $g \cdot \max\{h, h_0\}$ for some h_0 that depends on the machine.

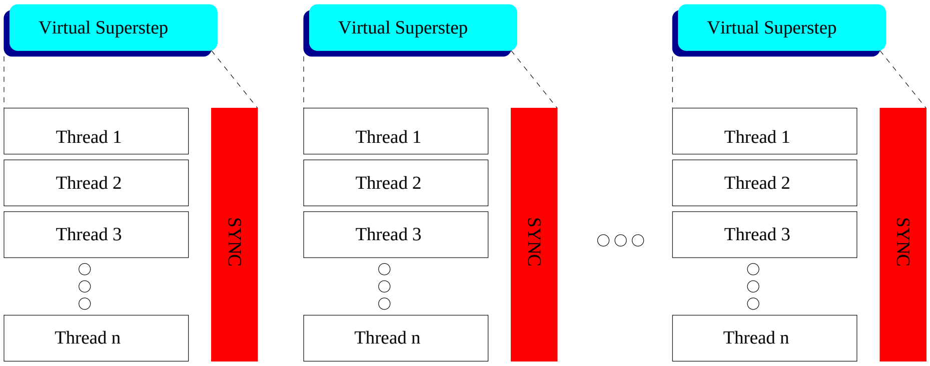

Figure 1.1 Representation of a BSP algorithm as a multithreaded decentralized computation.

This is due to a fixed communication initialization cost, which is irrelevant to the size of the $h-$relation to be implemented by the network infrastructure.

Finally, the following fact will be used in our time estimations in the sequel (for a justification of this fact, the reader is referred to [36, 6]):

Fact 3 *There exists a* BSP *algorithm that broadcasts a $k-$word message to N processors, that requires time at most $\mathcal{O}\left(\log N \cdot \max\{L, \frac{gk}{\log N}\}\right)$. Moreover, if $L \leq \frac{g \cdot k}{\log N}$ then the algorithm needs time $\mathcal{O}(g \cdot k)$.*

4.2.2 The Dynamic Simulation Model. Suppose that we are given a BSP algorithm $\mathcal{A}$, which is designed to run over a hypothetical, fault free $n-$processor BSP machine (the Virtual Machine, $\mathcal{VM}$). Because of its BSP nature, $\mathcal{A}$ assigns atomic work loads and imposes some demands for communication (e.g. the implementation of an $h-$relation) among the virtual processing elements, in each Virtual Superstep on $\mathcal{VM}$. Let's call the atomic work load of a specific virtual processor $Q_{\rm i}$, along with the portion of communication on behalf of $Q_{\rm i}$ for the current Virtual Superstep, thread $T_{\rm i}$. In that case, the execution of each Virtual Superstep (consisting of an LC-phase and a Comm-phase) may be considered as a balanced, multithreaded computation, while the execution of $\mathcal{A}$ may be considered as a sequence of multithreaded computations, distinguished by periodic barrier synchronization (SYNC) operations among the processing elements (see figure 1.1).

Suppose also that we are given an $n-$processor BSP machine which is prone to processor failures (the Real Machine, $\mathcal{RM}$)[3]. Then the task of $\mathcal{RM}$ will be to assure the progress of the simulation process until the end of the execution of the input algorithm $\mathcal{A}$, but also keep the total work evenly balanced among the remaining live processors and assure recovery from any sequence of faults at any time.

Let the (uniquely) assigned work load of a real processing element P_{i} in $\mathcal{RM}$ comprise its PRIMARYJOB. This means that the major task of P_{i} in each virtual superstep will be to simulate the execution of the threads which correspond to the virtual processors residing in its *PrimaryJobQueue*. Any additional work load that may be taken over by P_{i} (at its own initiative) during the simulation process, will comprise a SECONDARYJOB. This load is not necessarily uniquely assigned to the specific processor.

Initially we assume that there is a "1-1" correspondence between the processing elements of $\mathcal{RM}$ and $\mathcal{VM}$. This means that any thread of virtual processor Q_{i} is initially considered to be executed by the real processor P_{i} (which is its physical destination), unless P_{i} dies, in which case Q_{i} will have to migrate to another real processor that will have to execute its forthcoming threads. The necessary information for executing a specific thread T_{m}, is stored in a LOCALSTORAGE scheme comprised by the real processors. This information is stored using as a key the Virtual Identification number ($\mathcal{V}id$) of the corresponding virtual processor. This storage scheme is based on a BSP implementation of the well known Information Dispersal Algorithm (IDA, [66]) and assures the robustness against small bursts of processor failures.

Additionally, a GLOBALSTORAGE scheme has been adopted, for the periodic creation of some robust replicas of specific instances of $\mathcal{RM}$'s status during the simulation process (we call these securely stored instances *SafeStates*). The *SafeStates* play the role of reference points to which our simulation process will backtrack, in case of locally unrecoverable situations.

We also assume that processors may share a seed from a strong pseudorandom number generator (i.e., we have "coordinated randomness" among the operational nodes of $\mathcal{RM}$).

Definition 1 *A thread is called assigned if it is included in a live real processor's* PrimaryJobQueue*, otherwise it is called pending. Additionally, if a real processor has already completed the simulation of a specific thread (i.e., it has simulated its LC-phase and its Comm-phase and has safely stored the new status of the corresponding virtual processor), then this thread is considered to be completed for the current Virtual Superstep. Finally, a thread that corresponds to a virtual processor* Q_{i} *which is assigned to a real processor* P_{j} *with* $i \neq j$ *(i.e.,* P_{j} *is not the physical destination of* Q_{i}*), is called migrated.*

For the reader's easier understanding of the measured quantities, we adopt the following terminology:

A_{k}: the number of currently live processors at the beginning of virtual superstep VS_{k} (estimation).

$\mathcal{C}_{\mathrm{i}}(\mathcal{P}_{\mathrm{i}}/\mathcal{A}_{\mathrm{i}})$: the number of completed (pending/assigned) threads at the end of the SECONDARYJOB J_{i} of the current Virtual Superstep.

h: the size of the current virtual communication phase (i.e., the size of the $h-$relation among the n threads).

F_{k}: the number of faults that occur during Virtual Superstep VS_{k}.

$[n]$: the set $\{1, \ldots, n\}$.

For the representation of the *CurrentStatus* of the Virtual BSP Machine ($\mathcal{VM}$), we use the following data structures, which are robustly stored among the live processors of $\mathcal{RM}$, using a simple combination of a tentative storage scheme (the LOCALSTORAGE) and a definite one (the GLOBALSTORAGE), (the reader is referred to the full version of this work, for the description of the mixed storage scheme adopted):

VLM_{i}: the contents of the local memory used by the virtual processing element Q_{i}. The size of the VLMs is an input parameter for our simulation strategy. A reasonable assumption might be to consider a polylogarithmic size (e.g. $\mathcal{O}(\log^2 n)$) for these local memories.

LoU_{i}: a list of still undelivered outgoing messages, on behalf of the threads hosted by the live processor P_{i}.

PJ_{i}: the *PrimaryJobQueue* of P_{i}, containing the (uniquely assigned) threads that it executes during each PRIMARYJOB.

$\mathtt{N}_{\mathrm{u}}$: the size of a local neighborhood of real processors after u BACKTRACK operations have occurred, with $\mathtt{N}_0 = \log n$ and $\mathtt{N}_{\mathrm{u}} = 2 \cdot \mathtt{N}_{\mathrm{u}-1} = 2^{\mathrm{u}} \cdot \log n$.

4.2.3 Assumptions. Regarding the model of faults, we consider that an oblivious adversary determines a sequence of processor failures that are randomly distributed all over the whole simulation of a specific input algorithm $\mathcal{A}$, and these faults are unrecoverable (i.e., we adopt the non-restartable fail-stop model, where the dynamic faults are distributed uniformly during the whole simulation process).

This fault model tries to capture the behavior of a decentralized setting, such as a NETWORK OF WORKSTATIONS, or an arbitrary distributed computing environment, that behaves like a dynamically changing parallel machine through a virtual parallel interface, (e.g. MPI or PVM). This dynamic computing environment initially allocates processors to the simulation process according to the demands of the input algorithm, and then reclaims resources (according to the demands of the whole parallel setting) because of processor unavailability or temporary machine stalls.

It is also assumed that the amount of processor failures during the simulation process is upper-bounded by a fraction a, which is an input parameter to our simulation strategy. Clearly, this bound immediately affects the cost for creating *SafeStates* during the simulation process, to which the strategy will backtrack, when some unexpected behavior of the parallel setting occurs, without jeopardizing a total corruption of the whole simulation process.

From the above two restrictions (i.e. the random distribution of the faults during the simulation process and the existence of an overall upper-bound on the fraction of faults), it becomes apparent that if we divide the simulation time into sufficiently large (e.g. $\mathrm{polylog(n)}$) time intervals, there is an upper bound $r = r(a)$ on the fraction of faults that may occur in each of these intervals[4] and it is easily shown that the concentration around this bound is very sharp (this is easily understood by a simple application of the well known `Coupon Collector's Problem`). This implies that if a virtual superstep VS_{k} needs $\mathrm{polylog(n)}$ time to be executed, then $F_{\mathrm{k}} \leq rn$.

For the communication part, we suppose that the BSP algorithm to be executed imposes regular $h-$relation implementations (for some h, which is a parameter of the input algorithm), which means that the corresponding communication graph among the n threads is an $h-$regular digraph. In case that this is not true, some dummy messages could be used so as to have a regular communication digraph.

4.2.4 The Robust-$\mathcal{BSP}$ simulation strategy. Our simulation strategy for the dynamic case considers a BSP algorithm $\mathcal{A}$ of polynomial number of virtual supersteps, and divides it into epochs of K_0 virtual supersteps[5]. It then tries to robustly simulate the execution of the next epoch of $\mathcal{A}$ and create a new *SafeState* at the end of it, so as to have the work done up to this point committed.

In the sequel we deal with the simulation of a single epoch of virtual supersteps. The presentation begins with the simulation of a single vritual superstep, and then we describe the major techniques that we use for the mixed storage scheme, the dynamic load balancing of the work load and a tentative agreement protocol among the live processors of $\mathcal{RM}$. All these techniques are used by the Robust-$\mathcal{BSP}$ simulation strategy.

Simulation of a single virtual superstep

The purpose of the Robust-$\mathcal{BSP}$ strategy (see figure 1.2) is to initially let the live processors of $\mathcal{RM}$ try to execute the current virtual superstep VS_{k} as if there were no more processor failures and the work load were evenly distributed among the live processors of $\mathcal{RM}$ (this is done during the routine PRIMARY JOB). In case that VS_{k} is not yet completed, a number of SECONDARYJOBS[6] follows, so that the work of VS_{k} be finished and the pending work load be evenly redistributed among the remaining live processors of $\mathcal{RM}$.

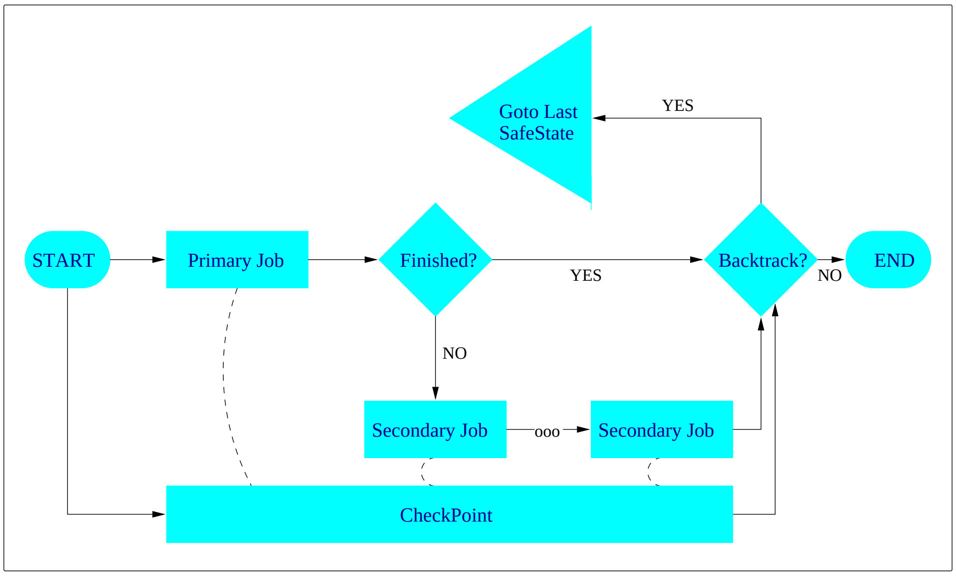

Figure 1.2 The simulation of a single virtual superstep.

As it will be demonstrated in the description of the PRIMARY JOB routine, the *PrimaryJobQueues* of the remaining live processors at the end of the previous Virtual Superstep ($VS_{\mathrm{k}-1}$) comprise a partition of the set of virtual processors into disjoint and almost evenly balanced sets of threads to be executed. Thus, provided that no more processor failures occur during the PRIMARY JOB of VS_{k} and the work load is already evenly balanced among the live processors of $\mathcal{RM}$, this part would be sufficient for the completion of the work of VS_{k}. Notice that no work replication is imposed at this part of our simulation strategy.

To assure the robustness of the simulation process against processor failures, a mixed storage scheme has been adopted (see figure 1.3). This storage scheme consists of two basic schemes: a tentative LOCAL STORAGE scheme[7], according to which the *CurrentStatus* of $\mathcal{VM}$ is stored during each superstep in some properly constructed neighborhoods of real processors[8], and a definite GLOBAL STORAGE scheme for creating a *SafeState* at the end of each epoch: that is a robustly stored replica of a specific instance of $\mathcal{VM}$'s *CurrentStatus* so as to be able to reconstruct it at any time, in case that a neighborhood of processors of the LOCAL STORAGE scheme has corrupted. The LOCAL STORAGE procedure is based on an implementation of the well known Information Dispersal Algorithm (IDA, [66]), while the GLOBAL STORAGE procedure creates the necessary replicas of LOCALSTORAGES so that up to arbitrary $a \cdot n$ processor failures can be overcome. The following lemma is taken from the work of

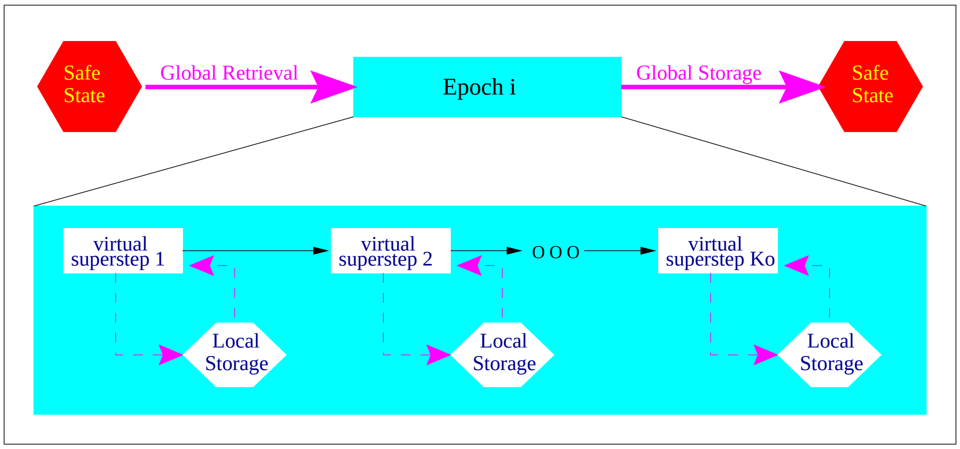

Figure 1.3 The MIXED STORAGE scheme

Kontogiannis et al. and gives the time costs for this MIXED STORAGE scheme:

Lemma 20 *Supposing that each live processor is responsible for at most λ threads, the time costs for the procedures of our storage scheme are:*

$$T_{\text{LocalStorage}} = 2\lambda|VLM|(\mathtt{N} + \max\{L, g\}) \tag{1.16}$$

$$T_{\text{GlobalStorage}} = \mathcal{O}(\lambda an|VLM|) \tag{1.17}$$

$$T_{\text{LocalRetrieval}} = \lambda(\mathtt{N_u}|VLM| + 2|VLM| \cdot \max\{L, g\}) \tag{1.18}$$

$$T_{\text{GlobalRetrieval}} = \mathcal{O}\left(\lambda an|VLM|\right) \tag{1.19}$$

Remark: A possible new processor failure (death) that will not let the corruption information be disseminated to all the live processors, or a failure of the GLOBAL STORAGE may cause no problem at all. This is because the problematic situation will be discovered during the next superstep, and a new BACKTRACK procedure will fix this abnormality. Additionally, each time a new BACKTRACK occurs, the probability of having a new corruption decreases significantly because of the redistribution of faults in the new, double-sized neighborhoods of processors, and thus there is no chance of an infinite loop in the same epoch.

In our simulation strategy an ADAPTIVE LOAD BALANCING scheme (ALB in short) is inherent. It tries to evenly distribute the work load of $\mathcal{VM}$ among the currently live processors of $\mathcal{RM}$. The purpose of ALB is to make the PRIMARY JOB routine work as close to optimal as possible, in case that the parallel setting is at a stable state (i.e. when no more processors die for a while).

This is a consequence of the “almost even work loads among the operational processors" property, achieved by our balancing scheme. This is actually an online load balancing technique based on ideas such as work stealing ([8]) and dynamic contest among the live processors for the assignment of new work. In section 4.2.5, ALB is shown to have a very good performance w.r.t. our simulation model.

The “FINISHED?” condition of the Robust-$\mathcal{BSP}$ strategy can be implemented by the network infrastructure that will have to add the number of completed threads in the *PrimaryJobQueues* of the live processors, and will signify the end of VS_{k} if n is reached by this sum. This added intelligence of the network infrastructure is necessitated by the fact that apart from the periodic synchronization of the real processors (that is provided by the underlying BSP machine, $\mathcal{RM}$) a periodic synchronization operation among the virtual processors is also necessary, that will signify the end of the virtual supersteps in the ideal machine, ($\mathcal{VM}$). Another approach could be the use of a tentative estimation for this boolean condition, by having the processors apply a BSP AGREEMENT protocol to decide whether the simulation of VS_{k} has been completed or not. Of course, a solution like this would require an assured Load Balancing property (i.e. very tight upper and lower bounds on the *PrimaryJobQueues*) so as for the failure probability to be extremely small. On the other hand, some checkpointing procedures such as those proposed in [32, 22] could be adapted to provide this virtual SYNC operation that would assure the integrity of $\mathcal{VM}$. Recall that a possible failure at this point would not be catastrophic, since it would be discovered anyway during the next virtual superstep.

As for the BACKTRACK? condition at the end of the currecnt virtual superstep, if a new corruption of the LOCAL STORAGE has been discovered by a processor, the dissemination of this exceptional information will occur the next time a BSP AGREEMENT protocol will be executed by the remaining live processors of $\mathcal{RM}$. This is done at the beginning of the next job (either a PRIMARY JOB or a SECONDARY JOB). At the end of the current superstep, a BACKTRACK routine is performed by all the live processors to the last *SafeState*, in case that a locally unrecoverable error has occurred, otherwise the simulation process proceeds with a new virtual superstep, or with the creation of a new *SafeState* and the work of the current epoch is then completed.

Primary Job

Suppose that the current virtual superstep that has to be executed is VS_{k}. As mentioned above, the task of PRIMARY JOB is to let all the live processors behave as if everything were fine and no new processor failures will occur during VS_{k}. In this phase, each live processor P_{i} executes its own (unique) chunk of threads, held in PJ_{i}. Before starting the execution of the threads held in its *PrimaryJobQueue*, each live processor P_{i} upper-bounds the work load that it will execute, according to an estimation A_{k} of the currently live processors,

[1]	BSPAGREEMENT: Find an estimation of the currently live processors, A_{k}.
[2]	**IF** $\vert PJ_{\mathrm{i}}\vert > \frac{c\cdot n}{A_{\mathrm{k}}}$ **THEN** Discard some randomly chosen of the assigned threads, until $\vert PJ_{\mathrm{i}}\vert \leq \frac{c_1\cdot n}{A_{\mathrm{k}}}$.
[3]	**LC-phase:** Execute all the LC-phases of the threads in PJ_{i}.
[4]	**Comm-phase:** Send all the outgoing messages to their physical destinations.
[5]	Register all the undelivered messages to the List of Undelivered messages, LoU_{i}.
[6]	SDOU: Try to send as many as possible of the undelivered messages, to their destinations.
[7]	LOCALSTORAGE: Store the VLMs of all the completed threads into the proper neighborhoods.

Figure 1.4 The Primary Job procedure.

so that no processor be unevenly overloaded. This estimation is based on an application of the tentative BSP AGREEMENT protocol.

Consequently, each live processor executes its own portion of work, sends the outgoing messages to their physical destinations, and tries (employing the SDOU routine, see section 4.2.6) to forward the undelivered messages which are heading for migrated threads. Finally, it stores the VLM of each completed thread in the proper neighborhood of the LOCAL STORAGE scheme, according to the corresponding virtual processor's (unique) $\mathcal{V}id$.

According to the analysis of this routine in [53], the overall cost of each PRIMARY JOB is upper-bounded by

$$T_{\mathrm{PrimJob}} = \mathcal{O}\left(\mu_{\mathrm{k}} \cdot \max\{h \cdot \log n, \log^4 n, T_{\mathrm{LC}} + T_{\mathrm{Comm}}\}\right), \qquad (1.20)$$

where $\mu_{\mathrm{k}} \equiv \frac{c_1 n}{A_{\mathrm{k}}}$ indicates the maximum load of each *PrimaryJobQueue* after the discarding of the excessive loads from the overloaded processors.

Secondary Job

The goal of each SECONDARY JOB J_{i} is to complete the outstanding work of VS_{k} and redistribute it among the live processors as evenly as possible. The new work assignment is done as follows: Each of the A_{k} live processing elements chooses $\mathbf{b} \equiv g(n) \cdot \frac{n}{A_{\mathrm{k}}}$ threads to cover (that is, to execute, if not yet completed) either at random, or derived by previous unsuccessful communication attempts with pending threads. For the purposes of the following analysis we consider all these choices independent and random, although it is our strong belief that the “biased" choices would make the processing elements focus exactly on the remaining unsatisfied threads.

Lemma 21 *The number of individual processors that cover a specific (pending) thread during* $J_{\rm i}$, *is* $c_3 \cdot \log n$, *with probability* $1 - n^{-c_4}$, $\forall c_3 > 0$, *and* c_4 *depending on* c_3.

Proof: The total number of random choices for covering the n threads, is $M \equiv n \cdot g(n)$. If we have $g(n) = \ln n + c_2$ then it is assured by the well known `Coupon Collector's Problem (CCP)` that with overwhelming probability ($e^{-e^{-c_2}}$, see section 3.6 of [58]) all the n threads will be covered by a live processor at least once. Here we want to prove a less strict property, i.e. that all the threads will be covered by at least $\mathcal{O}(\log n)$ processors with high probability (which implies that the failure probability will be polynomially small).

Let $X_{\rm i,j}$ be the indicator variable of the $i^{\rm th}$ trial hitting thread $T_{\rm j}$. Then $Y_{\rm j} = \sum_{\rm i=1}^{\rm M} X_{\rm i,j}$ is the random variable indicating the total number of "hits" for the thread $T_{\rm j}$. Obviously, $E[Y_{\rm j}] = g(n)$, and by applying Chernoff Bounds on these random Bernoulli trials, it is easily shown that

$$\forall j \in [n], \Pr[Y_{\rm j} < c_3 \cdot \log n] < \exp\left(-\frac{g(n)}{2} - c_3 \cdot \log n + \frac{c_3^2 \cdot \log n}{2g(n)}\right)$$

We want to show that

$$\Pr[\exists j \in [n], Y_{\rm j} < c_3 \cdot \log n] = \mathcal{O}(n^{-c_4}), \tag{1.21}$$

for some positive constant c_4 that depends on the number of independent trials.

$$\Pr[\exists j \in [n], Y_{\rm j} < c_3 \cdot \log n] \quad \leq \quad \sum_{\rm j \in [n]} \Pr[Y_{\rm j} < c_3 \cdot \log n] \leq n \cdot e^{-K},$$

where $K \equiv \frac{g(n)}{2} + c_3 \cdot \log n - \frac{c_3^2 \cdot \log n}{2g(n)}$. We now have that

$$\begin{aligned} n \cdot e^{-K} &= n^{-c_4} \Rightarrow c_4 = \frac{K}{\ln n} - 1 \Rightarrow \\ c_4 &= \frac{g(n)}{2 \ln n} + \frac{c_3 \cdot \log n}{\ln n} - \frac{c_3^2 \cdot \log n}{2g(n) \cdot \ln n} - 1 \end{aligned}$$

Clearly, for $g(n) = \Theta(\log n)$ we have shown that the joint probability of each thread being "hit" by less than $c_3 \cdot \log n$ trials is polynomially small. But we would like to know the number of distinct processing elements that cover a specific thread. As it is shown by the `Sparse Occupancy Problem` (see [53]), each processor hits a specific thread at most once when the number of trials $g(n)$ is significantly smaller than n, with probability that tends to unity. So we can guarantee that each thread is covered by at least $c_3 \cdot \log n$ processing elements with probability $1 - n^{-c_4}$. □

[1] BSPAGREEMENT: Check for new LOCALSTORAGE corruptions.
[2] Each live processor P_{i} creates a bucket $\mathcal{B}_{\mathrm{i}}$ of $\mathbf{b}$ randomly chosen threads.
[3] INFORMATIONGATHERING: for all the threads in $\mathcal{B}_{\mathrm{i}}$, get the corresponding VLMs from the appropriate neighborhoods.
[4] **LC-phase:** Execute the $LC - phases$ of the threads in $\mathcal{B}_{\mathrm{i}}$.
[5] **Comm-phase:** Send all the outgoing messages to their physical destinations.
[6] Register all the undelivered messages to the List of Undelivered messages, LoU_{i}.
[7] SDOU: Try to send as many of the undelivered messages as possible, to their destinations.
[8] LOCALSTORAGE: Store the VLMs of all the completed threads into the proper neighborhoods.

Figure 1.5 The Secondary Job procedure.

Conditioning now on the number of distinct processors that cover a random thread during a SECONDARY JOB and then applying the Markov Inequality, we can show that:

Theorem 1 *The probability of having at least c_5 pending threads at the end of a* SECONDARY JOB *is given by*

$$\Pr[\mathcal{P}_{\mathrm{i}} \geq c_5] \leq \frac{n^{-\min[c_3 \cdot \log(\frac{1-a}{r}), c_4]+1}}{c_5} \tag{1.22}$$

The time analysis of each SECONDARY JOB is quite similar to the analysis of a PRIMARY JOB and thus its cost is given by

$$T_{\mathrm{SecJob}} = \mathcal{O}\left(\mathbf{b} \cdot \max\{h \cdot \log n, \log^4 n, T_{\mathrm{LC}} + T_{\mathrm{Comm}}\}\right) \tag{1.23}$$

As for the number of SECONDARY JOBS that are necessary for the whole simulation process to work, this is determined by the following statement of [53]:

Lemma 22 *The failure probability of a Virtual Superstep VS_{k} to complete its work is $n^{-\mathcal{O}(\mathrm{loglog}\, n)}$, considering that $y = \mathrm{loglog}\, n$* SECONDARYJOBS *are executed, if necessary.*

Having a subpolynomially small failure probability for each of the virtual supersteps that comprise an epoch, it is now apparent that an epoch can contain

$K_0 = \Theta(n)$ virtual supersteps, and still have subpolynomially small failure probability.

Checkpointing and Backtracking

The CHECKPOINT procedure is actually a virtual process that is done by any processor during the simulation of the input algorithm. More specifically, CHECKPOINT signifies the failure of some LOCAL STORAGE or INFORMATIONGATHERING procedure call, discovered by a live processor, either because of an update failure of the LOCAL STORAGE routine, or because of an unsuccessful attempt to retrieve a specific *VLM*. The discovery of a problematic situation is disseminated to the rest of the live processors, the next time that a BSP AGREEMENT protocol is executed, as an exception code to the specific live processor's value. Notice that a new processor failure may not cause any trouble at all, because the problematic situation will be rediscovered by $\mathcal{O}(\log n)$ live processors that will try to cover the corresponding pending (because of the new death) thread during the next SECONDARY JOB.

When such an interruption to the flow of the simulation process is done, the BACKTRACK operation simply makes the live processors set their program counters to the last *SafeState*, set a new value $\mathtt{N}_\mathrm{u} = 2 \cdot \mathtt{N}_{\mathrm{u}-1}$ (u is the number of BACKTRACKS having occurred up to now), consider a new, random hash function for the size-$\mathtt{N}_\mathrm{u}$ equi-partition of the real processing elements into $\frac{n}{\mathtt{N}_\mathrm{u}}$ neighborhoods of real processors (see the description of the LOCAL STORAGE scheme in the full paper), and continue with the simulation process after having retrieved the *CurrentStatus* from the last *SafeState*.

The choice of a new (pseudo-) random hash function for the construction of the new neighborhoods of processors is done in order to redistribute the up to this point occurred faults evenly among the new neighborhoods, and protect the simulation from a malicious behavior of an adversary that would try to focus its power on a specific neighborhood. On the other hand, the size of each neighborhood is doubled in order to bound the total probability of having too many corruption, as indicated by the following technical lemma[9]:

Lemma 23 *The probability of having a new neighborhood corruption after* $\log\log n$ BACKTRACK *operations, is*

$$\mathcal{O}(n^{-\log n} \cdot \log^{-3} n).$$

Proof: Consider that we have n processors and we choose a random equi-partition of them into $n/\mathtt{N}_\mathrm{u}$ neighborhoods of size $\mathtt{N}_\mathrm{u}$ (u indicates the number of BACKTRACK occurrences, up to now). Suppose also that *D* processors have already died (and have caused the u BACKTRACKS) and $A = n - D$ remain live. We say that an equi-partition corrupts, if there exists a neighborhood that has at least $\frac{\mathtt{N}_\mathrm{u}}{2} + 1$ dead processors. The following claim will shed some light to our discussion:

Claim 2 *The probability $\Phi(D, \mathtt{N})$ that a randomly chosen size$-\mathtt{N}$ equi-partition corrupts because of D processor failures, is given by*

$$\Phi(D, \mathtt{N}) = \begin{cases} 0, \textit{if } D < \mathtt{N}/2 + 1, \\ \frac{n \cdot 2^{\mathtt{N}+\frac{1}{2}}}{\sqrt{\pi \mathtt{N}} \cdot (\mathtt{N}+2)} \cdot \exp[-(\mathtt{N}/2+1)(H_{\mathrm{n}} - H_{\mathrm{D}})], \; \textit{otherwise,} \end{cases}$$

where $H_{\mathrm{n}},\ H_{\mathrm{D}}$ are the corresponding harmonic numbers.

Proof: For the proof of this claim the reader is referred to [53]. □ The corruption probability of a size-$\mathtt{N}_{\mathrm{u}}$ equi-partition ($\mathtt{N}_{\mathrm{u}} = 2^{\mathrm{u}} \cdot \log n$) is given by $\Phi(D, N_{\mathrm{u}})$, which is always at most equal to $\Phi(an, N_{\mathrm{u}})$. Some calculation will help to see that this is a very good bound for the failure probability, which actually implies that at most $\mathrm{loglog}\, n$ backtrack operations may occur during the simulation of the input algorithm, with very high probability (wvhp). More specifically, considering that $H_{\mathrm{n}} - H_{\mathrm{a \cdot n}} = \ln(1/a) + \Theta(1) = \ln(1/a) + \omega_1$, we have

$$\Phi(D, \mathtt{N}_{\mathrm{u}}) \quad \leq \quad \Phi(an, \mathtt{N}_{\mathrm{u}}) \leq \frac{2^{\omega_2}}{\sqrt{\pi \mathtt{N}_{\mathrm{u}}} \cdot (\mathtt{N}_{\mathrm{u}} + 2)} \tag{1.24}$$

where,

$$\begin{aligned} \omega_2 \quad \equiv \quad & \mathtt{N}_{\mathrm{u}} + 1/2 + \log n - \mathtt{N}_{\mathrm{u}} \cdot \left(\frac{\log(1/a)}{2} + \frac{\omega_1}{2 \ln 2} \right) - \\ & - \left(\log(1/a) + \frac{\omega_1}{\ln 2} \right) = \mathcal{O}\left(-2^{\mathrm{u}} \cdot \log n \right) \end{aligned}$$

□

The Performance of the Robust-$\mathcal{BSP}$ simulation strategy

In this subsection we estimate the amortized cost of a virtual superstep executed by the Robust-$\mathcal{BSP}$ simulation strategy, and give a bound on the competitive ratio of our strategy, against an optimal offline strategy, that always lets the live processors execute a fully balanced work load[10].

Theorem 2 *The amortized cost for the simulation of a single virtual superstep is given by*

$$T_{\mathrm{VS_k}} = \mathcal{O}\left((\log n \cdot \mathrm{loglog}\, n)^2 \cdot T_{\mathrm{OPT}} + \mathrm{polylog(n)} \right) \tag{1.25}$$

with probability at least $\mathcal{O}(1 - n^{-\log n} \cdot \log^{-3} n)$.

Proof: According to the up to now analysis of the Robust-$\mathcal{BSP}$ simulation strategy, the parallel time of a single epoch of K_0 virtual supersteps is upper-bounded by the following equation:

$$\begin{aligned} T_{\mathrm{epoch}} \quad = \quad & (\mathcal{B}+1) K_0 \left(T_{\mathrm{PrimJob}} + \mathrm{loglog}\, n \cdot T_{\mathrm{SecJob}} \right) + \\ & + \mathcal{B} \cdot (\mu_{\mathrm{k}} + \mathbf{b} \cdot \mathrm{loglog}\, n) \cdot \\ & \cdot (T_{\mathrm{GlobalStorage}} + T_{\mathrm{GlobalRetrieval}}) \end{aligned} \tag{1.26}$$

where at most $\mathcal{B}$ BACKTRACKS occur during this epoch. But we know that with subpolynomially small failure probability, the number of BACKTRACK operations in the whole simulation process is no more than $\operatorname{loglog} n$. We also know that in each superstep, a live processor may request for at most $\mu_{\mathrm{k}} + b \cdot \operatorname{loglog} n$ threads.

Notice also that if we suppose that $T_{\mathrm{LC}} + T_{\mathrm{Comm}} = \mathcal{O}(g \cdot h) \geq \log^4 n)$, then $T_{\mathrm{PrimJob}} = \mu_{\mathrm{k}} \cdot \max\{h \cdot \log n, (T_{\mathrm{LC}} + T_{\mathrm{Comm}})\} = \mathcal{O}(\frac{\log n}{g} \cdot T_{\mathrm{OPT}})$ and $T_{\mathrm{SecJob}} = \mathbf{b} \cdot \max\{h \cdot \log n, (T_{\mathrm{LC}} + T_{\mathrm{Comm}})\} = \mathcal{O}(\frac{\log^2 n}{g} \cdot T_{\mathrm{OPT}})$. In that case, the amortized cost of a single virtual superstep is given by

$$
\begin{aligned}
T_{\mathrm{VS_k}} &= (\mathcal{B}+1) \cdot T_{\mathrm{PrimJob}} + \frac{an \cdot \operatorname{loglog} n}{K_0} \cdot T_{\mathrm{SecJob}} + \\
&\quad + \frac{\mathcal{B} \cdot (\mu_{\mathrm{k}} + \mathbf{b} \cdot \operatorname{loglog} n)}{K_0} \cdot (T_{\mathrm{GlobalStorage}} + T_{\mathrm{GlobalRetrieval}}) \\
&= \mathcal{O}((\log n \cdot \operatorname{loglog} n)^2) \cdot T_{\mathrm{OPT}} + \mathrm{polylog(n)}) \qquad (1.27)
\end{aligned}
$$

□

Remark: Since the probability of having more than $\operatorname{loglog} n$ BACKTRACK occurrences is $\mathcal{O}(n^{-\log n} \cdot \log^{-3} n)$, the expected (amortized) cost of each virtual superstep converges to the above value, since the subpolynomially small failure probability dominates over the cost of some extra BACKTRACK occurrences.

In the following sections we present some fundamental randomized techniques that were exploited by the Robust-$\mathcal{BSP}$ strategy. More specifically, we present the ADAPTIVE LOAD BALANCING scheme (section 4.2.5), the SDOU procedure that implements the indirect communication among the migrated threads (section 4.2.6), and the BSP AGREEMENT protocol exploited by the live processors to achieve a common estimation on the number of live processors at the beginning of a new virtual superstep (section 4.2.7).

4.2.5 The Adaptive Load Balancing strategy. A major result of this work which is also of independent interest, is the proposed algorithm for balancing the work of the n virtual processors among the currently live processors of $\mathcal{RM}$. In fact, this is a randomized, adaptive load balancing technique, since the sequence of the fault occurrences is not known a priori to the simulation process and the live processors contest for the pending threads and get some of them according to a probability depending on their current loads.

Starting from a balanced situation (at the last *SafeState*), we show in this subsection how we keep the work of the live processors balanced (in a tentative fashion) for a whole epoch of K_0 virtual supersteps[11]. Let $\mathcal{Z}_0(i,k)$ denote the size of P_{i}'s queue after discarding the excessive work (in case of overloaded real processors), and $\mathcal{Z}(i,k)$ denote its size at the end of VS_{k}. Our ADAPTIVE

LOAD BALANCING scheme (ALB in short), which is inherent in the Robust-$\mathcal{BSP}$ simulation strategy, is the following:

[1] At the beginning of each virtual superstep VS_{k}, each overloaded real processor P_{i} cuts off the excessive work (i.e. keeps at most $\frac{c_1 n}{A_{\mathrm{k}}}$ threads at random in PJ_{i}, according to the estimation A_{k} of the number of live processors in $\mathcal{RM}$, and makes the remaining threads pending). This is the Discarding step at the beginning of each PRIMARY JOB.

[2] All the pending threads of $\mathcal{VM}$ (due to either new deaths, or discardings) are rescheduled to still live processors of $\mathcal{RM}$ as follows:

(a) Live processors contest for $\mathbf{b} \cdot \operatorname{loglog} n$ randomly chosen threads to cover, during the $\operatorname{loglog} n$ SECONDARYJOBS.

(β) Each mailbox that takes over a thread T_λ during a round of SDoU, assigns it (if it is pending) to one of the live processors that contests for it, with probability

$$\Pr[P_{\lambda_{\mathrm{i}}} \text{ gets } T_\lambda] = \frac{\frac{1}{\mathcal{Z}_0(\lambda_{\mathrm{i}},k)}}{\sum_{\mathrm{j}\in \mathrm{C}} \frac{1}{\mathcal{Z}_0(j,k)}}$$

where $C = \{P_{\lambda_1}, P_{\lambda_2}, \ldots\}$ is the set of processors contesting for T_λ.

Theorem 3 *Let $\xi > 1$ be a constant. Then the load of each live processor P_{i} in $\mathcal{RM}$ at the end of VS_{k} is $\mathcal{Z}(i,k) \le \frac{c_1 n}{A_{\mathrm{k}}} \cdot \Theta(\operatorname{loglog} n)$, with probability at least*

$$\left(1 - n^{-\xi}\right)^{(\mathrm{k}+1)} \ge 1 - (k+1)n^{-\xi}$$

Proof: Let $\xi > 0$ be a constant. We shall prove the good behavior of ALB using induction on the epochs of the input algorithm $\mathcal{A}$.

Initial Step. At the beginning of our simulation process, each virtual processor is assigned to its **physical destination**, and thus, each live processor in $\mathcal{RM}$ has exactly one thread in its *PrimaryJobQueue*.

Inductive Hypothesis. Suppose that at the beginning of epoch e_{i}, we have

$$\forall j \in [n],\ \mathcal{Z}(i,k-1) \le \frac{c_1 n}{A_{\mathrm{k}-1}} \cdot \omega(n) \qquad (1.28)$$

where $\omega(n)$ is a constant times $\operatorname{loglog} n$.

Inductive Step. Recall that a *PrimaryJobQueue* PJ_{j} is overloaded iff

$$\frac{c_1 n}{A_{\mathrm{k}}} < \mathcal{Z}(i, k-1) \leq \frac{c_1 n}{A_{\mathrm{k}-1}} \cdot \omega(n) \tag{1.29}$$

The amount $\mathcal{P}$ of pending threads during VS_{k} will have to be rescheduled among the remaining operational processors of $\mathcal{RM}$. In $\mathcal{P}$ we measure only assignments of threads to processors that remain live until their completions, because a thread assigned to a processor that dies before achieving its completion has already been included in $\mathcal{P}$, either as a pending thread because of a new death, or as a discarded thread at the beginning of VS_{k}.

Let f_{k} denote the set of newly dead processors during VS_{k}, and D the number of discarded threads at the beginning of VS_{k}. Then $F_{\mathrm{k}} = |f_{\mathrm{k}}| \leq rn$. Clearly, $\mathcal{P} = D + \sum_{\mathrm{i} \in \mathrm{f_k}} \mathcal{Z}_0(i,k) \leq a \cdot n$. Consider now a specific thread T_λ and suppose that the operational processors $P_{\lambda_1}, P_{\lambda_2}, \ldots, P_{\lambda_{\mathrm{m}}}$ contest for it during a SECONDARYJOB. We already know that T_λ is covered by less than $c_3 \cdot \log n$ initially operational processors of $\mathcal{RM}$ with probability n^{-c_4}, for some positive constant c_3 and c_4 depending on c_3. So we can go on with our proof conditioning on the event "$m \geq \log n$". By the rescheduling rule we have:

$$\forall i \in [\log n],\ \Pr[P_{\lambda_{\mathrm{i}}} \text{ gets } T_\lambda] \leq \frac{\frac{1}{\mathcal{Z}_0(\lambda_{\mathrm{i}},k)}}{\sum_{\mathrm{j}=1}^{\log n} \frac{1}{\mathcal{Z}_0(\lambda_{\mathrm{j}},k)}} \tag{1.30}$$

It is now obvious that $\mathcal{Z}_0(\lambda_{\mathrm{j}}, k) \leq \frac{c_1 n}{A_{\mathrm{k}}}, \forall j \in [\log n]$. So, $\sum_{\mathrm{j}=1}^{\log n} \frac{1}{\mathcal{Z}_0(\lambda_{\mathrm{j}},k)} \geq \frac{A_{\mathrm{k}} \cdot \log n}{c_1 n}$ and thus we have:

$$\forall\ i \in [\log n],\ \Pr[P_{\lambda_{\mathrm{i}}} \text{ gets } T_\lambda] \leq \frac{c_1 n}{\mathcal{Z}_0(\lambda_{\mathrm{i}}, k) \cdot \log n \cdot A_{\mathrm{k}}} \tag{1.31}$$

Now, each live processor P_{i} will randomly (with replacement) choose $\mathbf{b} \cdot \operatorname{loglog} n = \frac{c_1 n}{A_{\mathrm{k}}} \cdot \log n \operatorname{loglog} n$ threads to cover during the SECONDARYJOBS of VS_{k}. Each random choice has a probability $\frac{\mathcal{P}}{n}$ of hitting a pending thread and is independent of the other choices. Thus the number of pending threads that P_{i} will contest for, is the number of successes from $\mathbf{b} \cdot \operatorname{loglog} n$ Bernoulli trials $B(q, \frac{\mathcal{P}}{n})$, where $q \equiv \frac{c_1 n}{A_{\mathrm{k}}} \cdot \log n \cdot \operatorname{loglog} n$. For each such thread, the probability of P_{i} prevailing over the other contestants (in a specific round of SDOU), has already been shown to be

$$\Pr[P_{\mathrm{i}} \text{ gets } T_\lambda] \leq \frac{c_1 n}{A_{\mathrm{k}} \cdot \log n} \cdot \frac{1}{\mathcal{Z}_0(\lambda_{\mathrm{i}}, k)} \equiv \varphi$$

Let Ξ_{i} be the number of pending threads that P_{i} contests for. Then

$$\forall \varepsilon \in (0,1), \Xi_{\mathrm{i}} \in \left[(1-\varepsilon) \cdot q \cdot \frac{\mathcal{P}}{n}, (1+\varepsilon) \cdot q \cdot \frac{\mathcal{P}}{n}\right]$$

with probability at least $1 - \exp\left(-\frac{\varepsilon^2}{2} \cdot q \cdot \frac{\mathcal{P}}{n}\right)$, by a simple application of Chernoff Bounds on the Bernoulli trials. But recall that $\mathcal{P} \leq a \cdot n$, and so

$$\forall \varepsilon \in (0,1), \Xi_{\mathrm{i}} \in [(1-\varepsilon) \cdot q \cdot a, (1+\varepsilon) \cdot q \cdot a]$$

with probability at least

$$1 - \exp\left(-\frac{\varepsilon^2}{2} \cdot \frac{ac_1}{1-a} \cdot \log n \cdot \mathrm{loglog}\, n\right) = 1 - n^{-\frac{\varepsilon^2 \cdot c_1 \cdot a \cdot \log e}{2(1-a)} \cdot \mathrm{loglog}\, n}$$

Now, given Ξ_{i}, due to our rescheduling rule, P_{i} will actually add at most $\Xi_{\mathrm{i}} \cdot \varphi$ new threads in its own *PrimaryJobQueue* with probability at least $1 - n^{-\mathrm{c}_9}$ as it is easily shown by a new application of Chernoff Bounds, considering the worst case for P_{i}, $\mathcal{Z}_0(i,k) = 1$, for which φ is maximized). So, with total probability at most $(1 - n^{-\mathrm{c}_9}) \cdot \left(1 - n^{-\frac{\varepsilon^2 \cdot c_1 \cdot a \cdot \log e}{2(1-a)} \cdot \mathrm{loglog}\, n}\right)$ each operational processor P_{i} gets at most $\Xi_{\mathrm{i}} \cdot \varphi$ threads, i.e.,

$$\Delta \mathcal{Z}(i,k) \leq (1+\varepsilon) \cdot a \cdot q \cdot \varphi \leq \frac{\beta}{\mathcal{Z}_0(i,k)}$$

with $\beta = (1+\varepsilon) a \left(\frac{c_1}{1-a}\right)^2 \cdot \mathrm{loglog}\, n$. Hence,

$$\mathcal{Z}(i,k) = \mathcal{Z}_0(i,k) + \Delta \mathcal{Z}(i,k) = \mathcal{Z}_0(i,k) + \frac{\beta}{\mathcal{Z}_0(i,k)}$$

which is maximized at $\max\{\mathcal{Z}(i,k)\} = \max\{1+\beta, 1+\gamma \cdot \mathrm{loglog}\, n\} \cdot \frac{c_1 n}{A_{\mathrm{k}}}$ and thus it satisfies the inductive hypothesis with probability of success at least

$$\left(1 - n^{-\xi}\right)^{\mathrm{k}} \cdot (1 - n^{-\mathrm{c}_9}) \cdot \left(1 - n^{-\frac{\varepsilon^2 \cdot c_1 \cdot a \cdot \log e}{2(1-a)} \cdot \mathrm{loglog}\, n}\right) \geq \left(1 - n^{-\xi}\right)^{\mathrm{k}+1}$$

with $\xi \leq \min\left\{c_9, \frac{\varepsilon^2 \cdot c_1 \cdot a \cdot \log e}{2(1-a)} \cdot \mathrm{loglog}\, n\right\}$, and $k \leq K_0$.

Remark: Notice that $\left(1 - n^{-\xi}\right)^{\mathrm{k}+1} \geq 1 - (k+1) \cdot n^{-\xi}$ which implies that our robust system can tolerate computations of polynomial length.

□

4.2.6 Secure Delivery of Undelivered Messages. In the SDoU procedure the primary objective is the safe delivery of messages heading for migrated threads, that have been kept in the LoUs of the live processors up to this point. Additionally, in the case of a SECONDARY JOB having preceded, this is also the rescheduling procedure, for the pending threads.

[1] **PARDO** ν times:
[2] **BSPRandomChoice:** Each live processor chooses a common random seed.
[3] Choose a new permutation $\Pi_{\rm j}$ according to the random seed and send all the messages to their corresponding mailboxes.
[4] **ThreadRequest:** Each processor claims the incoming messages for its own newly executed threads from the proper mailboxes (e.g. $T_{\rm m}$ from $P_{\Pi_{\rm j}(m)}$).
[5] **ThreadAssignment:** Each operational mailbox accepts ONLY one of the requests and sends the incoming messages of the thread that it serves to it.
[6] **ThreadCommittment:** Each live processor $P_{\rm i}$ that receives the mailbox of a thread $T_{\rm m}$, adds it to $PJ_{\rm i}$ and marks it as newly completed.
[7] **ENDPARDO**.

Figure 1.6 The Secure Delivery of messages to migrated threads.

The basic idea of this procedure (see figure 1.6) is the use of multiple mailboxes for each thread, until all threads have been served by at least one live mailbox. That is, using ν random permutations of $[n]$,$(\Pi_1, \Pi_2, \ldots, \Pi_\nu)$, we proceed in ν rounds of indirect communication attempts, where in round j, each live processor $P_{\rm i}$ tries to successfully send any message for $T_{\rm m}$, held in $LoU_{\rm i}$, to the mailbox $P_{\Pi_{\rm j}(m)}$. Consequently (in the same round), each mailbox sends the incoming messages of the corresponding thread to a (possibly one chosen among many) requester of $T_{\rm m}$, according to the following rescheduling rule: In the case of multiple contesting processors for a specific thread, the mailbox sends it to one of the contestants (and thus assigns the corresponding virtual processor to it), according to the contestants' current work load.

Lemma 24 *If the number ν of required rounds of implicit communication attempts is $\frac{c_6+1}{\log(1/a)} \cdot \log n$, then the failure probability is $\Pr[\textit{SDoU fails}] \leq n^{-c_6}$, $\forall c_6 > 0$.*

Proof: Let Π_0 be the identical permutation of $[n]$ and let $\Pi_1, \Pi_2, \ldots, \Pi_\nu$ be the required permutations so that each of the n threads be served by at least one live processor (to be its mailbox). For the following proof we shall need some definitions from the Theory of Negative Dependence of Random Variables (see [24, 23]):

Definition 2 *Let n be a positive integer.*

1. *The random variables* $J_1, \ldots, J_{\mathrm{n}}$ *are said to have the permutation distribution on* $[n]$*, if they take values in* $[n]$*, and for any permutation* $\sigma : [n] \to [n]$*,*

$$\Pr[J_1 = \sigma(1), \ldots, J_{\mathrm{n}} = \sigma(n)] = \frac{1}{n!}.$$

2. *Let* $x_1, \ldots, x_{\mathrm{n}}$ *arbitrary reals. The random variables* $X_1, \ldots, X_{\mathrm{n}}$ *are said to have a permutation distribution on* $(x_1, \ldots, x_{\mathrm{n}})$*, if there is a set of random variables* $J_1, \ldots, J_{\mathrm{n}}$ *with the permutation distribution on* $[n]$ *and* $X_{\mathrm{i}} = x_{\mathrm{J_i}}, \forall i \in [n]$.

Definition 3 *The random variables* $\mathbf{X} = (X_1, \ldots, X_{\mathrm{n}})$ *are negatively associated if for every index set*

$$I \subseteq [n], \mathrm{Cov}[f(X_{\mathrm{i}}, i \in I), g(X_{\mathrm{j}}, j \in [n] - I)] \leq 0,$$

for all non-decreasing functions $f : I\!R^{|\mathrm{I}|} \to I\!R$ *and* $g : I\!R^{|[\mathrm{n}]-\mathrm{I}|} \to I\!R$*.*

Consider now the following $\nu \cdot n$ indicator variables:

$$\forall i \in [\nu], j \in [n], X_{\mathrm{i,j}} = \begin{cases} 1, & \text{if } P_{\Pi_{\mathrm{i}}(j)} \text{ is live,} \\ 0, & \text{otherwise.} \end{cases}$$

For any fixed i, the vector $\mathbf{X}_{\mathrm{i}} = (X_{\mathrm{i},1}, X_{\mathrm{i},2}, \ldots, X_{\mathrm{i},n})$ have the permutation distribution on

$$Status = \{Status(1), Status(2), \ldots, Status(n)\}.$$

Thus, for any fixed $i \in [\nu]$, the vector $\mathbf{X}_{\mathrm{i}}$ follows the negative association property. Additionally, the vector $\mathbf{X}= (X_1, X_2, \ldots, X_\nu)$ consists of mutually independent components (since they correspond to randomly and independently chosen permutations), and thus $\mathbf{X}$ also has the negative association property.

Consider now the functions $Z_{\mathrm{j}} = \sum_{\mathrm{i}=1}^{\nu} X_{\mathrm{i,j}}$. Since these variables are actually non-decreasing functions on disjoint sets of negatively associated r.v.'s, the vector $\mathbf{Z} = (Z_1, \ldots, Z_{\mathrm{n}})$ follows also the negative association property. Now we can proceed with the failure probability of the SDoU routine to successfully serve all the assigned or pending threads:

$$\begin{aligned} \Pr[\text{SDoU fails}] &= \Pr[\exists j \in [n] : Z_{\mathrm{j}} = 0] \leq \sum_{\mathrm{j}=1}^{n} \Pr[Z_{\mathrm{j}} = 0] \\ \Pr[Z_{\mathrm{j}} = 0] &= \Pr[X_{\mathrm{i,j}} \leq 0, \forall i \in [\nu]] = \Pr[\bar{X}_{\mathrm{i,j}} \geq 1, \forall i \in [\nu]] \end{aligned}$$

Since $(X_{1,\mathrm{j}}, \ldots, X_{\nu,\mathrm{j}})$ are negatively associated, the same goes for the vector of complementary random variables $(\bar{X}_{1,\mathrm{j}}, \ldots, \bar{X}_{\nu,\mathrm{j}})$. Thus we have:

$$\left.\begin{aligned} \Pr[\bar{X}_{\mathrm{i,j}} \geq 1, \forall i \in [\nu]] &\leq \textstyle\prod_{i \in [\nu]} \Pr[\bar{X}_{\mathrm{i,j}} \geq 1] \\ \forall i \in [\nu], j \in [n], \Pr[\bar{X}_{\mathrm{i,j}} \geq 1] &= \Pr[X_{\mathrm{i,j}} \leq 0] \leq a \end{aligned}\right\} \Rightarrow \Pr[Z_{\mathrm{j}} = 0] \leq a^{\nu} \tag{1.32}$$

Now the total failure probability of $\mathrm{SD\scriptstyle{O}\displaystyle U}$ is given by

$$\Pr[\text{SDoU fails}] \leq n \cdot a^{\nu} = n^{-c_6}$$

where

$$n \cdot a^{\nu} = n^{-c_6} \Rightarrow \nu = \frac{c_6 + 1}{\log(1/a)} \cdot \log n \tag{1.33}$$

□

As indicated in [53], the time cost for such an indirect message propagation is given by

$$T_{\mathrm{SDoU}} = \mathcal{O}\left(h \cdot \log n \cdot \max\{\mathbf{b}, \mu_k\}\right) \tag{1.34}$$

4.2.7 A tentative agreement protocol. The purpose of an agreement protocol in a decentralized, fault-prone setting, is to have all the live processors agree in a specific value, or participate in the evaluation of a function whose outcome will be available to all the participants.

According to the BSP AGREEMENT protocol, each live processor achieves an InitialValue and subsequently, an agreement rule (e.g. majority, median, e.t.c.) given as an input of this protocol is employed, in order to have all the live processors agree in a unique value. For example, consider that we have to make a unique estimation of the number of currently live processors, common for all the live processors. At the beginning of the protocol, the InitialValue is calculated by each live processor (e.g. a RANDOM SAMPLING is applied so as each live processor to have an estimation of the number of live processors). Subsequently, some rounds of agreement are performed, in order for all the live processors to end up with the same value. In each round, each live processor estimates a new value, according to the agreement rule, that depends on the nature of the problem to be solved. For example, a “median" agreement rule for the choice of a unique random seed would destroy the randomness of common seed. On the other hand, if the problem is the estimation of the number of the currently live processors, A_{k}, then a “weighted median" among sampled values seems more reasonable than a “majority" rule that would have no sense in this case. As an example, we have used this technique for the estimation of the live processors of $\mathcal{RM}$ at the beginning of a new PRIMARY JOB, before discarding the excessive load of each live processor, according to this estimation.

Lemma 25 *The requested number of rounds of the* BSP AGREEMENT *protocol until all the processors have agreed (with polynomially small deviation probability) in a estimation of* A_{k}*, is* $\mathcal{O}(\frac{\log n}{\log \mathtt{N}})$*, where* $\mathtt{N}$ *is the size of the samples of the* RANDOM SAMPILNG *applications.*

Proof: We proceed in our analysis by following the BSP AGREEMENT routine step by step (see figure 1.7):

[1]	**InitialValues:** Each processor estimates independently an initial value (e.g. for estimating the number of currently live processors, $P_{\rm i}$ sends $k_1 \log n$ messages to randomly chosen destinations).	
[2]	**PARDO** x times:	
[2.1]		**Agreement:** Each processor sends $\log n$ messages to random target processors.
[2.2]		Each live processor calculates the new estimation as $NewEstimation = AgreementRule(samples)$

Figure 1.7 The BSP Agreement Protocol.

STEP 1. Each live processor $P_{\rm j}$ samples $\mathtt{N}$ real processors from $\mathcal{RM}$. Consider one indicator variable $X_{\rm i,j}$ for the $i^{\rm th}$ sample of $P_{\rm j}$, which is

$$X_{\rm i,j} = \begin{cases} 1, & \text{if the } i^{\rm th} \text{ sample of } P_{\rm j} \text{ corresponds to a live processor} \\ 0, & \text{otherwise.} \end{cases}$$

The RANDOM SAMPLING is actually the implementation of 2 $\mathcal{O}(\mathtt{N})$-relations, one during which each live processor polls $\mathtt{N}$ processors chosen u.a.r. from $\mathcal{RM}$ (with replacement), and one for getting the answers by the live polled processors. If at the beginning of $VS_{\rm k}$ the number of live processors is $A_{\rm k}$ and the RANDOM SAMPLING is performed at the beginning of $VS_{\rm k}$, then the probability of each sample hitting a live processor will be upper-bounded by $p_a \equiv \frac{A_{\rm k}}{n}$. Thus, $\Pr[X_{\rm i,j} = 1] \leq p_a$, and $\Pr[X_{\rm i,j} = 0] \geq 1 - p_a$. And since the RANDOM SAMPLING is a relatively fast routine p_a may be considered to be a very good approximation of the actual ratio of live processors in $\mathcal{RM}$ at the end of the RANDOM SAMPLING. Thus, these indicator variables are clearly i.i.d. (random samples with replacement from the same sample space) with success probability very close to p_a, which implies that

$$E[X_{\rm i,j}] = p_a, \text{ and } Var[X_{\rm i,j}] = p_a \cdot (1 - p_a), \forall i \in [\mathtt{N}], j \in [n].$$

So $P_{\rm j}$'s first estimation of the ratio of live processors in $\mathcal{RM}$ is given by the following relation:

$$R_{\rm j}^{(0)} = \frac{1}{\mathtt{N}} \cdot \sum_{\rm i=1}^{\mathtt{N}} X_{\rm i,j}.$$

Obviously, the expected value and the variance of these new random variables are

$$\begin{aligned} E[R_{\rm j}^{(0)}] &= p_a, \text{ and} \\ Var[R_{\rm j}^{(0)}] &= \frac{1}{\mathtt{N}^2} \cdot \left(\sum_{\rm i=1}^{\mathtt{N}} Var[X_{\rm i,j}] - 2 \cdot \sum_{\rm i>k} Cov(X_{\rm i,j}, X_{\rm k,j}) \right) = \frac{p_a \cdot (1 - p_a)}{\mathtt{N}} \end{aligned}$$

STEP 2. Each live processor $P_{\rm j}$ randomly selects processors from $\mathcal{RM}$, until it gets $\mathtt{N}$ live answers (this implies that each live processor will have to send $\frac{c}{1-a} \cdot \mathtt{N}$ polling messages to randomly chosen targets, and will accept the first $\mathtt{N}$ answers). The new estimation of $P_{\rm j}$ is

$$R_{\rm j}^{(1)} = \frac{1}{\mathtt{N}} \cdot \sum_{\rm i=1}^{\mathtt{N}} R_{\lambda_{\rm i}}^{(0)}$$

($P_{\rm j}$ has accepted the answers of $P_{\lambda_1}, P_{\lambda_2}, \ldots, P_{\lambda_{\mathtt{N}}}$). For this new estimation we have

$$E[R_{\rm j}^{(1)}] = p_a, \text{ and } Var[R_{\rm j}^{(1)}] = \frac{1}{\mathtt{N}^2} \cdot \left(\sum_{\rm i=1}^{\mathtt{N}} Var[R_{\lambda_{\rm i}}^{(0)}] - 2 \cdot \sum_{\rm i>k} Cov(R_{\lambda_{\rm i}}^{(0)}, R_{\lambda_{\rm k}}^{(0)}) \right)$$

Since $R_{\lambda_1}^{(0)}, P_{\lambda_2}^{(0)}, \ldots, R_{\lambda_{\mathtt{N}}}^{(0)}$ are variables randomly chosen (with replacement) from the same sample space, they are independent with each other, and this implies that $Cov(R_{\lambda_{\rm i}}^{(0)}, R_{\lambda_{\rm k}}^{(0)}) = 0, \forall j,k \in [\mathtt{N}]$. So, we have that

$$Var[R_{\rm j}^{(1)}] \quad = \quad \frac{1}{\mathtt{N}^2} \cdot \sum_{\rm i=1}^{\mathtt{N}} Var[R_{\lambda_{\rm i}}^{(0)}] = \frac{p_a \cdot (1-p_a)}{\mathtt{N}^2} \tag{1.35}$$

STEP 3:. Repeat STEP 2 x times. After these x repetitions we shall have:

$$\begin{aligned} E[R_{\rm i}^{(x+1)}] &= p_a \\ Var[R_{\rm j}^{(x+1)}] &= \frac{p_a \cdot (1-p_a)}{\mathtt{N}^{x+2}} \end{aligned} \tag{1.36}$$

Each processor $P_{\rm j}$ that is live estimates the number of live processors in $\mathcal{RM}$ at the beginning of the Virtual Superstep $VS_{\rm k}$ as $A_{\rm j,k} = n \cdot R_{\rm j}^{(x+1)}$, where

$$E[A_{\rm j,k}] = n \cdot p_a, \text{ and } Var[A_{\rm j,k}] = n^2 \cdot Var[R_{\rm j}^{(x+1)}] = \frac{n^2 p_a(1-p_a)}{\mathtt{N}^{x+2}} \leq \frac{n^2}{4\mathtt{N}^{x+2}}$$

In order to have small deviations among the live processors' estimations, the following must hold:

$$\lim_{n \to \infty} \frac{n^2}{4\mathtt{N}^{x+2}} = 0 \Rightarrow x = \mathcal{O}\left(\frac{\log n}{\log \mathtt{N}}\right), \tag{1.37}$$

considering that $\mathtt{N} = \log n$. For example, if $x = \frac{c \log n}{\log \mathtt{N}} - 2$ then $Var[A_{\rm j,k}] = \frac{n^{-\rm c+2}}{4}$, and by applying Chebysev's Inequality we have:

$$\Pr[|A_{\rm j,k} - p_a \cdot n| \geq 1] \leq \frac{n^{-2\rm c+4}}{16} \tag{1.38}$$

□

For the RANDOM SAMPLING that estimates the InitialValues of the live processors, an $\mathcal{O}(\log n)$-relation is necessary for the polling of a size-$(\log n)$ sample (each live processor throws independently and randomly $\mathcal{O}(\log n)$ polling messages which are sent to distinct targets with overwhelming probability (see the `Sparse Occupancy Problem` in full paper), and an $\mathcal{O}(\log n)$-relation for the responses of the sampled processors (since the choices of samples are random, the expected number of pollings for each processor is $\mathcal{O}(\log n)$ and the deviation is very sharp around the expected number).

Thus the cost for RANDOM SAMPLING is

$$T_{\text{RandomSampling}} = \mathcal{O}(\log n) \cdot \max\{L, g\} \tag{1.39}$$

For each of the BSP AGREEMENT steps, which is actually a new RANDOM SAMPLING among the processing elements of $\mathcal{RM}$, $\mathcal{O}(\log n)$-relations are required for each round, and the agreement rule is a "weighted median" operation among the samples of each live processor.

Remark: Similar uses of such an agreement protocol among live processors that cannot possibly know eahc other might be useful for the probabilistic estimation of other functions, such as boolean conditions. For example, the FINISHED? condition of the Robust-$\mathcal{BSP}$ simulation strategy might be able to be estimated with small failure probability by such an agreement protocol.

Acknowledgments

The authors wish to thank Dr. Spyros C. Kontogiannis (email: `kontog@cti.gr`) who has also co-authored Section 4 of this Chapter. This work was partially supported by the EU IST FET-OPEN Project ALCOM-FT . The authors wish to thank the Computer Technology Institute (CTI) for generous support.

Notes

1. In some works the malicious faults are also called byzantine faults.

2. The existence of smulations of arbitrary PRIORITY CRCW PRAM steps (which is the strongest PRAM variant) by at most $\mathcal{O}(\log n)$ EREW PRAM steps, implies corresponding results for the weakest variant of this cost model as well.

3. Notice that we do not exploit any slackness in our setting, which might indicate an "optimal" use of a subset of $\mathcal{RM}$ for the improvement of our results. This is because we want to have a realistic measure of the performance of our simulation process.

4. For example, if $T_{\rm s}$ is the total time of the simulation process, then $r(a) = \mathcal{O}\left(\frac{\text{polylog(n)}}{T_{\rm s}} \cdot a\right)$.

5. The size of each epoch can be as large as $K_0 = \Theta(n)$, as it will be demonstrated later on.

6. As it will be shown in the sequel, a double logarithmic number of SECONDARYJOBS is enough for our strategy to work.

7. The calls of LOCAL STORAGE are not shown in the Robust-$\mathcal{BSP}$ strategy, because they are performed at the end of each PRIMARY JOB and each SECONDARY JOB.

8. The size of these neighborhoods of real processors depends on the fault occurrences up to the specific point of the simulation process.

9. This also implies that the size of the neighborhoods will be at most $\mathcal{O}(\log^2 n)$ with subpolynomially small failure probability.

10. Recall that the optimal work load would be $\frac{n}{A_k}$ threads per live processor, and thus $T_{\rm OPT} = \frac{n}{A_k} \cdot (T_{\rm LC} + T_{\rm Comm})$.

11. Recall that K_0 can be as large as $\Theta(n)$.

References

[1] N. Alon and J. Spencer, "The Probabilistic Method", John Wiley and Sons, 1992.

[2] D. Angluin and L. Valiant, "Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings", J. of Comput. Systems Sci., vol. 18, pp. 155–193, 1979.

[3] Y. Aumann, M. Bender and L. Zhang, "Efficient execution of non-deterministic parallel programs on asynchronous systems", Proc. of the 8th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 270–276, 1996.

[4] Y. Aumann, Z. Kedem, K. Palem and M. Rabin, "Highly efficient asynchronous execution of large-grained parallel programs", Proc. of the 34th Annual Symposium on Foundations of Computer Science (FOCS), 1993.

[5] D. Bauer, F. Boesch, C. Suffel and R. Tindell, "Connectivity extremal problems and the design of reliable probabilistic networks", The Theory of Applications of Graphs, John Wiley and Sons, 1981.

[6] A. Baumker, W. Dittrich and F. Meyer auf der Heide, "Truly efficient parallel algorithms: c-optimal multisearch for an extension of the BSP model", Proc. of the 3rd Ann. European Symposium on Algorithms (ESA), Springer Verlag LNCS 979, pp. 17–30, 1995.

[7] P. Berenbrink, F. Meyer auf der Heide and V. Stemann, "Fault-tolerant shared memory simulations", Proc. of the 13th Ann. Symp. on Theoretical Aspects of Computer Science (STACS), Springer Verlag, pp. 181–192, 1996.

[8] R. Blumofe and C. Leiserson, "Scheduling multithreaded computations by work stealing", Proc. of the 35th Annual Symposium on Foundations of Computer Science (FOCS), pp. 356–368, 1994.

[9] F. T. Boesch, F. Harary and J. A. Cabell, "Graphs and models of communication networks vulnerability: connectivity and persistence", Networks, vol. 11, pp. 57–63, 1981.

[10] B. Bollobás, "Random Graphs", Academic Press, 1985.

[11] B. Bollobás, "A probabilistic proof of an asymptotic formula for the number of labeled regular graphs", Europ. J. Combinatorics, vol.1, pp.311–316, 1980.

[12] B. Bollobás, "The distribution of the maximum degree of a random graph", Discrete Math. 32, pp. 201–203, 1980.

[13] B. Bollobás and A. G. Thomason , "Random Graphs of small order", Random Graphs, Annals of Discr. Math., pp.47–97, 1985.

[14] A. Broder and E. Shamir, "On the second eigenvalue of random regular graphs", Proc. 19st ACM Symp. on Theory of Computing (STOC), pp. 286–294, 1987.

[15] H. Chernoff, "A measure of the asymptotic efficiency for tests of a hypothesis based on the sums of observations", Annals of Mathematical Statistics, 23, pp. 493–509, 1952.

[16] B. Chlebus, A. Gambin and P. Indyk, "Shared-Memory Simulations on a Faulty DMM", Proc. of Intl. Colloquium on Automata, Languages and Programming (ICALP), 1996.

[17] B. Chlebus, A. Gambin and P. Indyk, "PRAM computations resilient to memory faults", Proc. of the 2nd Ann. European Symp. on Algorithms (ESA), Springer Verlag LNCS 855, pp. 401–412, 1994.

[18] B. Chlebus, L. Gasieniec and A. Pelc, "Fast Deterministic Simulation of Computations on Faulty Parallel Machines", Proc. of the 3rd Ann. European Symp. on Algorithms, Springer Verlag LNCS 979, pp. 89–101, 1995.

[19] F. Chyzak, Ph. Flajolet and B. Salvy, "Studies in automatic Combinatorics, Volume II", INRIA 1998. Available at URL
http://pauillac.inria.fr/algo/libraries/autocomb/autocomb.html

[20] T. Cormen, C. Leiserson and R. Rivest, "Introduction to Algorithms", MIT Press, 1990.

[21] P. Dasgupta, Z. Kedem and M. Rabin, "Parallel processing on networks of workstations: a fault tolerant, high performance approach", Proc. of the the 15th International Conference on Distributed Systems, 1995.

[22] R. De Prisco, A. Mayer and M. Yung, "Time-Optimal Message-Efficient Work Performance in the Presence of Faults", Proc. of the ACM Symposium of Distributed Computing, 1994.

[23] D. Dubhashi, V. Priebe and D. Ranjan, "Negative Dependence through the FKG Inequality", BRICS Report Series, RS-96-27, ISSN 0909-0878.

[24] D. Dubhashi and D. Ranjan, "Balls and Bins: A study in Negative Dependence", BRICS Report Series, RS-96-25, ISSN 0909-0878.

[25] M.Dyer and A.Frieze, "The Solution of Some Random NP-hard Problems in Polynomial Expected Time", Journal of Algorithms, vol. 10, pp. 451–489, 1989.

[26] P. Erdös and A. Renyi, "On the evolution of random graphs", Magyar Tud. Akad. Math. Kut. Int. Kozl. 5, pp. 17–61, 1960.

[27] P. Flajolet, K. Hatzis, S. Nikoletseas and P. Spirakis, "Trade-offs between Density and Robustness in Random Interconnection Graphs", to appear *in the Proceedings of the IFIP International Conference on Theoretical Computer Science (TCS2000)*, Sendai, Japan, 2000.

[28] Ph. Flajolet and R. Sedgewick, "An introduction to the Analysis of Algorithms", Addison Wesley, 1996.

[29] Ph. Flajolet and R. Sedgewick, "Analytic Combinatorics", book in preparation, expected in 1999. (Individual chapters are available as INRIA Research Reports 1888, 2026, 2376, 2956, 3162).

[30] J. Friedman, J. Kahn and E. Szemeredi, "On the second eigenvalue of random regular graphs", Proc. 21st ACM Symp. on Theory of Computing (STOC), pp. 286–294, 1989.

[31] A. Frieze, "Probabilistic Analysis of Graph Algorithms", Feb. 1989.

[32] Z. Galil, A. Mayer, M. Yung, "Resolving Message Complexity of Byzantine Agreement and beyond", Proc. of the 36th IEEE Annual Symposium on Foundations of Computer Science (FOCS), 1995.

[33] J. Garofalakis, S. Rajsbaum, P. Spirakis and B. Tampakas, "Tentative and Definite Distributed Computations: An Optimistic Approach to Network Synchronization", Journal of Theoretical Computer Science (TCS), Volume 128 (Issues 1-2), pp. 63–74, 1994.

[34] M.Garey and D.Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H.Freeman and Co., NY, 1979.

[35] A. Goerdt, "Random Regular Graphs with Edge Faults: Expansion through Cores", Proc. of ISAAC 1998.

[36] A. Gerbessiotis and C. Siniolakis, "Communication efficient data structures on the BSP model with Applications", Oxford Univ. Technical Report, PRG-TR-13-96, May 1996.

[37] M. Gereb-Graus and T. Tsantilas, "Efficient optical communication in parallel computers", Proc. of the 4th Ann. ACM Symp. on Parallel Algorithms and Architectures (SPAA), pp. 41–48, 1992.

[38] A. Gerbessiotis and L. Valiant, "Direct Bulk-Synchronous Parallel Algorithms", Journal of Parallel and Distributed Computing, 22, pp. 251–267, 1994.

[39] C. Gkantsidis, K. Hatzis, S. Nikoletseas and P. Spirakis, "Dominating Sets in Sparse Random Graphs", Computer Technology Institute, Technical Report, November 1998.

[40] T. Hagerup and C. Rüb, “A guided tour of Chernoff bounds”, Information Processing Letters, 33, pp. 305–308, 1989/1990.

[41] J. Hromkovic, R. Klasing, E. Stoehr and H. Wagener, “Gossiping in Vertex-Disjoint Paths Mode in d-dimensional Grids and Planar Graphs”, Proc. 1st ESA, pp. 200-211, LNCS vol. 726, 1993.

[42] S.Janson, D. Knuth, T. Łuczak and B. Pittel, “The Birth of the Giant Component”, Random Structures and Algorithms, vol. 4, pp. 232–355, 1993.

[43] A. Kamath, R. Motwani,K. Palem and P. Spirakis, “Tail bound for occupancy and the satisfiability threshold conjecture”, Proc. of the 35th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp. 592–603, 1994.

[44] P. Kanellakis and A. Shvartsman, “Fault-Tolerant Parallel Computation”, Kluwer Academic Publishers, ISBN 0-7923-992-6, 1997.

[45] P. Kanellakis and A. Shvartsman, “Efficient parallel algorithms on restartable fail-stop processors”, In Proceedings of the 10th Ann. ACM Symposium on Principles of Distributed Computing (PODC), pp. 23–36, 1991.

[46] P. Kanellakis and A. Shvartsman, “Efficient parallel algorithms can be made robust”, Distributed Computing, 5, pp. 201-217, 1992.

[47] R. Karp and M. Sipser, “Maximum matching in sparse random graphs”, 22nd FOCS, pp. 364–375, 1981.

[48] Z. Kedem, K. Palem, and P. Spirakis, “Efficient Robust Parallel Computations”, Proc. 22nd ACM Symp. on Theory of Computing (STOC), pp. 138–148, 1990.

[49] Z. Kedem, K. Palem, A. Raghunathan and P. Spirakis, “Combining Tentative and Definite Executions for Very Fast Dependable Parallel Computing”, Proc. 23nd ACM Symp. on Theory of Computing (STOC), 1991.

[50] Z. Kedem, K. Palem, P. Spirakis and M. Yung, “Faulty Random Graphs: reliable efficient-on-the-average network computing”, Computer Technology Institute (Patras, Greece), Technical Report, 1993.

[51] D. E. Knuth, “The Art of Computer Programming”, vol. 1, 2nd edition, Addison Wesley, 1973.

[52] S. Kontogiannis, G. Pantziou and P. Spirakis, “Efficient Computations on Fault-Prone BSP Machines”, Proc. of the 9th ACM Symposium on Parallel Algorithms and Architectures, 1997. (Full version at `http://www.ceid.upatras.gr/~kontog/bsp/spaa97/`).

[53] S. Kontogiannis, G. Pantziou, P. Spirakis and M. Yung, “Dynamic -Fault-Prone BSP: a paradigm for robust computations in changing environ-

ments", Full version at
`http://www.ceid.upatras.gr/~kontog/bsp/spaa98/`

[54] F. T. Leighton, B. M. Maggs and R. K. Sitaraman, "On the Fault Tolerance of some popular Bounded-Degree networks", SIAM J. on Computing, Vol. 27 (5), 1998.

[55] C. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing", IEEE Transactions on Computers, C-34 (10):892–900, 1985.

[56] W.F. McColl, "Scaleable Parallel Computing: A grand unified theory and its practical development", Proc. of the IFIP World Congress, 1, pp. 539–546, Hamburg, August, 1994.

[57] R. Motwani, "Expanding graphs and the average-case analysis of algorithms for matching and related problems", 21st STOC, pp. 550–561, 1989.

[58] R. Motwani and P. Raghavan, "Randomized Algorithms", Cambridge University Press, 1995.

[59] S. Nikoletseas, G. Pantziou, P. Psycharis and P. Spirakis, "On the reliability of fat-trees", 3rd International European Conference on Parallel Processing (Euro-Par), pp. 208 – 217, Passau, Germany, 1997.

[60] S. Nikoletseas and P. Spirakis, "Near-Optimal Dominating Sets in Dense Random Graphs in Polynomial Expected Time", 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), 1993.

[61] S. Nikoletseas, J. Reif, P. Spirakis and M. Yung, "Stochastic Graphs Have Short Memory: Fully Dynamic Connectivity in Poly-Log Expected Time", 22nd ICALP, pp. 159 – 170, 1995.

[62] S. Nikoletseas, K. Palem, P. Spirakis and M. Yung, "Short Vertex Disjoint Paths and Multiconnectivity in Random Graphs: Reliable Network Computing", 21st International Colloquium on Automata, Languages and Programming (ICALP), Jerusalem, pp. 508 – 515, 1994.

[63] S. Nikoletseas, K. Palem, P. Spirakis and M. Yung, "Connectivity Properties in Random Regular Graphs with Edge Faults", in the Special Issue on Randomized Computing of the International Journal of Foundations of Computer Science (IJFCS), 2000.

[64] S. Nikoletseas and P. Spirakis, "Expander Properties in Random Regular Graphs with Edge Faults", 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS), München, pp. 421 – 432, 1995.

[65] N. Pippenger, "Telephone switching networks", The Mathematics of Networks, AMS, Providence, 1982.

[66] M. Rabin, “Efficient dispersal of information for security, load balancing and fault tolerance”, Journal of the Association for Computing Machinery, Vol. 36, No. 2, pp. 335-348, April 1989.

[67] A. Sinclair, “Algorithms for random generation and Counting”, ed. Birkhauser, 1992.

[68] M. Sipser and D. Spielman, “Expander codes”, Proc. of 35th Ann. Symp. on Foundations of Computer Science, pp. 566–576, 1994.

[69] J. Spencer, “Ten Lectures on the Probabilistic Method”, SIAM, 1987.

[70] P. Spira and A. Pan, “On finding and updating spanning trees and shortest paths”, SIAM J. Comput., 4, pp. 375–380, 1975.

[71] R. Tarjan, “Data structures and Network Algorithms”, 1985.

[72] P. Turán, “On an extremal problem in graph theory”, *Matematicko Fizicki Lapok*, 48, pp. 436–452, 1941.

[73] L. Valiant, “A Bridging Model for Parallel Computation”, CACM, vol. 33, no. 8, pp. 103–111, 1990.

[74] L. Valiant. “General purpose parallel architectures”, Handbook of Theoretical Computer Science, J. van Leeuwen ed., North Holland, 1990.

[75] N. Wormald, “The asymptotic distribution of short cycles in random regular graphs”, Journal of Combinatorial Theory (B) 31, pp. 168 – 182, 1981.