November 1990

UILU-ENG-90-2255
CRHC-90-16

Center for Reliable and High-Performance Computing

PROBABILISTIC ANALYSIS AND ALGORITHMS FOR RECONFIGURATION OF MEMORY ARRAYS

Weiping Shi
W. Kent Fuchs

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY none	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE none	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-90-2255 CRHC-90-16	5. MONITORING ORGANIZATION REPORT NUMBER(S) none
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Semiconductor Research Corporation (SRC)
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) P.O. Box 12053 Research Triangle Park, NC 27709
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SRC 8b. OFFICE SYMBOL (If applicable) N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 89-DP-109
8c. ADDRESS (City, State, and ZIP Code) P.O. Box 12053 Research Triangle Park, NC 27709	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) Probabilistic Analysis and Algorithms for Reconfiguration of Memory Arrays	
12. PERSONAL AUTHOR(S) Shi, Weiping and Fuchs, W. Kent	
13a. TYPE OF REPORT Technical 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1990 May 15. PAGE COUNT 22
16. SUPPLEMENTARY NOTATION none	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	average-case analysis, random graphs, repairable memories, reconfigurable arrays, reconfiguration algorithms

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Reconfiguration of memory arrays with spare rows and columns has been shown to be an NP-complete problem. Numerous heuristics for repairing memories have been recently proposed, however no average-case analysis has been published. This paper presents the first fundamental analysis of average-case time complexities of several existing heuristics. We also present the first provably average-case polynomial time algorithm for reconfiguration of memories with spare rows and columns. The implemented algorithm runs significantly faster than existing heuristics for most cases examined.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Probabilistic Analysis and Algorithms for Reconfiguration of Memory Arrays[1]

Weiping Shi and W. Kent Fuchs

Center for Reliable and High Performance Computing
Coordinated Science Laboratory
1101 W. Springfield Avenue
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Correspondent: W. Kent Fuchs
E-mail: fuchs@bach.csg.uiuc.edu
Phone: (217) 333-9731
Fax: (217) 244-1764

Abstract

Reconfiguration of memory arrays with spare rows and columns has been shown to be an NP-complete problem. Numerous heuristics for repairing memories have been recently proposed, however no average-case analysis has been published. This paper presents the first fundamental analysis of average-case time complexities of several existing heuristics. We also present the first provably average-case polynomial time algorithm for reconfiguration of memories with spare rows and columns. The implemented algorithm runs significantly faster than existing heuristics for most cases examined.

Keywords. Average-case analysis, random graphs, repairable memories, reconfigurable arrays, reconfiguration algorithms.

[1]This research was supported by the Semiconductor Research Corporation under Contract 89-DP-109.

1 Introduction

Algorithms for optimal repair of reconfigurable arrays using spare rows and columns has recently become an intensively studied problem. The general problem is the repair of defective elements in an m-by-n array with a set of spare rows and spare columns. A practical application of this problem is the repair of large memory integrated circuits in order to enhance manufacturing yield. A defective element is repaired by replacing either the entire row or the entire column containing the defective element with a spare row or column. A repair (reconfiguration) solution is obtained when all defective elements in the m-by-n array have been repaired. Kuo and Fuchs have shown that the general problem is NP-complete [11]. Recently, numerous heuristics with worst case exponential time complexity have been proposed for this problem. The state-of-the-art is described in the review paper by Fuchs and Chang [6], and papers by Hemmady and Reddy [10], Shen and Lombardi [13], and Wey and Lombardi [15].

Most existing reconfiguration heuristics are front ended with one or both of the following two strategies: early-abort or partial solution. The heuristics are then typically followed by, basically, an exponential time exhaustive search. In the *early abort* approach, the standard heuristic is to use a polynomial time algorithm to eliminate as many unrepairable structures as possible, thereby reducing the number of problems sent to the exhaustive search. In the *partial solution* approach, a polynomial time algorithm is employed to detect as many *mandatory repairs* as possible, thereby reducing the problem size sent to the final exhaustive search.

Although the worst-case time complexities of these complete reconfiguration algorithms are all exponential, very little is known about their average-case time complexity. So far the only published approach to analyzing the average case performance of these algorithms has been through ad-hoc

experiments and examples. Little is known about their fundamental average-case performance particularly as array sizes become large.

An analysis of the average-case time complexity must be based on some distribution of failures in the reconfigurable arrays. In our analysis we assume each memory cell has the same probability of being faulty. Situations where an entire row or column is faulty are not discussed here, since these kinds of faults can be detected easily and repaired before the reconfiguration algorithm begins.

In Section 3, we show that when the failure rate of each cell is too high compared to the number of spares, then the array is almost never repairable. We also show that when the failure rate is too low compared to the number of spares, then the array can be trivially repaired, but spares are wasted. We determine the number of rows and columns and the failure rate for which the resulting yield is satisfactory and spares are not wasted. In Sections 4 and 5, a probabilistic analysis of these heuristics is presented and their performance under different failure rates are estimated. Finally in Section 6, we give an algorithm which runs in $O(n^2)$ average-case time, based on a failure rate which is neither too high nor too low.

2 Random Graph Model

Throughout the paper, we assume the readers are familiar with basic graph theory terminology [8] and algorithmic terminology [1]. For a given array, we construct a bipartite graph $G = (V_1 \cup V_2, E)$. The vertices of V_1 and V_2 correspond to rows and columns in the array respectively. If array element (i, j) is faulty, then we assign an edge between vertex i in V_1 and vertex j in V_2. For simplicity, we assume $|V_1| = |V_2| = n$, i.e., the array is of size $n \times n$ elements. We also assume the number of spare

rows is αn and the number of spare columns is βn, where $0 < \alpha, \beta < 1$ are constants. It will be clear later in the paper that if these assumptions do not hold, the argument can be modified slightly to obtain similar results.

A subset of vertices $C \subseteq V_1 \cup V_2$ is called a *vertex cover* if for every edge $(v_i, v_j) \in E$, either $v_i \in C$ or $v_j \in C$. A subset of vertices C is called a bipartite vertex cover of size (s, t), if C is a vertex cover and $|C \cap V_1| \leq s$ and $|C \cap V_2| \leq t$. The repair problem is to find a bipartite vertex cover of size $(\alpha n, \beta n)$ for the given graph. It is not hard to see that this repair problem, though not identical to the problem of Kuo and Fuchs [11], is still NP-complete.

We assume each single fault is limited to a single element, each fault appears with equal probability $p(n)$ and faults are statistically independent of each other. Correspondingly, in our bipartite random graph model, this assumption implies that for each pair of vertices $v_i \in V_1$ and $v_j \in V_2$, with probability $p(n)$ there is an edge between v_i and v_j, where $n = |V_1| = |V_2|$. We call $p(n)$ the *edge probability*. A random bipartite graph will be described as $G_p(n) = (V_1 \cup V_2, E)$, where $n = |V_1| = |V_2|$.

Throughout the paper, the phrase *almost always* or *almost every* means *with probability* 1 *as* $n \rightarrow \infty$. For example, if we say almost all graphs have property Q, that means $\lim_{n\rightarrow\infty} P\{G_p(n)$ *has* $Q\} = 1$. When we say an algorithm almost always runs in polynomial time, we mean with probability 1 as $n \rightarrow \infty$, the algorithm runs in polynomial time. Similarly, the phrase *almost never* or *almost no* means *with probability* 0 *as* $n \rightarrow \infty$.

Let the set of graphs we are studying be $\mathbf{G}$, which is a subset of bipartite graphs. We define a *graph property* to be a subset of $\mathbf{G}$. An important fact in random graph theory is that most properties appear rather suddenly: for some $p = p(n)$, almost no G_p has Q while for "slightly" larger p almost

every G_p has Q [3, 14]. This is exactly the case here, when $p(n)$ is low compared to the number of spares, almost all arrays are repairable. When $p(n)$ is high compared to the number of spares, almost no array is repairable. Formally, we have the following definition.

Definition 2.1 A threshold function for property Q is a function $p(\epsilon, n)$ such that almost every $G_{p(\epsilon,n)}$ has Q if $\epsilon > 0$ and almost no $G_{p(\epsilon,n)}$ has Q if $\epsilon < 0$, where ϵ is a constant.

This definition is different from those in Bollobás [3] and Spencer [14], but satisfies our need and is easy to deal with technically.

Finally, there are some basic preliminaries from probability theory [5] that are important in random graph analysis. *Markov's Inequality* states that if ξ is a non-negative random variable and $t > 0$, then $P\{\xi \geq t\} \leq E(\xi)/t$, in particular, if ξ is integer valued, then $E(\xi) \to 0$ implies $P(\xi = 0) \to 1$. Also, if ξ is a non-negative random variable and $t > 0$, then $P\{|\xi - E(\xi)| \geq t\} \leq Var(\xi)/t^2$, or equivalently, $P\{|\xi - E(\xi)| < t\} \geq 1 - Var(\xi)/t^2$. This is known as *Chebyshev's Inequality.*

3 Repairability in Terms of Failure Rate and Spares

In this section, we examine the relationship between failure rate $p(n)$, the number of spares and the repairability. It will be shown that when $p(n) < c/n$, the array is almost always easily repairable, and when $p(n) > c'/n$, the array is almost never repairable, where c and c' are constants depending on the number of spares. In other words, the threshold function for the row/column repair to be effective is near $\Theta(1/n)$.

Theorem 3.1 If the edge probability $p(n) = c/n$, where $c > \ln 4/(1-\alpha)(1-\beta)$, then almost every bipartite graph G_p has no bipartite vertex cover of size $(\alpha n, \beta n)$.

Proof. Let X be a random variable denoting the number of bipartite vertex covers of size $(\alpha n, \beta n)$, and X_i be a $\{0,1\}$-random variable denoting whether a specific choice of αn and βn vertices is a vertex cover. Then

$$\begin{aligned} E(X) &= \sum_i E(X_i) = \binom{n}{\alpha n}\binom{n}{\beta n} P\{X_i = 1\} \\ &= \binom{n}{\alpha n}\binom{n}{\beta n}(1-p(n))^{(1-\alpha)n(1-\beta)n} \\ &\leq 2^n 2^n (1-p(n))^{(1-\alpha)(1-\beta)n^2} \\ &= \left(4\left(1-\frac{c}{n}\right)^{(1-\alpha)(1-\beta)n}\right)^n = \left(4e^{-c(1-\alpha)(1-\beta)}\right)^n \end{aligned}$$

Since $c > \ln 4/(1-\alpha)(1-\beta)$, $E(X) \to 0$. From Markov's inequality, $\lim_{n\to\infty} P\{X = 0\} = 1$. □

If the failure rate p is constant, then intuitively it would seem that the array is repairable if $\alpha+\beta-\alpha\beta \gg p$, because the total number of spare elements is $(\alpha+\beta-\alpha\beta)n^2$, and the total number of faulty elements is approximately pn^2. The theorem says that this is not the case.

We can also determine when the arrays are almost always repairable.

Theorem 3.2 Almost every bipartite graph $G_p(n)$ has a bipartite vertex cover of size $(\alpha n, \beta n)$, if

$$p(n) = \frac{\alpha+\beta}{n} - \frac{\omega(n)}{n^{3/2}} \tag{1}$$

where $\omega(n)$ is any function such that $\lim_{n\to\infty} \omega(n) = \infty$.

Proof. We show the total number of edges is almost always less than $(\alpha+\beta)n$, if $p(n)$ satisfies (1). Let X be a random variable denoting the number of edges, then X obeys the binomial distribution

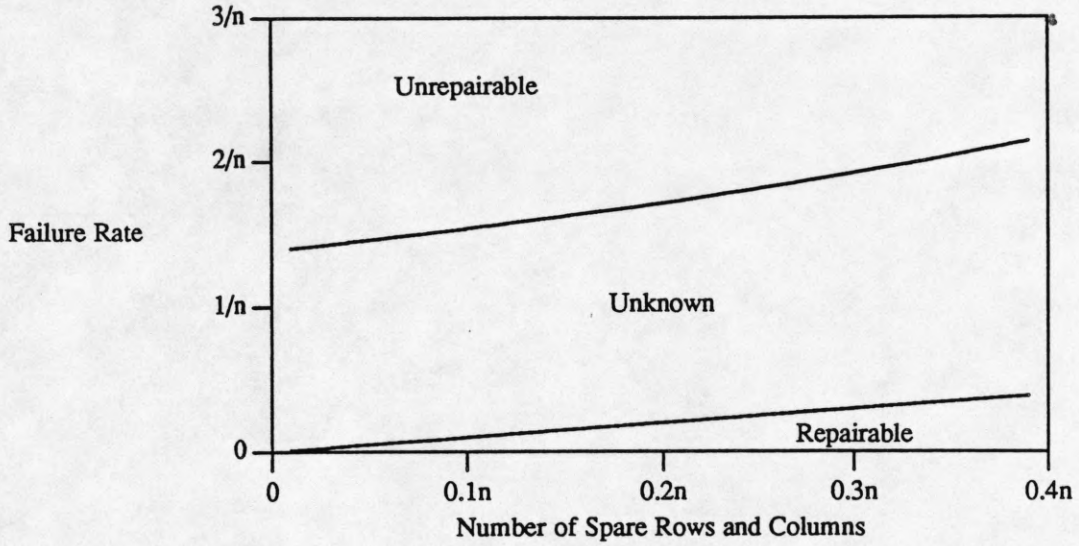

Figure 1: Regions where the array is almost always repairable and almost never repairable.

and $E(X) = n^2 p(n)$, $Var(X) = n^2 p(n)(1 - p(n))$. Since

$$\begin{aligned} P\{X < (\alpha+\beta)n\} &\geq P\{(\alpha+\beta)n - 2\sqrt{n}\omega(n) < X < (\alpha+\beta)n\} \\ &= P\{|X - E(X)| < \sqrt{n}\omega(n)\} \end{aligned}$$

From Chebyshev's inequality,

$$\begin{aligned} P\{X < (\alpha+\beta)n\} &\geq 1 - \frac{Var(X)}{n\omega^2(n)} = 1 - O\left(\frac{1}{\omega^2(n)}\right) \\ &= 1 \quad as\ n \to \infty \end{aligned}$$

□

The results in Theorem 3.1 and 3.2 are illustrated in Figure 1, where the horizontal axis is the total number of spares and $\alpha = \beta$.

The above discussion does not consider the failure of spares. If the spare elements have the same failure probability as the array elements, then the problem is given an n-by-n array, to find a

sub-array of size αn-by-βn, for some fixed $\alpha, \beta \in (0,1)$. Fuja and Heegard gave an estimation for the special case where p is a constant [7]. Write $\overline{K}_{n_1,n_2}$ as the complement of the complete bipartite graph K_{n_1,n_2}. In other words, $\overline{K}_{n_1,n_2}$ is an empty graph on n_1 and n_2 vertices.

Theorem 3.3 If $p(n) = c/n$, for any $c > \ln 4/(\alpha\beta)$, then almost all bipartite graphs $G_p(n)$ have no induced $\overline{K}_{\alpha n,\beta n}$.

Proof. Let X be a random variable denoting the number of copies of induced $\overline{K}_{\alpha n,\beta n}$, and X_i be a $\{0,1\}$-random variable denoting whether a specific set of αn and βn vertices induces an $\overline{K}_{\alpha n,\beta n}$. Then

$$\begin{aligned} E(X) &= \sum_i E(X_i) = \binom{n}{\alpha n}\binom{n}{\beta n}(1-p(n))^{\alpha\beta n^2} \\ &\leq 2^n 2^n \left(\left(1-\frac{c}{n}\right)^n\right)^{\alpha\beta n} \leq \left(4e^{-c\alpha\beta}\right)^n \end{aligned}$$

Since $c > \ln 4/(\alpha\beta)$, we have $E(X) = 0$ as $n \to \infty$. □

Since the results of Theorem 3.1 and Theorem 3.3 are very close, we will follow the tradition of Kuo and Fuchs [11] and treat the problem as a bipartite vertex cover problem instead of an induced-subgraph problem in the remainder of the paper.

4 Early-Abort Heuristics

In this section, we examine several heuristics which are front-ended with some polynomial time algorithms to detect as many unrepairable instances as possible. We will show under what $p(n)$, these heuristics can effectively detect unrepairable instances.

4.1 Diagonal Test

This method was proposed by Bindels, et al. [2], by observing that any two faulty elements on the diagonal line of a memory cannot be repaired by using only one spare row or only one spare column. If the number of faulty elements on the diagonal line is greater than the total number of spare rows and columns, the memory is unrepairable.

Theorem 4.1 For a random graph $G_p(n)$, $p(n) = \alpha + \beta + \epsilon\ \omega(n)/\sqrt{n}$ is the threshold for $G_p(n)$ having $(\alpha+\beta)n$ edges in any specified set of n pairs of vertices, where $0 < \alpha+\beta < 1$ and $\omega(n) \to \infty$.

Proof. Let S be a set of designated n pairs of vertices, and X be a random variable denoting the number of pairs of S having an edge between them. Then X is in binomial distribution. $E(X) = np(n)$ and $Var(X) = np(n)(1-p(n))$. When $\epsilon > 0$, we show the total number of edges is almost always greater than $(\alpha+\beta)n$.

$$\begin{aligned} P\{X > (\alpha+\beta)n\} &\geq P\{(\alpha+\beta)n + 2\omega(n)\sqrt{n} > X > (\alpha+\beta)n\} \\ &= P\{|X - E(X)| < \omega(n)\sqrt{n}\} \\ &\geq 1 - \frac{Var(X)}{(\omega(n)\sqrt{n})^2} = 1 - O\left(\frac{1}{\omega^2(n)}\right) \\ &= 1 \quad as \quad n \to \infty \end{aligned}$$

When $\epsilon < 0$ we show the total number of edges is almost always less than or equal to $(\alpha+\beta)n$.

$$\begin{aligned} P\{X < (\alpha+\beta)n\} &\geq P\{(\alpha+\beta)n - \omega(n)\sqrt{n} < X < (\alpha+\beta)n\} \\ &= P\{|X - E(X)| < \omega(n)\sqrt{n}\} \\ &= 1 \quad as \quad n \to \infty \end{aligned}$$

□

If we choose the diagonal n elements as the set S in the theorem, then the theorem says the diagonal almost always contains more than $(\alpha+\beta)n$ faults if $\epsilon > 0$, and almost never contains more than $(\alpha+\beta)n$ faulty elements when $\epsilon < 0$. A natural extension of the diagonal test concept is to consider all sets $D(i) = \{((i+j)_{mod\ n}, j) \mid j = 1, 2, ..., n\}$, where $D(0)$ is the diagonal. This extension has only a small effect on the last term of the threshold function.

4.2 Maximum Matching

For a bipartite graph, the well known Egervary-König's Theorem says the size of the maximum matching equals the size of the minimum vertex cover [12]. If the size of the maximum matching is greater than $(\alpha+\beta)n$, then the array is unrepairable [11].

Theorem 4.2 If $p(n) = c/n$ where $c > \ln 4/(1-\alpha-\beta)^2$, then the size of maximum matching is at least $(\alpha+\beta)n$.

Proof. Given a bipartite graph $G = (V_1 \cup V_2, E)$, for any $W \subseteq V_1$, the *deficiency* of W, $\delta(W)$, is defined as $|W| - |R(W)|$, where $R(W)$ are W's neighbors in V_2. Formally, $R(W) = \{v_j \in V_2 \mid \exists v_i \in W, (v_i, v_j) \in E\}$. The *deficiency* of G is defined as $\delta(G) = \max \delta(W)$ for all $W \subseteq V_1$. The size of the maximum matching equals $n - \delta(G)$ [12].

We now show the size of the maximum matching cannot be smaller then $(\alpha+\beta)n$. Otherwise, there would be a subset of V_1 which has deficiency at least $(1-\alpha-\beta)n$. Let X be the number of such subsets and write $1-\alpha-\beta = \gamma$.

$$E(X) \leq \sum_{k=\gamma n}^{n} \binom{n}{k} \binom{n}{k-\gamma n} (1-p(n))^{k(n-k+\gamma n)}$$

$$\leq \; n2^n 2^n \left(1 - \frac{c}{n}\right)^{\gamma^2 n^2} = n4^n e^{-c\gamma^2 n} \to 0$$

A contradiction. □

Therefore, the edge probability for G_p having a maximum matching of size greater than αn is very close to the edge probability for G_p being unrepairable.

4.3 Total Faults

As previously stated, the maximum number of faulty elements that can be repaired by αn spare rows and βn spare columns is $N = (\alpha+\beta-\alpha\beta)n^2$. If a memory has more than N faults, it is unrepairable [4]. Using a similar argument to Theorem 4.1, we can prove the following theorem.

Theorem 4.3 For a random bipartite graph G_p, $p(n) = \alpha + \beta - \alpha\beta + \epsilon\ \omega(n)/n$ is the threshold function for G_p having more than $(\alpha + \beta - \alpha\beta)n^2$ edges, where $\omega(n) \to \infty$.

5 Partial Solution Heuristics

In this section, we study several heuristics which are front-ended with polynomial time algorithms to reduce the problem size by detecting as many mandatory repairs as possible. We show under what $p(n)$, these heuristics can reduce the problem size, and by how much.

5.1 Must-repair

A row (or column) which contains more than αn (or βn) faults must be replaced, since all the faults in the row (or column) cannot be repaired by using spare columns (or rows) only.

Theorem 5.1 The threshold function for G_p having some vertex with degree greater than αn is $p(n) = \alpha + \epsilon\omega(n)/n$, where $\omega(n) = \Omega(n^{9/14})$ and $\omega(n) \to \infty$.

Proof. Let X be a random variable denoting the number of vertices having degree greater than αn faults, X_i be a random variable denoting the degree of vertex v_i. Then $E(X) = \sum_{i=1}^{n} P\{X_i \geq \alpha n\} = nP\{X_i \geq \alpha n\}$. When $\epsilon > 0$, for every row, we have

$$\begin{aligned} P\{X_i > \alpha n\} &\geq P\{\alpha n + 2\omega(n) > X_i > \alpha n\} \\ &= P\{|X_i - E(X_i)| < \omega(n)\} \\ &\geq 1 - \frac{Var(X_i)}{\omega^2(n)n} = 1 - O(\frac{1}{\omega^2(n)}) \\ &= 1 \ \ as \ \ n \to \infty \end{aligned}$$

On the other hand, when $\epsilon < 0$,

$$\begin{aligned} E(X) &= nP\{X_i \geq \alpha n\} \\ &= n \sum_{k=\alpha n}^{n} \binom{n}{k} p^k(n)(1 - p(n))^{n-k} \end{aligned}$$

To estimate the tail of the binomial distribution, we need the DeMoivre-Laplace Limit Theorem [3]. It says if $npq \to \infty$, $0 < h = x\sqrt{pqn} = o\{(pqn)^{2/3}\}$ and $x \to \infty$, then

$$\sum_{k=pn+h}^{n} b(k; n, p) \sim \frac{1}{x\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Let $h = \omega(n)$ and $x = n^{1/7}$. We have $E(X) \to 0$. □

By applying Bollobás' proof of general graphs [3], we can show when $p(n) = c/n$, the maximum degree of almost every bipartite graph $G_p(n)$ is $\Theta(\ln n / \ln \ln n)$, which is much less than αn and βn.

5.2 Isolated Faults

When a fault does not share a column and row with any other faults, we call the fault an isolated fault. When a set F of faults share a common column (or row), but none of the faults in F shares any row (or column) with any other faults, we call the column (or row) a linear fault line. These concepts have been used in several heuristics, such as the one proposed by Hemmady and Reddy [10]. It is usually easy to decide a repair solution with isolated faults and linear fault lines.

Theorem 5.2 The threshold function for a bipartite graph G_p to contain an isolated edge is $p(n) = (0.5 + \epsilon) \ln n/n$.

Proof. Let X be a random variable denoting the number of isolated edges, X_i be a $\{0,1\}$-random variable denoting whether edge i is an isolated edge, then

$$\begin{aligned} E(X) &= \sum_i E(X_i) = n^2 p(n) (1 - p(n))^{2(n-1)} \\ &= n^2 \left(\frac{1}{2} + \epsilon\right) \frac{\ln n}{n} \left(1 - \left(\frac{1}{2} + \epsilon\right) \frac{\ln n}{n}\right)^{2(n-1)} \\ &= \left(\frac{1}{2} + \epsilon\right) \frac{n \ln n}{n^{1+2\epsilon}} \end{aligned}$$

When $\epsilon > 0$, $E(X) \to 0$. To study $E(X)$ when $\epsilon < 0$, we use the second moment method [3, 14]. The second moment method says that for a random variable X, $P\{X = 0\} \leq (E(X^2) - E(X)^2)/E(X)^2$. In particular, $E(X^2)/E(X)^2 \to 1$ implies $P\{X = 0\} \to 0$.

$$\begin{aligned} E(X^2) &= E((\sum_i X_i)^2) = E(\sum_i X_i^2) + E(\sum_{i \neq j} X_i X_j) \\ &= E(X) + \sum_{i \neq j} E(X_i X_j) \\ &= E(X) + n^2(n-1)^2 p^2(n)(1 - p(n))^{2(n-2)} \end{aligned}$$

Therefore, $E(X^2)/E(X)^2 \to 1$ as $E(X) \to \infty$. □

The single fault line corresponds to an isolated edge. The above theorem says such isolated edges appear around the failure rate $(0.5+\epsilon)\ln n/n$. The linear fault line corresponds to an isolated star (a star is a $K_{1,k}$ for some k). We now consider how many such isolated faults and linear faults there are when the failure rate is c/n.

Theorem 5.3 Let $p(n) = c/n$ for any $c > 0$. The number of isolated edges is $ce^{-2c}n$ and the percentage of isolated edges is e^{-2c}. The total number of edges in stars is $e^{-2c}(2e^{ce^{-c}} - 1)cn$ and the percentage of edges in stars is $e^{-2c}(2e^{ce^{-c}} - 1)$.

Proof. Let X be the number of isolated edges, then $E(X) = n^2p(n)(1-p(n))^{2(n-1)} = ce^{-2c}n$. Since the total number of edges is cn, the percentage of isolated edges is e^{-2c}.

Let Y_k be a random variable denoting the number of copies of $K_{1,k}$ with only one vertex in V_1, then

$$\begin{aligned} E(Y_k) &= n\binom{n}{k}p^k(n)(1-p(n))^{n-k}(1-p(n))^{k(n-1)} \\ &= n\left(\frac{(np)^k}{k!}e^{-np}\right)\left(1-\frac{c}{n}\right)^{k(n-1)} = n\frac{c^k}{k!}e^{-c}e^{-ck} \end{aligned}$$

Therefore, the total number of edges in stars is

$$\begin{aligned} 2\sum_{k=1}^{n} kE(Y_k) - E(X) &= 2ne^{-c}\sum_{k=1}^{n} k\frac{(ce^{-c})^k}{k!} - ce^{-2c}n \\ &= e^{-2c}(2e^{ce^{-c}} - 1)cn \end{aligned}$$

divide by the total number of faults which is cn and the theorem is proved. □

This theorem says that to consider isolated faults and linear fault lines we may decrease the problem size by a fixed percentage. When failure rate is $1/n$, for example, about 25% of faults are isolated or on a linear fault line.

5.3 Critical Set

The critical set heuristic by Hasan and Liu [9] is an enhanced version of maximum matching. Essentially the critical set is the intersection of all minimum vertex covers. When there are too many edges, the graph almost always contains a perfect matching. If the graph has a perfect matching, then the critical set is empty. On the other hand, when there are too few edges, these edges are almost all isolated edges in which case the critical set is again empty. When $p(n) = n^{-(3/2+\epsilon)}$ for any $\epsilon > 0$, we will show the expected number of vertices with degree greater than 1 goes to 0.

$$\begin{aligned} E(X) &= 2n \sum_{k=2}^{n} \binom{n}{k} p^k (1-p)^{n-k} \leq 2n \sum_{k=2}^{\infty} \frac{(pn)^k}{k!} \\ &= 2n(e^{pn} - 1 - pn) \end{aligned}$$

Using Hospital's rule several times, we have $E(X) \to 0$.

To conclude, the critical set method works when the failure rate is between $p(n) = \ln n/n$ where the graph has a perfect matching and $p(n) = n^{-(3/2+\epsilon)}$ where all edges are isolated.

5.4 Comparison of Heuristics

Results in Section 4 and Section 5 are summarized in Table 1. The first four heuristics are applicable only when the failure rate $p(n)$ is a constant. For any $p(n) \to 0$ as $n \to \infty$, these heuristics can neither detect any unrepairable instances, nor perform any mandatory repair. The isolated fault heuristic

Table 1: Probability of successful application of heuristics.

Heuristics	Range Applicable
diagonal test	$p(n) > \alpha + \beta$
total faults	$p(n) > \alpha + \beta - \alpha\beta$
must repair (row)	$p(n) > \alpha$
must repair (column)	$p(n) > \beta$
isolated faults	$0.5 \ln n/n > p(n)$
maximum matching	$\ln 4/n(1-\alpha-\beta)^2 > p(n)$
critical set	$\ln n/n > p(n) > n^{-3/2}$

is applicable when the failure rate is less than $0.5 \ln n/n$, but it can only decrease the problem size by a constant factor when the failure rate is c/n. Therefore, the average-case time complexities of these heuristics are exponential. If we consider the applications where entire rows or columns may also be defective, then row and column must repair would also be practically useful heuristics. The applicable ranges of maximum matching and critical set cover the case when $p(n) = c/n$.

6 An Average-Case Polynomial Time Algorithm

In this section, we present an algorithm for memory repair such that when $p(n) = c/n$ for any $c < 1$, the algorithm runs almost always in $O(n^2)$ time and for any $c < 1/2$, the algorithm runs in expected $O(n^2)$ time. In Figure 1, $p(n) = c/n$ for $c < 1$ covers a considerable amount of the "depend on fault distribution" region, especially when the number of spares is small.

Lemma 6.1 If $p(n) = c/n$ for any $c < 1$, then the expected number of vertices of G_p in tree components is $2n + O(1)$. The number of vertices in non-tree components is almost always less than $\log n$.

Proof. For general graphs when $p(n) = c/n$ where $c < 1$, Erdös and Rényi proved the expected number of vertices in tree components is $n + O(1)$ [3]. This implies that for a bipartite graph $G_p(n)$ with $p(n) = c/n$, where $c < 0.5$, the expected number of vertices in tree components is $2n + O(1)$. In the following we improve the bound to $c < 1$ for bipartite graphs.

A vertex in a non-tree component must be connected to a cycle of length $2k$ through a path of length m, for some $k \geq 1$ and $m \geq 0$. If we sum over the expected number of vertices in the cycles and the paths over all $n \geq k \geq 1$ and $2n \geq m \geq 0$, we have an upper bound on the expected number of vertices in non-tree components. There are at most n^{2k+m} ways to form a cycle of length $2k$ and a path of length m, and each way appears with probability $(c/n)^{2k+m}$. So the expected number of vertices in non-tree components is at most

$$\sum_{k=1}^{n} \sum_{m=0}^{2n} (2k+m)c^{2k+m}$$

Since when $c < 1$,

$$\sum_{k=0}^{\infty} \sum_{m=0}^{\infty} c^{2k+m} = \left(\sum_{k=0}^{\infty} c^{2k}\right)\left(\sum_{m=0}^{\infty} c^{m}\right) = \frac{1}{1-c^2}\frac{1}{1-c}$$

We have

$$\sum_{k=1}^{n} \sum_{m=0}^{2n} (2k+m)c^{2k+m} \leq \frac{c(1+3c)}{(1+c)^2(1-c)^3}$$

which is a constant. Therefore, the probability that the number of non-tree vertices is greater than $\log n$ goes to 0. □

Lemma 6.2 If a given bipartite graph $G = (V_1 \cup V_2, E)$ is a forest, then we can decide in $O(n^2)$ time whether there is a bipartite vertex cover of size (s, t).

Proof. The method is a dynamic programming algorithm [1]. For a tree T with root v, we define two sets $OPT^+(T)$ and $OPT^-(T)$. $OPT^+(T) = \{(n_1, m_1), (n_2, m_2), \ldots, (n_k, m_k)\}$ where $0 \leq n_1 < n_2 < \cdots < n_k \leq s$ and $t \geq m_1 > m_2 > \cdots > m_k \geq 0$. $OPT^+(T)$ contains all minimum bipartite vertex covers of size (i, j), for $0 \leq i \leq s$ and $0 \leq j \leq t$, with the root v in the vertex cover. Similarly, $OPT^-(T)$ contains all minimum bipartite vertex covers of size (i, j) with the root v not in the vertex cover.

The sets are constructed bottom up. For a vertex $v \in V_1$ with k sons $v_1, v_2, \ldots, v_k$, we first construct the sets $OPT^+(T_i)$ and $OPT^-(T_i)$ of the k subtrees $T_1, T_2, \ldots, T_k$. Define a Cartesian sum of two binary relations $R_1 \oplus R_2 = \{(n_1 + n_2, m_1 + m_2) | (n_1, m_1) \in R_1 \; and \; (n_2, m_2) \in R_2\}$. Then, we first compute:

$$S_1 = \bigoplus_{i=1}^{k} (OPT^+(T_i) \cup OPT^-(T_i))$$

After deleting redundant tuples from S_1, we have $OPT^+(T) = S_1 \oplus \{(1, 0)\}$. Similarly,

$$S_2 = \bigoplus_{i=1}^{k} OPT^+(T_i)$$

After deleting redundant tuples from S_2, we have $OPT^-(T) = S_2$.

To reduce the time complexity, the sets of $T_1, T_2, \ldots, T_k$ are merged two at a time: T_1 and T_2 are merged first, then merged with T_3, then $T_4, \ldots$, and finally T_k. Disjoint trees are merged in a similar fashion. Finally, the graph has a bipartite vertex cover of size (s, t) if and only if $OPT^+(G)$ or $OPT^-(G)$ contains element (n_i, m_i) such that $n_i \leq s$ and $m_i \leq t$.

Denote $C(T)$ as the time complexity for tree T, we will prove by induction on the size of T that $C(T) \leq cn^2$, where $|T| = n$ and c is some constant greater than 1. When $n = 1$, it is trivially true.

Let the subtrees of T be $T_1, T_2, \ldots, T_k, k \geq 1$, and we write $|T_i| = n_i$, then

$$\begin{aligned} C(T) &\leq n_1 n_2 + (n_1 + n_2)n_3 + \cdots + (n_1 + \cdots + n_{k-1})n_k + \textstyle\sum_{i=1}^{k} C(T_i) \\ &\leq n_1 n_2 + (n_1 + n_2)n_3 + \cdots + (n_1 + \cdots + n_{k-1})n_k + \textstyle\sum_{i=1}^{k} cn_i^2 \\ &= (cn_1)n_1 + (n_1 + cn_2)n_2 + \cdots + (n_1 + \cdots + n_{k-1} + cn_k)n_k \\ &\leq (n_1)cn_1 + (n_1 + n_2)cn_2 + \cdots + (n_1 + \cdots + n_k)cn_k \\ &\leq c(n_1 + \cdots + n_k)^2 \\ &= cn^2 \end{aligned}$$

Therefore, the total time is at most $O(n^2)$. Since the set is always of size $\min(s,t)$, a careful analysis shows the time complexity is $O(n \min(s,t))$. □

Theorem 6.3 When $p(n) = c/n$ for any $c < 1$, there is an algorithm which finds the bipartite vertex cover of size (s,t), and runs almost always in $O(n^2)$ time. When $c < 1/2$, the algorithm runs in expected $O(n^2)$ time.

Proof. The algorithm first finds all tree components of the given graph, which we call the forest. This can be done easily in $O(n)$ time [1]. The algorithm of Lemma 6.2 is then used to find solutions for the forest. Finally, we treat the non-tree components by enumerating all possibilities. Since the number of vertices in non-tree components is at most $\log n$, there will be at most $2^{\log n} = n$ different cases. The algorithm compares each of these different cases with the table of the forest to see if it is a solution. For each case, it takes $O(n)$ time to verify. The resulting total time complexity is $O(n^2)$.

When $c < 1/2$, the probability that a non-tree component is of size k is at most $n^k p(n)^k = c^k$. Therefore, we can afford $O(2^k)$ time for a non-tree component of size k since the expected time is $O(2^k)O(c^k) = O(1)$. The time for tree components is still $O(n^2)$ in the worst case. Therefore, the

total expected time is $O(n^2)$. □

The following is the highlight of the algorithm, which finds a bipartite vertex cover of size (s,t) for G if any, or indicates there is no solution. The operation "find $OPT(F)$" is described in the proof of Lemma 6.2.

Algorithm 1. Repair(G,s,t)
Input. G is a bipartite graph, s and t are two integers.
Output. "Yes" and the repair solution, or "no".
Method.
1: find all tree components of G: $T_1,\ldots,T_k$;
2: $F = \cup_{i=1}^{k} T_i$; $C = G - F$;
3: find $OPT(F)$;
4: **for** each bipartite vertex cover (U_1,U_2) of C
  if $OPT(F) + (|U_1|,|U_2|) \leq (s,t)$
  then return "yes" and solution;
5: **return** "no".
End of Algorithm.

The algorithm has been implemented and executed with previously published test input. In addition to the algorithm described in Theorem 6.3, we also included the row/column must repair heuristic to take care of faults covering entire rows and columns. Unlike many existing heuristics, our algorithm has no special case short cuts. Using the test data, our algorithm has a significant performance improvement over existing heuristics such as the branch and bound algorithm of Kuo and Fuchs [11], and the critical set algorithm of Hasan and Liu [9], see Table 2. The Kuo and Fuchs algorithm and our algorithm execution times are for a Sun 3/50. Hasan and Liu's algorithm is for a Pyramid.

Table 2: Sample performance of algorithms.

Array size	# of spare rows	# of spare columns	# of defective cells	repair-able	Kuo & Fuchs alg. (seconds)	Hasan & Liu alg. (seconds)	our alg. (seconds)
128 × 128	4	4	5	yes	0.12	0.11	0.04
128 × 128	4	4	15	no	0.14	0.08	0.06
256 × 256	5	5	10	yes	0.20	0.15	0.06
256 × 256	5	5	30	no	0.38	0.28	0.06
512 × 512	5	5	15	yes	0.28	0.13	0.44
512 × 512	10	10	19	yes	0.40	0.28	0.12
512 × 512	10	10	45	no	0.92	0.55	0.14
512 × 512	20	20	45	yes	1.32	1.18	0.40
1024 × 1024	20	20	40	yes	1.06	0.63	0.40
1024 × 1024	20	20	60	no	1.78	0.96	0.44
1024 × 1024	20	20	200	no	28.26	11.58	0.50
1024 × 1024	20	20	400	no	178.12	40.71	0.64

7 Conclusions

Random graph theory was used to examine the average-case time complexities of several heuristics for reconfigurable arrays as summarized in Table 1. Although the problem is worst-case NP-complete, we presented the first provably average-case polynomial time algorithm. The algorithm runs in asymptotically $O(n^2)$ time on average. We expect that the random graph analysis used in this paper can be applied to other reconfiguration heuristics to study average case time complexity and also in the development of efficient heuristics.

Acknowledgments. The authors thank Joel Spencer, who suggested the idea behind the proof of Lemma 6.1, and also Ming-Feng Chang and Douglas B. West for helpful discussions.

References

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

[2] J. F. M. Bindels, et al., "Cost-effective yield improvement in fault tolerant VLSI memory," *Proc. IEEE International Solid State Circuit Conference*, pp. 82–83, 1981.

[3] B. Bollobás, *Random Graphs*, Academic Press, London, 1985.

[4] R. C. Evans, "Testing repairable RAMs and mostly good memories," *Proc. International Testing Conference*, 1985, pp. 49–55.

[5] W. Feller, *An Introduction to Probability Theory and its Applications*, Vol. I, John Wiley and Sons, New York, 1966.

[6] W. K. Fuchs and M. F. Chang, "Diagnosis and repair of large memories: A critical review and recent results," *Proc. International Workshop on Defect and Fault Tolerance in VLSI Systems*, Plenum Press, New York, 1989, pp. 213–225.

[7] T. Fuji and C. Heegard, "Row/column replacement for the control of hard defects in semiconductor RAMs," *IEEE Trans. on Computers*, Vol. C-35, No. 11, Nov. 1986, pp. 996–1000.

[8] F. Harary, *Graph Theory*, Addison-Wesley, Reading Mass., 1969.

[9] N. Hasan and C. L. Liu, "Minimum fault coverage in reconfigurable arrays," *Proc. 18th International Symposium on Fault-Tolerant Computing*, 1988, pp. 348–353.

[10] V. G. Hemmady and S. M. Reddy, "On the repair of redundant RAMs," *Proc. 26th ACM/IEEE Design Automation Conference*, 1989, pp. 710–713.

[11] S. Y. Kuo and W. K. Fuchs, "Efficient spare allocation in reconfigurable arrays," *IEEE Design Test*, Vol 4, Feb. 1987, pp. 24–31.

[12] C. L. Liu, *Introduction to Combinatorial Mathematics*, McGraw Hill, New York, 1968.

[13] Y. N. Shen and F. Lombardi, "Yield enhancement and manufacturing throughput of redundant memories of repairability/unrepairability detection," *J. Electronic Testing: Theory and Applications*, Vol. 1, 1990, pp. 43–57.

[14] J. Spencer, *Ten Lectures on the Probabilistic Method*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, 1987.

[15] C. L. Wey and F. Lombardi, On the repair of redundant RAMs, *IEEE Trans. CAD*, Vol. 6, No. 2, 1987, 222–231.