Computing the Probability for Data Loss in Two-dimensional Parity RAIDs

Lars Nagel
Zentrum für Datenverarbeitung
Johannes Gutenberg-Universität
Mainz, Germany
Email: nagell@uni-mainz.de

Tim Süß
Zentrum für Datenverarbeitung
Johannes-Gutenberg-Universität
Mainz, Germany
Email: suesst@uni-mainz.de

***Abstract*—Parity RAIDs are used to protect storage systems against disk failures. The idea is to add redundancy to the system by storing the parity of subsets of disks on extra parity disks. A simple two-dimensional scheme is the one in which the data disks are arranged in a rectangular grid, and every row and column is extended by one disk which stores the parity of it.**

In this paper we describe several two-dimensional parity RAIDs and analyse, for each of them, the probability for data loss given that f random disks fail. This probability can be used to determine the overall probability using the model of Hafner and Rao. We reduce subsets of the forest counting problem to the different cases and show that the generalised problem is #P-hard. Further we adapt an exact algorithm by Stones for some of the problems whose worst-case runtime is exponential, but which is very efficient for small fixed f and thus sufficient for all real-world applications.

***Index Terms*—RAID, complexity, data loss, graph, Tutte polynomial, counting problem**

I. Introduction

Data storage is one of the corner stones of information technology, and the amount of data stored on tapes and disks grows at an impressive rate [1]. Since more and more data are stored on disks, there is ongoing research on how they can be efficiently secured using RAID schemes and erasure codes, *e.g.* [2]–[10]. The idea is to prevent data loss due to failing disks by adding redundancy and by automatically restoring the data of failed disks on new ones (*forward error correction*).

This paper considers two-dimensional RAID schemes with data and parity disks. The most common scheme arranges the disks in a two-dimensional grid such that each row and each column forms a parity set together with an extra parity disk; *i.e.*, the XORed contents of the data disks are stored on the parity disk. This allows to restore any disk of the parity set by XORing the contents of the other disks. In the remainder of this section, we will describe this *standard scheme* and variations of it, explain in which situations data loss occurs and phrase the specific problems.

Generally, we are interested in the following problem: What is the probability for data loss given that a fixed number of disks, randomly chosen, fails? The solution to this problem is a requirement for computing the overall data loss probability using, for example, the model by Hafner and Rao [11]. This model has been applied in the past [2], [4], [5] to solve the problem for specific RAID schemes, but only for a limited number of disk failures.

Our contributions are:

- We formally describe the known two-dimensional schemes and define the new *general scheme* as the superset of two of them (Section II).
- For each parity scheme, we define the problem of computing the probability for data loss given that f random disks fail and describe how the general probability can be computed based thereon (Section III).
- We show that computing the data loss probability for the general scheme is #P-hard and that, for the other schemes considered, the problem is (at least) as hard as evaluating the Tutte polynomial $T(G; x, y)$ for complete bipartite graphs G at $(2, 1)$ (Section IV-B).
- We describe exact algorithms for two types of the schemes which use the algorithm of Stones [12] as a subroutine. Although their worst-case runtime is exponential, they are efficient for a fixed number of disk failures (Section IV-C).
- Finally, we demonstrate that having a lower data loss probability in case of f failed disks does not imply that the probability of the RAID is also lower for $f' > f$ (Section IV-C).

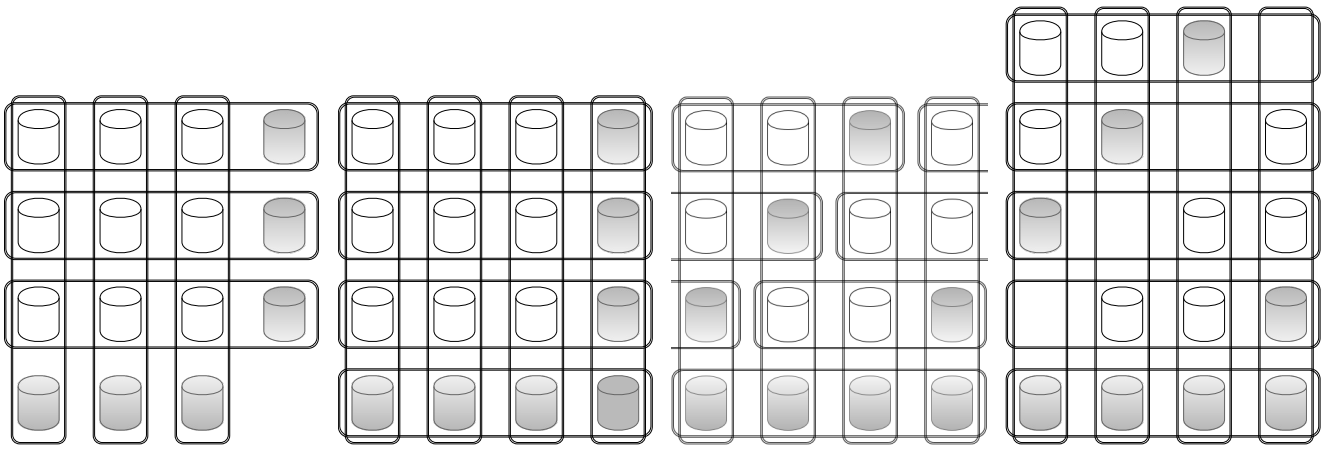
The related work is discussed in Section V before we conclude the paper in Section VI.

II. Parity Schemes

The parity schemes that we consider can each be arranged in a two-dimensional grid. They are obviously a special case of k-dimensional parity schemes, $k \in \mathbb{N}$, which would arrange data and parity disks in a k-dimensional grid. *RAID 4* [7] is an example for $k = 1$, the schemes in Figure 1 are examples for $k = 2$.

This paper investigates only schemes which keep the parity information on extra disks (like in RAID 4). Schemes in which the parity information is distributed over the data disks (like RAID 5 and RAID 6 implementations [7], [9], [10]), are not considered.

Standard scheme: The common two-dimensional scheme arranges the data disks in a $(n_1 - 1) \times (n_2 - 1)$ grid; $n_1 \geq 2$, $n_2 \geq 2$. Additionally every row and every column has a



(a) Standard scheme (b) Full scheme (c) Lonestar scheme (d) Lonestar scheme stretched

Fig. 1. Two-dimensional parity schemes. The shaded disks are the parity disks; row parity disks are shaded at the top, column parity disks at the bottom, superparity disks completely.

parity disk which is in the n_2-th column or n_1-th row, respectively; the corner (n_1, n_2) is left empty (see Figure 1(a)). For $i_1 \in \{1, ..., n_1 - 1\}$, each parity disk $d_{i_1,n_2} = \bigotimes_{i_2=1}^{n_2-1} d_{i_1,i_2}$ contains the bitwise XORed contents of the row's first $n_2 - 1$ disks. Likewise, for $i_2 \in \{1, ..., n_2-1\}$, $d_{n_1,i_2} = \bigotimes_{i_1=1}^{n_1-1} d_{i_1,i_2}$ contains the XORed contents of the column disks. Together with their parity disk, each row and each column forms a parity set: If one disk fails in this set, but all other disks remain intact, the data of the faulty disk can be restored on a new disk (which then replaces the faulty one). We call this scheme the *standard scheme*. When data are written to a data disk d_{i_1,i_2}, its parity disks, d_{i_1,n_2} and d_{n_1,i_2}, must be updated, too. The standard scheme can correct any two errors.

Full scheme: A simple way of making the standard scheme 3-error correcting is adding a corner disk as a *superparity disk* (Figure 1(b)). It stores the parity of the parity disks in the last row and thus also the parity of the parity disks of the the last column which is identical. Whenever data are written to a disk, it is now necessary to update the corner disk as well. We call this scheme, which is described in [4], the *full scheme*.

Lonestar scheme: A generalisation of the full scheme is the scheme that was developed for the *Lonestar* archival system [2], [3]. Here the horizontal parity groups do not have to fill one row exactly, but can be shorter. The next group is simply added behind the previous and wrapped into the next row if it does not fit into the current one. The last row consists of disks which each hold the parity of their column. For simplicity, we demand that the first $n_1 - 1$ rows are completely filled by the groups and that the group size y does not divide n_2 unless $y = n_2$. Note that the latter condition is no restriction because every scheme with $y \mid n_2$ actually consists of n_2/y separate full schemes. In case $y = n_2$, the *Lonestar scheme* is a full scheme.

It is easy to prove that the last row also forms a parity set if $y \nmid n_2$. An example with group length $y = 3$ and row length $n_2 = 4$ is given in Figure 1(c).

General scheme: For the theoretical analysis we introduce the *general scheme* which generalises the full scheme as well as the Lonestar scheme (if every horizontal parity set gets its own row as in Figure 1(d)). The general scheme is based on an $n_1 \times n_2$ grid, but there does not have to be a disk at every grid point. The only requirements are:

- every row / column forms a parity set (in between write operations), and
- the set of disks is 2-connected

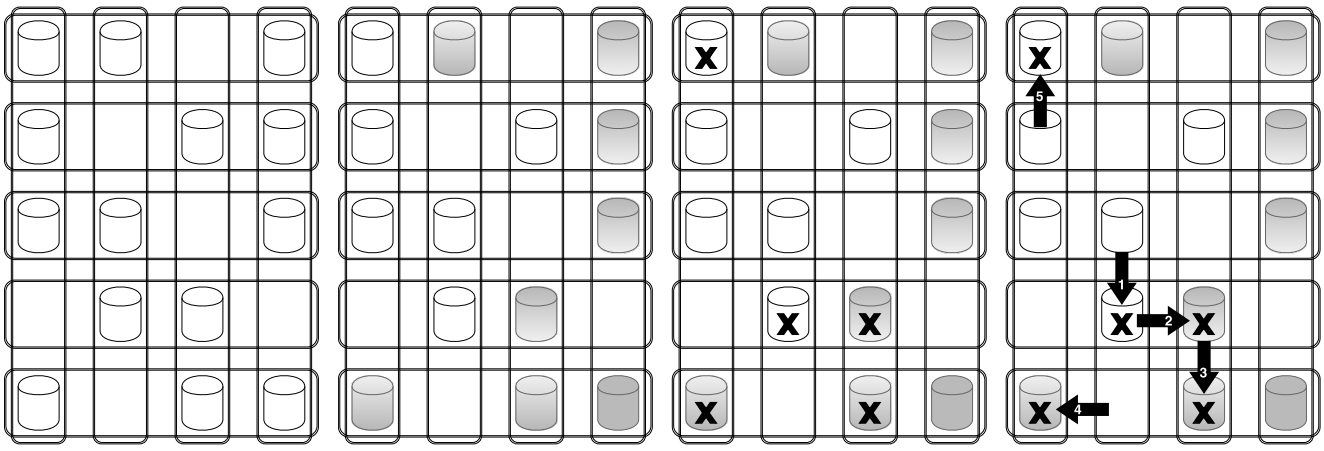
where 2-connected is defined as follows: Let a *cycle* be a sequence $d_1, d_2, ..., d_\ell$ of disks such that a pair of disks is in the same parity set if and only if the disks are successive (*i.e.*, d_i and d_{i+1} for $i \in \{1, ..., \ell - 1\}$) or the first and the last disk of the sequence (*i.e.*, d_1 and d_ℓ). Then a set of disks is 2-*connected* if and only if every pair of disks shares a cycle. For an example refer to Figure 2(c), in which the disks marked with an **x** form a cycle together with disk $d_{1,2}$, *i.e.* the second disk in the first row. All disks, marked or unmarked, in this figure are 2-connected. (We use the terms cycle and 2-connectedness because of their equivalence to the ones in graph theory that we will use later.)

One can easily prove that every instance of the full and the Lonestar scheme fulfils the two requirements above. An instance of the standard scheme, on the other hand, is never of the general scheme: The last row and column do not form a parity set, and this also implies that the RAID is not 2-connected.

In the following lemma we show that every grid which fulfils these two conditions can indeed be used as a parity RAID scheme.

Observation 1. *If a set of disks on an $n_1 \times n_2$ grid, in which every row and column forms a parity set, is 2-connected, it can be used as a parity RAID, where $n_1 + n_2 - 1$ disks are parity disks.*

Proof. We show that the set of disks can be used for storing and securing data by the following construction: First choose any disk d_{i_1,i_2} as the superparity disk and make every row



(a) General scheme (b) Parity RAID (c) Failed disks (d) Restoring data

Fig. 2. (a) Example general scheme on a 5×4 grid. (b) $5 + 4 - 1$ disks are turned into parity disks. (c) Example pattern of failed disks. (d) One way of restoring the data. (The shaded disks are the parity disks; row parity disks are shaded at the top, column parity disks at the bottom, superparity disks completely.)

disk $d_{i_1,c}$ the parity disk of their respective column c and every column disk d_{r,i_2} the parity disk of their respective row r. In the next step go through all these columns c and rows r and turn every disk into a parity disk of their respective row or column, unless there is already a parity disk in it. Repeat this step for the resulting set of parity disks, until there are no more disks that can be turned into parity disks.

Note that, by this construction, every row (column) has at most one row (column) parity disk and that no parity disk but the superparity disk is the parity disk for its row and column at the same time. It remains to show that every row / column has a parity disk after the construction. Assume some column c does not have a parity disk. Then all rows r which contain a disk $d_{r,c}$ have no parity disks, otherwise $d_{r,c}$ would have also been chosen. Since the RAID is 2-connected, there is a path to a superparity disk, and this recursive argument will result in the contradiction that there is no parity disk in the row or column of the superparity disk.

Due to symmetry, we can apply the same argument to rows.

When data are written to a data disk d_{i_1,i_2}, the parity disks $d_{i_1,c}$ and d_{r,i_2} in the same column and row are updated, followed by the updates of the parity disks in column c and row r, and so on, until the superparity disk is reached. (In the full scheme, the second step already updates the superparity disk, but in the general scheme a few more iterations might be necessary before the superparity disk is reached.) The disks involved form a cycle and, thus, in every row and every column zero or two disks are changed. Since the same operation is performed on all these disks, the XORed contents of each row and each column will not change and remain zero. □

Note that, in between write operations, every disk $d_{i,j}$ contains the parity of the other disks in its row and the parity of the other disks in its column. Due to this symmetry, a parity or superparity disk is, at least for our purposes, in no way exceptional.

An example for constructing a parity RAID on top of a general scheme is given in Figure 2(a) and 2(b). We use the algorithm provided in the proof of Lemma 1 and arbitrarily select the corner disk $d_{5,4}$ as the superparity disk. Then all disks in its row and column are turned into parity disks of their column or row, respectively. In the next step the first column having a parity disk itself, *i.e.* column 1, is checked whether any of the disks can be turned into a row parity disk. None of them can. In the next such column, column 3, $d_{4,3}$ is turned into a parity disk of row 4. The search for row parity disks can be stopped then because every row has one. Finally, for finding column parity disks the same procedure is used for the rows that have a parity disk themselves. In the first row $d_{1,2}$ is turned into a parity disk for column 2, and then the process stops completely because all columns have a parity disk as well.

III. Problem Statement

Data loss occurs if a failed disk d cannot be restored. This happens only if, in all parity sets of d, there is at least one other disk that cannot be restored. This recursive argument leads to the conclusion that there must be a set of failed disks such that each such disk has at least one other failed disk in each of its parity sets. We call such a set of disks a *fatal pattern*.

The smallest fatal pattern in a two-dimensional general scheme is a subset of four disks which form the corners of a rectangle – as we can easily show: Assume any disk d_{i_1,i_2} fails. Then its data are only lost if some other disk $d_{i_1,c}$ in its row and some other disk d_{r,i_2} in its column fail and if these two disks cannot be recovered themselves. This last condition implies that at least a fourth disk must fail to cause data loss. (As we will see in the proof of Lemma 2, this observation can actually be generalised to the statement that data loss happens if and only if the failed disks form a pattern containing at least one cycle.)

Figure 2(c) and 2(d) show an example failure pattern for a general scheme that is not fatal and one way how the data can be restored: Since there is only one failed disk in column 2, namely $d_{4,2}$, it can immediately be restored. Afterwards there is only one failure in row 4 ($d_{4,3}$); and after correcting that, there is only one failed disk in column 3 ($d_{5,3}$), which can also be restored. Finally, the chain of restorations is continued and concluded with $d_{5,1}$ and $d_{1,1}$. Note that the pattern would be fatal if $d_{1,2}$ were included.

Since the full scheme and the Lonestar scheme are special cases of the general scheme, it follows that up to 3 failures can always be corrected in these schemes as well. The standard scheme, on the other hand, allows fatal patterns of three disks, namely all sets $\{d_{i_1,i_2}, d_{n_1,i_2}, d_{i_1,n_2}\}$ [4].

But, as one can easily see, more disk failures (than three or four) do not necessarily imply data loss. If the set of failed disks does not contain a fatal pattern, all data can be restored. In the following we will investigate the probability for data loss in case f disks fail; in particular we will consider the following four problems:

Problem: DATALOSSPROBSTD
Input: Standard scheme on $n_1 \cdot n_2 - 1$ disks including $n_1 + n_2 - 2$ parity disks; positive integer f.
Output: The probability for data loss if f randomly chosen disks fail.

Problem: DATALOSSPROBFULL
Input: Full scheme on $n_1 \cdot n_2$ disks including $n_1 + n_2 - 1$ parity disks; positive integer f.
Output: The probability for data loss if f randomly chosen disks fail.

Problem: DATALOSSPROBLS
Input: Lonestar scheme on $n_1 \cdot n_2$ disks including $(n_1 - 1) \cdot n_2/y + n_2$ parity disks; positive integer f.
Output: The probability for data loss if f randomly chosen disks fail.

Problem: DATALOSSPROBGEN
Input: A general scheme; positive integer f.
Output: The probability for data loss if f randomly chosen disks fail.

DATALOSSPROBLS, DATALOSSPROBSTD and DATALOSSPROBFULL were analysed in [2] and [4]. Using the *reliability model* introduced in [11], these papers compute the *mean time to data loss* (MTTDL) of a data storage system which – as the name suggests – is the expected time until disk failures lead to (irreparable) data loss. But the analysis in [2] considers only a small number of scenarios with a fixed number of disks, and both studies limit the number of disk failures to 5. We will show in the next section that the generalised problem, while easy for small numbers, is in fact #P-hard for an arbitrary number of disk failures and provide an exponential-time algorithm for DATALOSSPROBSTD and DATALOSSPROBFULL.

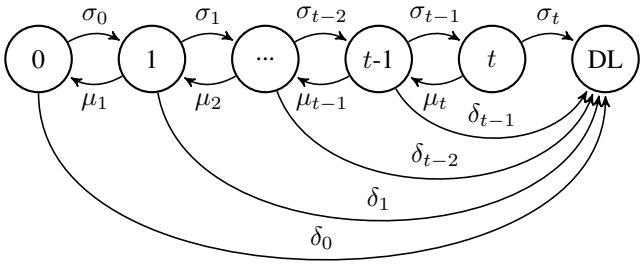


Fig. 3. Reliability model (picture taken from [2], [11]).

The reliability model [11] is generally meant for erasure codes for which the Hamming distance d of the code does not strictly separate the reparable from the irreparable error patterns. In contrast, assume a code that can recover from any number of errors less than d, but from no error pattern with at least d failures. Then computing the error probabilty for any number f of failed disks would be trivial. Codes having this property are equated with *maximum distance separable* (MDS) in [11] (which is not completely accurate because the separation is not necessarily strict for MDS codes), and thus the model addresses non-MDS codes which include the RAID schemes discusssed in this paper, but also many other codes (*e.g.* [5]). Examples for MDS codes are the *row-diagonal parity code* [9] and the *EVENODD code* [10].

As shown in Figure 3, the reliability model is based on a simple Markov model consisting of $t+2$ states and transition probabilities σ_i, μ_i and δ_i where t is the maximum size of any non-fatal error pattern; *i.e.*, any $t+1$ errors will definitely cause data loss. The states i, $i \in \{1, ..., t\}$, merge all non-fatal system states with i failures, the state DL all states with definite data loss. In this paper we do not focus on the probability for another disk to fail, but on the probability whether the system is in state f or in state DL given that f failures occur. Together with the probabilities for disk failures, one can determine the transition probabilities of the Markov model. For the computation of the MTTDL, see [11].

IV. ANALYSIS

In this section we provide properties, complexity results and algorithms for the DATALOSSPROB problems defined in the previous section. For this we make use of the equivalence subsequently described which links bipartite graphs to parity RAIDs.

A. Equivalences

Every bipartite graph $G = (A, B, E)$ can be expressed as a (reduced) adjacency matrix M in which the $|A|$ rows represent the vertices of A and the $|B|$ columns the vertices of B. Let a_i be the vertex of row i and b_j be the vertex of column j. If $(a_i, b_j) \in E$, then $M_{i,j} = 1$, otherwise $M_{i,j} = 0$. The same matrix also describes the grid of a two-dimensional parity RAID. If there is a disk at the grid position (i, j), then $M_{i,j} = 1$; otherwise $M_{i,j} = 0$.

Let the parity RAID described by a matrix be of the general form in that every row and every columns forms a parity set. Then, if a bipartite graph contains cycles, the respective parity RAID (*i.e.*, described by the same matrix) also contains cycles. One can easily prove that the same bits inducing a cycle in one also induce a cycle in the other. From this it also follows that a bipartite graph is 2-connected if and only if the parity RAID is 2-connected.

The following lemma expresses the probability for data loss via graph properties, more precisely, as a function of $\phi(G, f)$ which denotes the number of cycle-free subgraphs (subforests) of the respective graph G with f edges.

Lemma 2. *Let $G = (A, B, E)$ be a bipartite graph such that every vertex in G has degree at least 2, and let P denote the equivalent parity RAID whose rows and columns form parity sets. Further let $\mathcal{D}_P$ denote the event of data loss in P. Then,*

$$Pr[\mathcal{D}_P \mid f \text{ disks fail}] = 1 - \frac{\phi(G, f)}{\binom{|E|}{f}}.$$

Proof. Consider the reduced adjacency matrix of G and the parity RAID it describes. If f disks fail and if the matrix entries of all healthy disks are set to 0, then the resulting matrix describes a subgraph of G with f edges. And since every subgraph also describes a pattern, this mapping is a simple bijection between the set of subgraphs with f edges and the set of patterns with f disks. Both sets have therefore size $\binom{|E|}{f}$.

In order to prove the statement, we additionally show that a subgraph of G contains a cycle if and only if the respective pattern causes data loss in the parity RAID. Assume that an acyclic graph (*i.e.* a forest) T maps to a fatal pattern and that T is minimal in the sense that no proper subgraph of it maps to a fatal pattern. Note that, since every vertex in G has degree at least 2, each parity set of the respective RAID contains at least two disks. Since T is a forest with at least one edge, there must be at least one vertex having degree 1 and thus one parity set with only one faulty disk. Hence, this disk can be repaired, which is a contradiction to the minimality of the subgraph. For the other direction, consider a minimal fatal pattern. Due to the minimality, every row and every column of this pattern must contain at least two failed disks because otherwise it would be intact or could be repaired. So each vertex of the resulting subgraph has degree at least 2 for which reason the subgraph cannot be a forest and must contain a cycle.

Due to the equivalences shown, the probability for data loss equals the number of cyclic subgraphs on f edges divided by the number of all graphs on f edges:

$$Pr[\mathcal{D}_P \mid f \text{ disks fail}] = \frac{\binom{|E|}{f} - \phi(G, f)}{\binom{|E|}{f}}.$$

□

B. Complexity

In this section we will provide reductions of graph problems to the data loss probability problems. Here, $\leq_T^p$ denotes the polynomial Turing reduction, $\leq_1^p$ and $\equiv_1^p$ the polynomial one-one reduction and equivalence. We assume that all graphs are labelled and simple; *i.e.* they are undirected and do not have loops or multiple edges. A *spanning forest* of a graph is a cycle-free spanning subgraph.

The graph problems are:

Problem: FORESTCOUNT$[\mathcal{G}]$
Input: Graph $G \in \mathcal{G}$.
Output: Number of spanning forests of G.

Problem: FORESTCOUNTE$[\mathcal{G}]$
Input: Graph $G \in \mathcal{G}$, positive integer f.
Output: Number of spanning forests of G with exactly f edges.

Based on these definitions, we can phrase the following observation:

Observation 3. *Let $\mathcal{G}$ be any graph class, then:*

$$\text{FORESTCOUNT}[\mathcal{G}] \leq_T^p \text{FORESTCOUNTE}[\mathcal{G}].$$

Proof. FORESTCOUNT$[\mathcal{G}] \leq_T^p$ FORESTCOUNTE$[\mathcal{G}]$ holds because the number of forests can be determined by summing up the number of forests with $f \geq 0$ edges. □

We consider FORESTCOUNTE$[\mathcal{G}]$ and FORESTCOUNT$[\mathcal{G}]$ for the subsequently defined graph classes. As formally described in Lemma 5, the graph classes $\mathcal{G}_{bv}$ and $\mathcal{G}_{cb}$ are equivalent to the data loss problems of the general and the full scheme, respectively.

- $\mathcal{G}_b = \{G \mid G \text{ bipartite}\}$
- $\mathcal{G}_{bv} = \{G \mid G \text{ bipartite and 2-connected}\}$
- $\mathcal{G}_{cb} = \{G \mid G \text{ complete bipartite and 2-connected}\}$

Our next observation provides an equation which sets the data loss probability of the standard and the full scheme in relation. It follows from the observation that the standard scheme equals the full scheme with a broken corner disk.

Observation 4. *Let $\mathcal{D}^{full}_{n_1,n_2}$ and $\mathcal{D}^{std}_{n_1,n_2}$ respectively denote the event of data loss in the full and standard scheme on an $n_1 \times n_2$ grid. Then*

$$Pr[\mathcal{D}^{std}_{n_1,n_2} \mid f \text{ disks fail}] = Pr[\mathcal{D}^{full}_{n_1,n_2} \mid f+1 \text{ disks fail}].$$

The next lemma reduces different flavours of FORESTCOUNTE to the DATALOSSPROB problems:

Lemma 5. *Let $\mathcal{G}_{bv}$ and $\mathcal{G}_{cb}$ be the graph classes defined above. Then:*

(i) FORESTCOUNTE$[\mathcal{G}_{bv}] \equiv_1^p$ DATALOSSPROBGEN
(ii) FORESTCOUNTE$[\mathcal{G}_{cb}] \equiv_1^p$ DATALOSSPROBFULL
(iii) FORESTCOUNTE$[\mathcal{G}_{cb}] \equiv_1^p$ DATALOSSPROBSTD
(iv) FORESTCOUNTE$[\mathcal{G}_{cb}] \leq_1^p$ DATALOSSPROBLS

Proof. We use that every $n_1 \times n_2$ bit matrix uniquely describes a bipartite graph as well as a RAID scheme in which every row and column forms a parity set and that data loss occurs exactly if the graph contains a cycle. Hence, for (i) and (ii), we only need to show for each of the graph classes that the

underlying reduced adjacency matrices describe excactly the RAID schemes of the respective data loss probability problem. For (i) this follows from the fact that a bipartite graph is 2-connected if and only if the respective RAID scheme is 2-connected. For (ii) it is sufficient to observe that every full scheme is described by an $n_1 \times n_2$ bit matrix in which every bit is set to 1 and for which $n_1 \geq 2$ and $n_2 \geq 2$. These matrices are exactly the reduced adjacency matrices of the complete bipartite, 2-connected graphs. Statement (iii) follows from (ii) and Observation 4. Statement (iv) follows from the fact that the full scheme is a special case of the Lonestar scheme. □

Due to $\mathcal{G}_{cb} \subset \mathcal{G}_{bv}$, this lemma immediately implies:

Corollary 6. *If* FORESTCOUNT$[\mathcal{G}_{cb}]$ *is #P-hard, then all data loss probability problems described in this paper are #P-hard.*

To the best of our knowledge there is no precise classification of FORESTCOUNT$[\mathcal{G}_{cb}]$. (We discuss work related to FORESTCOUNT in Subsection V.)

But nonetheless we can classify DATALOSSPROBGEN. FORESTCOUNT equals the evaluation of the Tutte polynomial $T(G; x, y)$ at $(2, 1)$ (*e.g.* [12], [13]). Jaeger et al. showed in [13] that evaluating $T(G; x, y)$ at (x, y) is #P-hard unless (x, y) is in a certain set which they define in their theorem and which does not contain $(2, 1)$. This result was extended to (planar) bipartite graphs by Vertigan and Welsh:

Theorem 7 (Vertigan et al. [14])**.** *The evaluation of the Tutte polynomial $T(G; x, y)$ of planar bipartite graphs G at a point (a, b) is #P-hard except when $(a, b) \in \{(1, 1), (-1, -1), (j, j^2), (j^2, j) \mid j = e^{2\pi i/3}\} \cup \{(x, y) \mid (x - 1)(y - 1) = 1 \vee (x - 1)(y - 1) = 2\}$.*

Corollary 8. FORESTCOUNT$[\mathcal{G}_b]$ *is #P-hard.*

Proof. As mentioned, $T(G; 2, 1)$ is the number of (spanning) forests of G, and from the theorem it follows that evaluating $T(G; 2, 1)$ remains hard for the class of bipartite graphs G. □

Lemma 9. FORESTCOUNT$[\mathcal{G}_{bv}]$ *is #P-hard.*

Proof. We show that FORESTCOUNT$[\mathcal{G}_b]$ is reducible to FORESTCOUNT$[\mathcal{G}_{bv}]$. Note that isolated vertices in the base graph can be ignored because spanning forests only differ in the number and distribution of the edges.

Consider the subset E_n of edges that are not contained in any cycle. Computing the number of forests for a connected component of $G' = (V, E \setminus E_n)$ is a problem in FORESTCOUNT$[\mathcal{G}_{bv}]$. Computing the number of forests for $G_n = (V, E_n)$ can be done in polynomial time because G_n is a forest. The number of forests of the connected components and G_n are independent. Therefore the number of forests of G is the product of the counts, and FORESTCOUNT$[\mathcal{G}_b]$ can be reduced to FORESTCOUNT$[\mathcal{G}_{bv}]$. □

From Lemma 5 and Lemma 9 we can finally derive:

Theorem 10. DATALOSSPROBGEN *is #P-hard.*

C. Algorithm

Although FORESTCOUNTE$[\mathcal{G}_{cb}]$ is not yet precisely classified, there Stones [12] developed an exponential algorithm for this problem and gave a rough bound on the worst case runtime:

Theorem 11 (by Stones [12])**.** *Let $n = \max\{n_1, n_2\}$, then* FORESTCOUNTE$[\mathcal{G}_{cb}]$ *can be computed in time $O(e^{const \cdot \sqrt{n}} \cdot n^{0.5 \cdot \sqrt{n} + const})$. For a fixed number f of edges, the runtime is $O(\log(n_1 \cdot n_2))$.*

This algorithm can be used to compute DATALOSSPROBFULL and DATALOSSPROBSTD:

Corollary 12. *The problems* DATALOSSPROBFULL *and* DATALOSSPROBSTD *can be computed in time $O(e^{const \cdot \sqrt{n}} \cdot m^{0.5 \cdot \sqrt{n} + const})$. For a fixed number of failures, the runtime is $O(\log(n_1 \cdot n_2))$.*

Proof. The algorithm of Stones computes the number $\phi(G, f)$ of subforests of the complete bipartite graph G with f edges. Now, to compute the solution to a DATALOSSPROBFULL problem with f failed disks, one only has to interpret the binary matrix representation of the full scheme instance as the reduced adjacency matrix of a complete bipartite graph G, compute $\phi(G, f)$ and apply the formula of Lemma 2. Similarly one can solve DATALOSSPROBSTD. Given any grid and any f, one obtains the data loss probability for the standard scheme by computing the probability for the full scheme for $f+1$ (see Observation 4). Hence, one can again apply the algorithm of Stones to get the result. □

In order to compare parity RAIDs with each other, it is not necessary to calculate the exact probability for data loss, but it would be sufficient to show that one RAID is better than the other. This raises the question whether some RAID A is better than another RAID B for all numbers of errors f if it is better for $f = 4$. Unfortunately, this is not the case.

Claim 13. *If some parity RAID A of the full (standard, Lonestar, general) scheme is better than some parity RAID B of the full (standard, Lonestar, general) scheme for $f = 4$, then it is also better for $f = 5$. But this does not hold for arbitrary m, n, f and $f + 1$.*

Proof. We show the statements for the full scheme. Since the Lonestar and the general scheme are generalisations of the full scheme and since the probability for the standard scheme can be derived from the probability of the full scheme, the statements must also hold for them.

There is exactly one fatal pattern of four disks, namely the one in which the failed disks form the corners of a rectangle. Yet, there is no fatal pattern of five disks because one of the five failed disks must be the only one in its row or its column such that its data can be restored. The probability for $f = 4$ is the number of rectangles divided by the number of all possibilities:

$$Pr[\mathcal{D}^{full}_{m,n} \mid 4 \text{ fail}] \quad = \quad \frac{\binom{m}{2} \cdot \binom{n}{2}}{\binom{m \cdot n}{4}}.$$

For $f = 5$, it is the number of rectangles times the possibilities to add another disk divided by all possibilities:

$$\begin{aligned} Pr[\mathcal{D}^{full}_{m,n} \mid 5 \text{ fail}] &= \frac{\binom{m}{2} \cdot \binom{n}{2} \cdot (m \cdot n - 4)}{\binom{m \cdot n}{5}} \\ &= \frac{\binom{m}{2} \cdot \binom{n}{2} \cdot 5}{\binom{m \cdot n}{4}} \\ &= 5 \cdot Pr[\mathcal{D}^{full}_{m,n} \mid 4 \text{ fail}]. \end{aligned}$$

This equality expressing that the probability grows by a constant factor and thus independent of m, n and f, implies the first statement.

The second statement can be shown by presenting examples. The RAID with $m = 2$ and $n = 7$, for instance, has a greater probability for $f = 5$ than the RAID with $m = 4$ and $n = 4$, but a smaller one for $f = 6$. (This was calculated using the algorithm given in the proof of Corollary 12.) □

V. Related Work

In this section we give an overview of the work related to the description and analysis of two-dimensional RAID schemes as well as to the Tutte polynomial and the forest counting problem.

A. Two-dimensional RAID Schemes

Except for the general scheme, the parity schemes are taken from the literature. The standard and the full scheme are well-known "folklore", and it is not clear when they were described for the first time. A description and analysis of both schemes can be found in a paper by Pâris *et al.* [4]. The authors point out that the standard scheme can tolerate only two failed disks in the worst case and suggest the full scheme which tolerates at least three failed disks. For their analysis of the data loss probability, they use the model by Hafner and Rao [11], but the computation is only accurate up to five failed disks.

An improved version of the standard scheme is described in [6]. The authors suggest to make the $n \times n$ standard scheme adaptive: Whenever a first parity disk fails, the parity sets are rearranged in such a way that each data disk is again part of two parity sets. The authors argue that the reorganisation involves only n out of n^2 disks and doubles the mean time to data loss. Further reorganisations are not performed.

The Lonestar scheme was first described by Grawinkel *et al.* [3] who use it as the basis for a disk-based archival system. They also use the model by Hafner and Rao to determine the data loss probability, and their analysis is also inaccurate for more than five disk failures. The description and analysis of the Lonestar archival system was recently extended in [2].

B. Tutte Polynomial and Forest Counting

Aside from the papers already mentioned [12]–[14], there is a multitude of publications on Tutte polynomials and forest counting of which we only cite the most relevant ones. For a more extensive overview, see the references in the papers named in this subsection.

An exponential-time algorithm for evaluating $T(G; x, y)$ at any (x, y) (including $(2, 1)$) for general graphs G was developed by Björklund *et al.* [15]. A representation of the Tutte polynomial for complete graphs was found by Pak [16]. The formula provided can be evaluated at $(2, 1)$ in polynomial time. Martin and Reiner [17] found an exponential generating function for the Tutte polynomial of a complete bipartite graph, but it is not applicable in the case $(2, 1)$. Jaeger *et al.* [13] showed that evaluating $T(G; x, y)$ at (x, y) is #P-hard for all pairs – including $(2, 1)$ – that are not in a certain set that they provide in their paper. Vertigan *et al.* [14] extended this result to (planar) bipartite graphs and thus to our use case (general scheme). Gebauer and Okamoto [18] proved #P-completeness of ForestCount for the three classes of bounded-degree, regular and chordal graphs and also present exponential-time algorithms for these problems. Stones [12] introduced and analysed the exponential-time algorithm for ForestCount on complete bipartite graphs that we use in this paper.

Another related problem is the enumeration of rooted spanning forests of complete bipartite graphs. Jin and Liu [19] give an easy-to-compute formula and prove that it equals the number of rooted spanning forests for any tuple (n_1, r_1, n_2, r_2) where n_i is the number of vertices and r_i the number of roots of vertex set i.

VI. Conclusion

In this paper we have characterised various two-dimensional RAID schemes and analysed the probability for data loss for each of them. More precisely, we considered the probability for data loss in the case that f disks fail. While previous analyses [2], [4] are only precise for at most five erasures, we have shown that the algorithm of Stones, which works for any f, can be applied to two of the schemes. Although this algorithm has an exponential worst-case running time, it is fast for small f. It remains open to find algorithms for the Lonestar and the general scheme, and in our future work we will also address the k-dimensional cases for $k > 2$ and other non-MDS codes.

We have further proven that the generalised problem, DataLossProbGen, is #P-hard. The exact classification of DataLossProbStd, DataLossProbFull and DataLossProbLS remains open even though we have shown that this question would be automatically answered (at least for the first two problems) by classifying the evaluation of the Tutte polynomial $T(G; x, y)$ at $(2, 1)$ for complete bipartite graphs G. So far the best known algorithm is the one by Stones.

Acknowledgment

We would like to thank Dr Rebecca Stones for providing the C code of her algorithm (see Section IV-C) and Dr Klaus Nagel for interesting discussions on these topics.

This work was partly developed in the *ADA-FS* project funded by the DFG Priority Program "Software for Exascale Computing" (SPPEXA, SPP 1648).

REFERENCES

[1] M. Grawinkel, L. Nagel, M. Mäsker, F. Padua, A. Brinkmann, and L. Sorth, "Analysis of the ECMWF Storage Landscape," in *Proceedings of the 13th USENIX Conference on File and Storage Technologies, FAST 2015, Santa Clara, CA, USA, February 16-19, 2015*, 2015, pp. 15–27. [Online]. Available: https://www.usenix.org/conference/fast15/technical-sessions/presentation/grawinkel

[2] M. Grawinkel, L. Nagel, and A. Brinkmann, "LoneStar RAID: Massive Array of Offline Disks for Archival Systems," *Trans. Storage*, vol. 12, no. 1, pp. 5:1–5:29, Jan. 2016. [Online]. Available: http://doi.acm.org/10.1145/2840810

[3] M. Grawinkel, G. Best, M. Splietker, and A. Brinkmann, "Lone Star Stack: Architecture of a Disk-Based Archival System," in *9th IEEE International Conference on Networking, Architecture, and Storage (NAS)*, 2014, pp. 176–185. [Online]. Available: http://dx.doi.org/10.1109/NAS.2014.35

[4] J.-F. Pâris, T. J. E. Schwarz, A. Amer, and D. D. E. Long, "Highly Reliable Two-dimensional RAID Arrays for Archival Storage," in *31st IEEE International Performance, Computing and Communications Conference (IPCCC)*, Dec 2012, pp. 324–331.

[5] J.-F. Pâris and A. Amer, "Using Shared Parity Disks to Improve the Reliability of RAID Arrays," in *28th IEEE International Performance, Computing and Communications Conference (IPCCC)*, Dec 2009, pp. 129–136.

[6] J.-F. Pâris, T. J. E. Schwarz, and D. D. E. Long, "Self-Adaptive Two-Dimensional RAID Arrays," in *26th IEEE International Performance, Computing and Communications Conference (IPCCC)*, April 2007, pp. 246–253.

[7] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-performance, Reliable Secondary Storage," *ACM Comput. Surv.*, vol. 26, no. 2, pp. 145–185, Jun. 1994. [Online]. Available: http://doi.acm.org/10.1145/176979.176981

[8] C. Huang and L. Xu, "STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures," *Computers, IEEE Transactions on*, vol. 57, no. 7, pp. 889–901, July 2008.

[9] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal Parity for Double Disk Failure Correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, ser. FAST'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 1–1. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973374.1973375

[10] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An Efficient Scheme For Tolerating Double Disk Failures in RAID Architectures," *Computers, IEEE Transactions on*, vol. 44, no. 2, pp. 192–202, Feb 1995.

[11] J. L. Hafner and K. K. Rao, "Notes on Reliability Models for Non-MDS Erasure Codes," IBM Research Division, Tech. Rep., 2006.

[12] R. J. Stones, "Computing the Number of h-edge Spanning Forests in Complete Bipartite Graphs," *Discrete Mathematics & Theoretical Computer Science*, vol. 16, no. 1, pp. 313–326, 2014. [Online]. Available: http://www.dmtcs.org/dmtcs-ojs/index.php/dmtcs/article/view/2311

[13] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh, "On the Computational Complexity of the Jones and Tutte Polynomials," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 108, pp. 35–53, 7 1990.

[14] D. L. Vertigan and D. J. A. Welsh, "The Computational Complexity of the Tutte Plane: the Bipartite Case," *Combinatorics, Probability and Computing*, vol. 1, pp. 181–187, 6 1992.

[15] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, "Computing the Tutte Polynomial in Vertex-Exponential Time," in *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Oct 2008, pp. 677–686.

[16] I. Pak, "Computation of Tutte Polynomials of Complete Graphs," 1993, available at http://www.math.ucla.edu/ pak/papers/research.htm.

[17] J. L. Martin and V. Reiner, "Cyclotomic and Simplicial Matroids," *Israel Journal of Mathematics*, vol. 150, no. 1, pp. 229–240, 2005. [Online]. Available: http://dx.doi.org/10.1007/BF02762381

[18] H. Gebauer and Y. Okamoto, "Fast Exponential-time Algorithms for the Forest Counting in Graph Classes," in *Proceedings of the Thirteenth Australasian Symposium on Theory of Computing - Volume 65*, ser. CATS '07. Darlinghurst, Australia: Australian Computer Society, Inc., 2007, pp. 63–69.

[19] Y. Jin and C. Liu, "Enumeration for Spanning Forests of Complete Bipartite Graphs," *Ars Comb.*, vol. 70, 2004.