TUCS

Toni Ernvall

On Distributed Storage Codes

Turku Centre for Computer Science

TUCS Dissertations
No 192, February 2015

On Distributed Storage Codes

Toni Ernvall

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in Quantum Auditorium on February 13, 2015, at 12 noon.

University of Turku
Department of Mathematics and Statistics
FI-20014 Turku, Finland

2015

Supervisors

Professor Camilla Hollanti
Department of Mathematics and Systems Analysis
Aalto University
FI-00076 Aalto
Finland

Docent Jyrki Lahtonen
Department of Mathematics and Statistics
University of Turku
FI-20014 Turku
Finland

Doctor Roope Vehkalahti
Department of Mathematics and Statistics
University of Turku
FI-20014 Turku
Finland

Reviewers

Professor Alexandros G. Dimakis
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
USA

Professor Chao Tian
Department of Electrical Engineering & Computer Science
The University of Tennessee at Knoxville
Min H. Kao Building, Rm605
1520 Middle Drive
Knoxville, TN 37996
USA

Opponent

Professor Joachim Rosenthal
Department of Mathematics
University of Zurich
Winterthurerstrasse 190
CH-8057 Zürich
Switzerland

ISBN 978-952-12-3166-7
ISSN 1239-1883

Abstract

Distributed storage systems are studied. The interest in such system has become relatively wide due to the increasing amount of information needed to be stored in data centers or different kinds of cloud systems. There are many kinds of solutions for storing the information into distributed devices regarding the needs of the system designer. This thesis studies the questions of designing such storage systems and also fundamental limits of such systems. Namely, the subjects of interest of this thesis include heterogeneous distributed storage systems, distributed storage systems with the exact repair property, and locally repairable codes. For distributed storage systems with either functional or exact repair, capacity results are proved. In the case of locally repairable codes, the minimum distance is studied.

Constructions for exact-repairing codes between minimum bandwidth regeneration (MBR) and minimum storage regeneration (MSR) points are given. These codes exceed the time-sharing line of the extremal points in many cases. Other properties of exact-regenerating codes are also studied. For the heterogeneous setup, the main result is that the capacity of such systems is always smaller than or equal to the capacity of a homogeneous system with symmetric repair with average node size and average repair bandwidth. A randomized construction for a locally repairable code with good minimum distance is given. It is shown that a random linear code of certain natural type has a good minimum distance with high probability. Other properties of locally repairable codes are also studied.

Tiivistelmä

Tämä tutkielma käsittelee hajautettuja tallennusjärjestelmiä. Tällaisista järjestelmistä on tullut hyvin kiinnostavia tutkimuskohteita datakeskuksissa ja erilaisissa pilvipalveluissa säilytettävän tiedon määrän jatkuvan kasvun vuoksi. Riippuen järjestelmän suunnittelijan tarpeista tietoa voidaan säilyttää monin eri tavoin erilaisissa hajautetuissa tallennusjärjestelmissä. Tässä tutkielmassa tarkastellaan tällaisten järjestelmien suunnittelua ja niihin liittyviä rajoituksia. Tärkeimmät käsitteet ovat heterogeeniset hajautetut tallennusjärjestelmät, tarkasti korjaavat hajautetut tallennusjärjestelmät ja paikallisesti korjaavat koodit. Sekä funktionaalisesti että tarkasti korjaaville hajautetuille tallennusjärjestelmille todistetaan kapasiteettituloksia ja paikallisesti korjaavien koodien tapauksessa tutkitaan minimietäisyyttä.

Työssä esitellään tarkasti korjaavien koodien konstruktioita MBR- ja MSR-ääripisteiden välillä. Monissa tapauksissa nämä konstruktiot ylittävät ääripisteiden interpoloinnilla saavutettavan triviaalin koodin suorituskyvyn. Tämän lisäksi myös muita tarkasti korjaavien koodien ominaisuuksia tutkitaan. Heterogeenisessä tapauksessa päätulos on, että tällaisen tallennusjärjestelmän kapasiteetti on aina korkeintaan symmetrisesti korjaavan homogeenisen tallennusjärjestelmän kapasiteetti, jossa tallennusyksikön koko on sama kuin heterogeenisen järjestelmän keskimääräinen tallennusyksikön koko ja jossa korjauskaistanleveys on sama kuin heterogeenisen järjestelmän keskimääräinen korjauskaistanleveys. Paikallisesti korjaaville koodeille esitellään satunnaiskonstruktio. Lisäksi osoitetaan, että satunnaisella lineaarisella koodilla, joka on tiettyä luonnollista tyyppiä, on suurella todennäköisyydellä suuri minimietäisyys. Myös paikallisesti korjaavien koodien osalta tutkitaan muitakin ominaisuuksia.

Acknowledgements

First of all, I want to thank my advisors Professor Camilla Hollanti, Docent Jyrki Lahtonen, and Doctor Roope Vehkalahti. They all have had an important role in my studies. Jyrki presented me interesting problems in MIMO MAC, that lead to my first conference paper and hence gave a good start for my studies. Camilla introduced me to the world of distributed storage that became the subject of my thesis. Roope has guided me through my doctoral studies in practical and nonpractical matters. His support and opinions have had a huge positive impact on my work.

I am thankful for Professor Alexandros G. Dimakis and Professor Chao Tian for the preliminary examination of my work. It was an honor for me that Professor Joachim Rosenthal agreed to act as the opponent.

I thank Doctor Dave Karpuk for English corrections. This thesis would have been nonreadable without them.

I am grateful for funding provided by the Turku Centre for Computer Science and the University of Turku Graduate School. The Department of Mathematics and Statistics is full of helpful and friendly people. Hence, I want to thank my colleagues for providing such a nice working environment.

I also thank all my collaborators. Especially, I would like to thank Professor Salim El Rouayheb for teaching me a lot of practical things about distributed storage and for the nice visit at the Princeton University.

Finally, and most importantly, I want to thank my family, *i.e.*, my parents, Ame, and Mari for everything.

Turku, January 2015 — Toni Ernvall

List of original publications

1. T. Ernvall, S. El Rouayheb, C. Hollanti, H. V. Poor: Capacity and Security of Heterogeneous Distributed Storage Systems, *IEEE Journal on Selected Areas in Communications*, Volume 31, Issue 12, pp. 2701-2709 (2013).

2. T. Ernvall: Codes Between MBR and MSR Points With Exact Repair Property, *IEEE Transactions on Information Theory*, Volume 60, Issue 11, pp. 6993-7005 (2014).

3. T. Ernvall, T. Westerbäck, C. Hollanti, R. Freij: Constructions and Properties of Linear Locally Repairable Codes, *IEEE Transactions on Information Theory*, submitted (2014). Available: arXiv:1410.6339 [cs.IT].

Contents

1 Summary **1**
1.1 Introduction 1
1.2 Codes 2
1.3 Distributed Storage Systems and Capacity 3
1.4 Our Three Main Topics 6
1.4.1 Exact-Regenerating Codes 6
1.4.2 Heterogeneous Distributed Storage Systems 8
1.4.3 Locally Repairable Codes 9
1.5 Conclusion 11

Chapter 1

Summary

1.1 Introduction

The need for systems which store huge amount of information is growing fast in the internet era. It took about four years for Dropbox to acquire its first 100 million users, but only 10 additional months for this number to double [29]. The information flood has forced us to think about how we store information. Is it safe against malfunctioning machinery? Are privacy needs being met? How expensive is it to store data?

The main concept of this thesis is *a code*. By a code, we mean a nonempty subset C of Q^n where Q is a set of size q. For example, Q can be a finite field $\mathbb{F}_q$ of q elements. In classical coding theory codes were used for error correction, *i.e.*, to protect a message against interference during the transmission process. Here, the purpose of the codes is different. We are storing information rather than transmitting it. In this case the interference is not the problem. The problem is that the devices in which we store information maybe become unavailable, break, or malfunction. However, despite the probability of a device becoming unavailable is small, when the number of devices is large, the probability of losing at least one device is too large.

Let us say we have $n = 1000$ devices and the probability that a device breaks in some time frame is $p = 0.001$. Then, the probability of the event that at least one of the devices breaks in the given time frame is

$$\begin{aligned} &\mathbb{P}(\text{at least one of the devices breaks}) \\ =&1 - \mathbb{P}(\text{none of the devices breaks}) \\ =&1 - (1-p)^n \\ \approx&0.63. \end{aligned} \tag{1.1}$$

This is quite a large probability for a real world application. It is reasonable to expect to lose devices in a storage system.

Hence, protecting information against device losses is of great importance. How can one do this? Of course, replicating all the data would protect the information quite well. Or taking two copies of the data stored in each device would do this even better. However, this would be highly expensive when compared to the achieved level of protection. This is the reason one uses codes in storage process.

1.2 Codes

In this section, we introduce some basic concepts from coding theory. A good reference for the concepts and results of classical coding theory is [59]. Recall that a code is a nonempty subset C of Q^n where Q is a set of size q. A code with only one element is called trivial and a code with at least two elements is called nontrivial.

For codewords $\mathbf{x}, \mathbf{y} \in Q^n$, $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n)$ define the Hamming distance

$$d(\mathbf{x}, \mathbf{y}) = |\{i \mid 1 \leq i \leq n, x_i \neq y_i\}|.$$

Note that the distance function defines a metric on Q^n. The weight of $\mathbf{x}$ is

$$w(\mathbf{x}) = d(\mathbf{x}, \mathbf{0}).$$

The minimum distance of a nontrivial code C is

$$d_{\min}(C) = \min\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in C, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}.$$

If a nontrivial code C is a subspace of $\mathbb{F}_q$, it is called *linear*. Notice that for a linear code C the minimum distance has a simpler expression

$$d_{\min}(C) = \min\{w(\mathbf{x}) \mid \mathbf{x} \in C \setminus \{\mathbf{0}\}\}.$$

A linear code of length n and dimension $n - d_{\min}(C) + 1$ is called a *maximum distance separable* (MDS) code.

How does one use codes for storage? Suppose we have a q-ary code C of length n, size $|C| \geq 2$, and minimum distance $d_{\min}(C)$. We have n storing devices, say, memory sticks. We can build a storage system for a file of size $B = \lfloor \log_q |C| \rfloor$ in the following way: Map the elements of $\mathbb{F}_q^B$ onto C using any bijection $f : \mathbb{F}_q^B \to C$. If the mapped file is $(x_1, \ldots, x_B)$ and $f(x_1, \ldots, x_B) = (y_1, \ldots, y_n)$ then store y_j into jth storing device for $j = 1, \ldots, n$. Now, any $n - d_{\min}(C) + 1$ can recover the stored file. Hence, if we lose $d_{\min}(C) - 1$ or fewer devices, we have not still lost the saved file. This idea is the starting point for using codes in storage.

There are codes for different purposes and needs. The above concepts are familiar from classical coding theory. Despite the many similarities between classical coding theory and the theory of distributed storage systems, there are still big differences, beginning with the definition of a code. In the next section we will define more exactly the most important object of this thesis, namely, *a distributed storage system*.

What is essential for the systems we use for storing information? Certainly, the reliability and the ease of use in storing information in and reading information from the system. To achieve reliability, we will distribute information into several storing devices as described in the example above. The particular use of the system dictates what we require of it. In most cases, reliability is the main concern. But after that, we have still choices to make. One must decide which is more important: storing the information using as little space as possible, making repairs using as little bandwidth as possible, or contacting as few devices as possible during the repair process?

Let us return to the example above where we used a code for storing information. The drawback of the solution is the complexity of the regeneration of the lost devices. If we lose a device we may contact any $n-d_{\min}(C)+1$ surviving devices (or more) and using their content build the original file again and then reconstruct the content of the lost node. However, this procedure, which regenerates one Bth of the original file, requires as much bandwidth as recovering the whole file. By comparison, if we had just split the original file into B pieces and saved some number of each of their replicas in other devices, then the regeneration process would have required only the amount of one Bth of the file to regenerate the lost node. One can see that both solutions have benefits and detriments.

1.3 Distributed Storage Systems and Capacity

In a distributed storage system, the size of a stored file clearly sets requirements for the system. From that point of view, a maximum distance separable code gives an optimal solution. However, as "hardware failure is the norm rather than the exception" [2] in a large distributed storage system one has to make sure that the repair process of lost nodes can be handled fluently.

In the literature, three kinds of repair cost metrics are studied: *repair bandwidth* [10], *disk-I/O* [53], and *repair locality* [20, 31, 34, 38]. This thesis studies the first and the last of the three, namely, the repair bandwidth and the repair locality.

When the repair bandwidth is the main concern, we assume that each set of d nodes can repair a lost device. The value we try to minimize is the total repair bandwidth, *i.e.*, the bandwidth needed to repair a lost node.

In the following we will define distributed storage systems with exact repair and distributed storage systems with functional repair. For a distributed storage system with exact repair we follow the formal definition given in [56] that is generalized in a natural way. The definition of a distributed storage system with functional repair is from [10].

Definition 1.3.1 *(A distributed storage system with exact repair): A distributed storage system with exact repair with parameters (n, k, d), corresponding to the number of nodes (*i.e.* storage devices), the data regeneration degree, and the node repair degree, respectively, with node size α, total repair bandwidth $\gamma = d\beta$, and storage capacity B is an injective function $f : U^B \to T^n$ with the following properties. Here, U is a set of size $q > 1$, $T = U^\alpha$, and $V = U^\beta$. Write $DSS = f(U^B)$ to be the image of the function. The minimum distance $d_{\min}$ of DSS is*

$$d_{\min} \geq n - k + 1.$$

Suppose $i, j \in [n]$, $i \in S \subseteq [n]$ with $|S| = d$ and $j \notin S$. For all such i, j, S we have the functions $g^{helper}_{i,j,S} : T \to V$ and $h^{repair}_{j,S} : V^d \to T$ with the following properties.

If $f(\mathbf{x}) = (y_1, y_2, \ldots, y_n)$ and $S = \{i_1, i_2, \ldots, i_d\}$, then

$$h^{repair}_{j,S}(g^{helper}_{i_1,j,S}(y_{i_1}), g^{helper}_{i_2,j,S}(y_{i_2}), \ldots, g^{helper}_{i_d,j,S}(y_{i_d})) = y_j.$$

The definition of a distributed storage system with functional repair has some differences when compared to exact-repairing codes. Again, the parameters n, k, d, α, γ and β are as in the definition of distributed storage system with exact repair. Instead of being a time-invariant code, the system varies over time during the node loss/repair processes.

A distributed storage system with functional repair consists of n storage devices (called *nodes*), each of size α. Each set of nodes of size $k(< n)$ must be able to recover the file stored in the system. The node repair process is as follows: If we lose a node v_i^{old} we replace it with a new node v_i^{new} (called *a newcomer*) while preserving the file recovery property and the node repairing property of the system. In the node repair process it is assumed that each of the d nodes involved in the repair process transmits an amount β of information to the newcomer and hence the total repair bandwidth γ is $d\beta$.

For the functionally repairing codes, there are two assumptions that can be made: requiring that such a code must work for an unlimited time period, or that it only has to be able to work some given time period. By limited or unlimited time period we mean that it must be able to tolerate bounded or respectively unbounded number of node failures/repairs. In the first paper of this thesis [16] only the requirement of a bounded number of failures/repairs is considered. However, results which dictate something to be impossible for a bounded number of repairs/failures naturally dictate the same thing to be impossible for an unbounded number of repairs/failures. Notice also that in the case of distributed storage system with exact or functional repair we may assume that $k \leq d$ since if any d nodes can repair any node then they certainly can repair k nodes and hence recover the whole stored file.

The capacity of a distributed storage system is the largest file size that can be stored in a system with given parameters. We formally define capacity as follows:

Definition 1.3.2 *(Capacity): Let $(C_j)_{j\in\mathbf{Z}_+}$ be a sequence of functionally repairing (respectively exact repairing) codes corresponding parameters (n, k, d), node size α_j, total repair bandwidth $\gamma_j = \alpha_j\gamma/\alpha$ and storing a file of size B_j. A file size $c(n, k, d, \alpha, \gamma)$ is said to be achievable under the assumption of functional repair (respectively exact repair) if*

$$\lim_{j\to\infty} \frac{C_j}{\alpha_j} = \frac{c(n, k, d, \alpha, \gamma)}{\alpha}.$$

The capacity $C_{n,k,d}(\alpha, \gamma)$ of functionally repairing (respectively $C^{exact}_{n,k,d}(\alpha, \gamma)$ of exact repairing) codes is the supremum of all achievable file sizes under the assumption of functional repair (respectively exact repair).

It is easy to check that if s is a positive number, then

$$C_{n,k,d}(s\alpha, s\gamma) = sC_{n,k,d}(\alpha, \gamma) \text{ and } C^{\text{exact}}_{n,k,d}(s\alpha, s\gamma) = sC^{\text{exact}}_{n,k,d}(\alpha, \gamma).$$

The pioneering work [10] by Dimakis *et al.* gives an elegant solution to the problem of finding the capacity under the assumption of functional repair. The capacity in that case is

$$C_{k,d}(\alpha, \gamma) := C_{n,k,d}(\alpha, \gamma) = \sum_{j=0}^{k-1} \min\{\alpha, (d - j)\beta\} \tag{1.2}$$

where $\gamma = d\beta$ as before. This result was proved using results from network coding proved in [1] by Ahlswede *et al.* The two seminal observations here were the following: When repairing a node, it is enough to transmit less data than the size of the stored file. Also, by increasing the number of repairing nodes the total repair bandwidth can be decreased. There is a trade-off between the size of information stored in one node and the total repair bandwidth. Storing more than one kth of the information in each node can make the repair process easier.

The concept of functional repair was introduced in [10]. There, the number of node failures/repairs was assumed to be restricted. However, the upper bound for the number of failures was allowed to be arbitrarily large. In [61] it was proved that the capacity under the assumption of functional repair is achievable even under the stronger assumption that the number of node failures is not restricted. Notice also that unlike in the case of functional repair it is not known if the capacity under the assumption of exact repair depends on the number of nodes.

1.4 Our Three Main Topics

This thesis considers three topics: exact-regenerating codes, heterogeneous distributed storage systems, and locally repairable codes. In this section we introduce these three topics by providing the necessary definitions and describing the subsequent results.

1.4.1 Exact-Regenerating Codes

If the size of the stored file is fixed as B, the expression 1.2 for the capacity under the assumption of functional repair defines a tradeoff between the node size α and the total repair bandwidth γ. The two extreme points are called the minimum storage regeneration (MSR) point and the minimum bandwidth regeneration (MBR) point. The MSR point is achieved by first minimizing α and then minimizing γ to obtain

$$\begin{cases} \alpha_{\text{MSR}} = \frac{B}{k} \\ \gamma_{\text{MSR}} = \frac{dB}{k(d-k+1)}. \end{cases} \tag{1.3}$$

By first minimizing γ and then minimizing α leads to the MBR point

$$\begin{cases} \alpha_{\text{MBR}} = \frac{2dB}{k(2d-k+1)} \\ \gamma_{\text{MBR}} = \frac{2dB}{k(2d-k+1)}. \end{cases} \tag{1.4}$$

These extreme points are also achievable under the stronger assumption of exact repair [7, 39]. For the MSR point the achievability is proved asymptotically and the MBR point is strictly achievable. Despite the fact that both the MBR and the MSR points are known to be at least asymptotically achievable also under the assumption of exact repair, very little was known about the achievability of the interior points for a long time. Shah *et al.* has shown that almost all the interior points in the functional-capacity curve were impossible to achieve in the non-asymptotic case [46]. However, in the asymptotic scenario the question was still open. On the other hand, constructions better than the trivial time-sharing of the extremal points, did not exist either. In 2013 Tian proved that generally there exists a non-vanishing gap between the capacities under the assumptions of exact and functional repair [56]. This was done by studying the case $(n, k, d) = (4, 3, 3)$. Also, in the same year three constructions for the interior points were presented. Namely, these were [57] by Tian *et al.*, [43] by Sasidharan *et al.*, and [13] by the author.

The extended version of the last one [14] gives two constructions with almost the same performance, both exceeding the time-sharing line for many triples of (n, k, d). The performances of these constructions are also close to optimal in the case that the values n, k, and d are close to each other and relatively large. The performance of the first of these constructions is

$$P^1_{n,k,d}\left(\alpha, \frac{(d-k+i)\alpha}{d-k+1}\right) = \frac{ni\alpha}{n-k+i}.$$

The article also studies the relationships between exact-repairing codes corresponding to different parameters. The first result ties together the capacities of exact-repairing codes and functionally repairing codes in the following way

$$C^{\text{exact}}_{hn,hk,hd}(\alpha, \gamma) \leq C^{\text{exact}}_{n,k,d}(\alpha_2, \gamma_2),$$

where h is a positive integer and

$$\alpha_2 = C_{h,hd}(\alpha, \gamma) \text{ and } \gamma_2 = C_{h,hd}(\gamma, \gamma).$$

The second result generalizes the method of puncturing to storage codes leading to the following bound

$$C^{\text{exact}}_{n-1,k-1,d-1}(\alpha, \frac{(d-1)\gamma}{d}) \geq C^{\text{exact}}_{n,k,d}(\alpha, \gamma) - \alpha.$$

In addition to the above mentioned constructions, Goparaju *et al.* have a construction [22] for the same regime. This was presented in 2014.

In their construction, Sasidharan *et al.* combine a combinatorial structure with MDS codes to get codes they call canonical codes, for which $k = d$. These canonical codes are combined with polynomial evaluations to get codes for the case $k < d$. Tian *et al.* exploit block designs with MDS codes to build exact-repairing codes. In their work, MDS codes are used in two steps. Roughly speaking, one step for node repair and another step for file regeneration.

In contrast to these two works, in my work MSR codes are used instead of MDS codes. The benefit of this is that MSR codes already include a nontrivial repair menchanism, which is not the case with general MDS codes. My construction also exploits the code homogenizing procedure introduced in [16] that significantly shortens the construction and makes it clearer.

Goparaju *et al.* build codes by using methods presented in my work and in the work by Sasidharan *et al.* with a new ingredient. This new component is that they add MSR codes that are optimal for all suitable values of d to get new codes with good performance.

In addition to Tian's result for the case $(n, k, d) = (4, 3, 3)$, there are some other upper bounds for the capacity of exact-repairing codes. Sasidharan *et al.* [44] have presented an upper bound for the capacity of distributed storage system under the assumption of exact repair. Also, Duursma presented another upper bound for the same capacity in [12]. These all differ from the bounds I present in [14] in that the former two bounds give explicit bounds depending on for example node size and repair bandwidth. In contrast to these, my bounds illustrate the relationships between storage systems with different parameters. They also give a way to derive new codes from already existing ones.

1.4.2 Heterogeneous Distributed Storage Systems

In the literature the most typical setup for a distributed storage system with either exact or functional repair is homogeneous, in the sense that each storage device stores an equal amount of information and the repair bandwidth is also constant. This is the setup we assumed above. However, this does not have to be the case. In many applications, for example in data centers, this is a justified assumption but also in many applications the assumption is too narrow. Indeed, peer-to-peer (p2p) cloud storage systems and internet caching systems for video-on-demand applications are examples of systems that are more natural to be modeled using the heterogeneous setup.

In a heterogeneous setup the ith node is of size α_i, and in the repair process, when the ith node repairs the jth node and S is the set of indices of all helper nodes, then amount of information transmitted from the ith node to the jth node is β_{ijS}. For the average total repair bandwidth γ_j of the jth node we write

$$\gamma_j = \binom{n-1}{d}^{-1} \sum_{\substack{S:j\notin S\\ |S|=d}} \sum_{i\in S} \beta_{ijS}. \tag{1.5}$$

For the average node size we write $\overline{\alpha} = \frac{1}{n}\sum_{i=1}^{n}\alpha_i$ and average total repair bandwidth $\overline{\gamma} = \frac{1}{n}\sum_{i=1}^{n}\gamma_i$.

In a homogeneous setup we have $\alpha_i = \alpha$ for all i and

$$\sum_{i\in S}\beta_{ijS} = \gamma$$

for each subset $S \subseteq \{1,2,\ldots,n\} \setminus \{j\}$ of size d. However, it does not necessarily hold that the values β_{ijS} are fixed, *i.e.*, $\beta_{ijS} = \beta$. If we also assume that the repair is symmetric, then $\beta_{ijS} = \beta$ for all i, j, S.

In [16] heterogeneous distributed storage systems are studied. Both capacity with exact or functional repair and capacity under assumption of different secrecy aspects are objects of interest. The main theorems state that homogeneous setup with symmetric repair maximizes the system capacity in all cases. This has been the underlying assumption before but not proved until now. In the functional-repair case the main result can be formulated as

$$C \leq C_{k,d}(\overline{\alpha}, \overline{\gamma})$$

where C is the system capacity, $\overline{\alpha}$ is the average node size, and $\overline{\gamma}$ is the average total repair bandwidth.

Let us next study the secrecy capacity of the system. The model followed here is from [36]. The secrecy capacity C_s of the system is defined to be the maximum amount of information that can be delivered to a user without revealing any information to the eavesdropper. The eavesdropper is assumed to be passive, *i.e.*, she can only read data but not modify it. This is studied under the assumption of information theoretic security. Information theoretic security refers to the system's capability to provide data confidentially, independently of cryptographic methods. Also in the case of secrecy capacity, the homogeneous model is shown to be the best choice.

1.4.3 Locally Repairable Codes

In many applications the number of contacted nodes in the repair process is a crucial issue. In a distributed storage system the repair degree was lower bounded by the file regeneration degree. However, if we relax the requirement that *any* set of given size can repair a lost node, we can reduce this number.

In the case of repair locality, the object to be minimized is the number of helper nodes in a repair process. In this scenario it is not required that each set of a given size can be a repairing set. Instead, it is assumed that for each node, there exists a set of nodes that can repair it even if some of repairing nodes are unavailable. Locally repairable codes were introduced in [20, 31, 34]. The generalized definition of (r, δ)-locality is from [38]. Locally repairable codes are used at least in two large-scale distributed storage systems, namely in Windows Azure Storage and in Distributed File System RAID used by Facebook [52].

Definition 1.4.1 *(A locally repairable code): Given a finite field $\mathbb{F}_q$ with q elements and an injective function $f : \mathbb{F}_q^k \to \mathbb{F}_q^n$, let C denote the image of f. We say that C is a locally repairable code and has* all-symbol (r, δ)-locality *with parameters (n, k, d), if the code C has minimum distance d and all the n symbols of the code have (r, δ)-locality. The jth symbol has (r, δ)-locality if there exists a subset $S_j \subseteq \{1, \ldots, n\}$ such that $j \in S_j$, $|S_j| \leq r + \delta - 1$ and the minimum distance of the code obtained by deleting code symbols corresponding the elements of $\{1, \ldots, n\} \setminus S_j$ is at least δ. Locally repairable codes are defined when $1 \leq r \leq k$.*

Remark 1.4.1 *Notice the different purposes of k and d in the the definitions of locally repairable codes and distributed storage systems with either* *functional or exact repair. In the case of distributed storage system with exact or functional repair, k is the data regeneration degree. In the case of locally repairable code, it is the dimension. Also, in the case of distributed storage system with exact or functional repair, d is the node repair degree. In the case of locally repairable code, it is the minimum distance. We use this unfortunately slightly misleading notation to be consistent with established notation in the literature.*

In [38, 26] it is shown that we have the following bound for a linear locally repairable code C of length n, dimension k, minimum distance d and all-symbol (r, δ)-locality:

$$d \leq n - k - \left(\left\lceil \frac{k}{r} \right\rceil - 1 \right) (\delta - 1) + 1. \tag{1.6}$$

Codes achieving this bound are called *optimal*. The information theoretic analogy of the above bound for any (linear or nonlinear) code is proved in [32] in the case $\delta = 2$. Optimal locally repairable codes are constructed in *e.g.* [52, 49, 41, 51].

Locally repairable codes are studied in [17]. The paper gives methods to find bigger and smaller codes from already existing codes. In several cases when the code used as a starting point is optimal, the resulting code is also optimal. In addition, a construction to find codes that are near to optimal is given. Using the same ideas as in the construction it is shown that almost every matrix of certain natural type generates an almost optimal code if the field size is large enough. One notable aspect is that these results are proved using only some results from elementary linear algebra. However, we use the notation of *circuit* from matroid theory to simplify our arguments. In this case a circuit has a simple interpretation in the language of linear algebra. In addition, constructing linear locally repairable codes over small fields is studied.

There are several constructions of locally repairable codes. In [26] regenerating codes with locality are constructed. Tamo *et al.* use Reed-Solomon codes with MDS codes to construct LRCs in [52]. In [49] MDS codes combined with techniques from network coding are used to construct optimal codes for several parameter sets. This construction and our construction have a common property that in both constructions the generator matrix is built iteratively by searching new column vectors that are linearly independent with certain previous column vectors. In [51] LRCs that can be seen as a generalization of Reed-Solomon codes are constructed. Silberstein *et al.* construct LRCs by using Gabidulin codes in [48].

The main difference between our construction and the constructions mentioned above is that we do not try to restrict the used field size. This is a drawback of our construction. However, as a benefit we get very general codes. Also, we show that by using random matrices with guaranteed locality, we get good codes with high probability.

1.5 Conclusion

In this thesis, we study methods for constructing codes and analyzing fundamental limits of codes in the context of distributed storage. Distributed storage systems with both functional and exact repair with homogeneous and heterogeneous setup are subjects of interest. Also, locally repairable codes with all-symbol locality are studied. For the exact-repairing distributed storage systems, the capacity is the main quantity of study. In the case of locally repairable codes, the largest achievable minimum distance is of great interest.

Open problems for future study include finding the capacity of distributed storage system under the assumption of exact repair, especially in the case that d is notably smaller than n. Also, the exact expression for the largest achievable minimum distance for a locally repairable code with all-symbol locality is not completely solved.

Bibliography

[1] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R.W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] D. Borthakur. The Hadoop distributed file system: Architecture and design. `https://hadoop.apache.org/docs/r0.18.3/hdfs_design.html`. Accessed: 2014-10-10.

[3] V. Cadambe and A. Mazumdar. An upper bound on the size of locally recoverable codes. In *Proceedings of 2013 International Symposium on Network Coding (NetCod)*, pages 1–5, 2013.

[4] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li. Optimal repair of MDS codes in distributed storage via subspace interference alignment. *arXiv:1106.1250 [cs.IT]*, 2011.

[5] V. R. Cadambe, C. Huang, and J. Li. Permutation code: Optimal exact-repair of a single failed node in MDS code based distributed storage systems. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 1225–1229, 2011.

[6] V. R. Cadambe, S. A. Jafar, and H. Maleki. Distributed data storage with minimum storage regenerating codes - exact and functional repair are asymptotically equally efficient. *arXiv:1004.4299v1 [cs.IT]*, 2010.

[7] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh. Asymptotic interference alignment for optimal repair of MDS codes in distributed storage. *IEEE Transactions on Information Theory*, 59(5):2974–2987, 2013.

[8] T. Chan, A. Grant, and T. Britz. Quasi-uniform codes and their applications. *IEEE Transactions on Information Theory*, 59(12):7915–7926, 2013.

[9] D. Cullina, A. G. Dimakis, and T. Ho. Searching for minimum storage regenerating codes. *arXiv:0910.2245*, 2009.

[10] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, 2010.

[11] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh. A survey on network codes for distributed storage. *arXiv:1004.4438*, 2010.

[12] I. M. Duursma. Outer bounds for exact repair codes. *arXiv:1406.4852 [cs.IT]*, 2014.

[13] T. Ernvall. Exact-regenerating codes between MBR and MSR points. In *Proc. IEEE Information Theory Workshop*, pages 1–5, 2013.

[14] T. Ernvall. Codes between MBR and MSR points with exact repair property. *IEEE Transactions on Information Theory*, 60(11):6993–7005, 2014.

[15] T. Ernvall, S. El Rouayheb, C. Hollanti, and H. V. Poor. Capacity and security of heterogeneous distributed storage systems. In *Proc. IEEE International Symposium on Information Theory, Istanbul, Turkey*, pages 1247–1251, 2013.

[16] T. Ernvall, S. El Rouayheb, C. Hollanti, and H. Vincent Poor. Capacity and security of heterogeneous distributed storage systems. *IEEE Journal on Selected Areas in Communications*, 31(12):2701–2709, 2013.

[17] T. Ernvall, T. Westerbäck, C. Hollanti, and R. Freij. Constructions and properties of linear locally repairable codes. *IEEE Transactions on Information Theory*, submitted, 2014.

[18] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *Proc. 19th ACM Symposium on Operating Systems Principles*, pages 29–43, 2003.

[19] N. Golrezaei, A. G. Dimakis, , and A. F. Molisch. Wireless device-to-device communications with distributed caching. In *Proc. IEEE International Symposium on Information Theory (Cambridge, MA)*, pages 2781–2785, 2012.

[20] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin. On the locality of codeword symbols. *IEEE Transactions on Information Theory*, 58(11):6925–6934, 2011.

[21] S. Goparaju and R. Calderbank. Binary cyclic codes that are locally repairable. In *2014 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 676–680, 2014.

[22] S. Goparaju, S. El Rouayheb, and A. R. Calderbank. New codes and inner bounds for exact repair in distributed storage systems. In *2014 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1036–1040, 2014.

[23] A. Ha. P2P Storage Startup Space Monkey Raises $2.25m Led By Google Ventures And Venture51. `http://techcrunch.com/2012/07/11/space-monkey-seed-round/`. Accessed: 2014-10-10.

[24] C. Huang, M. Chen, and J. Lin. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. In *Sixth IEEE International Symposium on Network Computing and Applications*, pages 79–86, 2007.

[25] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin. Erasure coding in Windows Azure storage. In *Proc. 2012 USENIX Annual Technical Conference (ATC)*, pages 15–26, 2012.

[26] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar. Codes with local regeneration. *arXiv:1211.1932*, 2012.

[27] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Shea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 190–201, 2000.

[28] D. Leong, A. G. Dimakis, and T. Ho. Distributed storage allocations. *arXiv:1011.5287v3*, 2012.

[29] R. Miller. How Dropbox stores stuff for 200 million users. `http://www.datacenterknowledge.com/archives/2013/10/23/how-dropbox-stores-stuff-for-200-million-users/`. Accessed: 2014-10-10.

[30] V. Ntranos, G. Caire, and A. G. Dimakis. Allocations for heterogeneous distributed storage. In *Proc. IEEE International Symposium on Information Theory (Cambridge, MA)*, pages 2761–2765, 2012.

[31] F. Oggier and A. Datta. Self-repairing homomorphic codes for distributed storage systems. In *INFOCOM, 2011 Proceedings IEEE*, pages 1215–1223, 2011.

[32] D. S. Papailiopoulos and A. G. Dimakis. Locally repairable codes. In *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 2771–2775, 2012.

[33] D. S. Papailiopoulos, A. G. Dimakis, and V. R. Cadambe. Repair optimal erasure codes through hadamard designs. *IEEE Transactions on Information Theory*, 59(5):3021–3037, 2013.

[34] D. S. Papailiopoulos, J. Luo, A. G. Dimakis, C. Huang, and J. Li. Simple regenerating codes: Network coding for cloud storage. In *INFOCOM, 2012 Proceedings IEEE*, pages 2801–2805, 2012.

[35] S. Pawar, S. El Rouayheb, and K. Ramchandran. On secure distributed data storage under repair dynamics. In *Proc. IEEE International Symposium on Information Theory (Austin, TX)*, pages 2543–2547, 2010.

[36] S. Pawar, S. El Rouayheb, and K. Ramchandran. Securing dynamic distributed storage systems against eavesdropping and adversarial attacks. *IEEE Transactions on Information Theory*, 57(10):6734–6753, 2011.

[37] S. Pawar, S. El Rouayheb, H. Zhang, K. Lee, and K. Ramchandran. Codes for a distributed caching based video-on-demand system. In *Proc. Asilomar Conference on Signals, Systems, and Computers (Pacific Grove, CA)*, pages 1783–1787, 2011.

[38] N. Prakash, Govinda M. Kamath, V. Lalitha, and P. Vijay Kumar. Optimal linear codes with a local-error-correction property. In *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 2776–2780, 2012.

[39] K. V. Rashmi, N. B. Shah, and P. V. Kumar. Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction. *IEEE Transactions on Information Theory*, 57(8):5227–5239, 2011.

[40] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran. Explicit construction of optimal exact regenerating codes for distributed storage. *arXiv:0906.4913v2*, 2009.

[41] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath. Optimal locally repairable and secure codes for distributed storage systems. *arXiv:1210.6954 [cs.IT]*, 2012.

[42] S. El Rouayheb and K. Ramchandran. Fractional repetition codes for repair in distributed storage systems. In *Proc. 48th Annual Allerton*, pages 1510–1517, 2010.

[43] B. Sasidharan and P. V. Kumar. High-rate regenerating codes through layering. *arXiv:1301.6157*, 2013.

[44] B. Sasidharan, K. Senthoor, and P. V. Kumar. An improved outer bound on the storage-repair-bandwidth tradeoff of exact-repair regenerating codes. *arXiv:1312.6079*, 2013.

[45] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur. XORing elephants: Novel erasure codes for big data. *arXiv:1301.3791*, 2013.

[46] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran. Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff. *IEEE Transactions on Information Theory*, 58(3):1837–1852, 2012.

[47] N. B. Shah, K. V. Rashmi, and V. Kumar. A flexible class of regenerating codes for distributed storage. In *Proc. IEEE International Symposium on Information Theory (Austin, TX)*, pages 1943–1947, 2010.

[48] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath. Optimal locally repairable codes via rank-metric codes. In *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1819–1823, 2013.

[49] W. Song, S. H. Dau, C. Yuen, and T. J. Li. Optimal locally repairable linear codes. *arXiv:1307.1961 [cs.IT]*, 2013.

[50] C. Suh and K. Ramchandran. On the existence of optimal exact-repair MDS codes for distributed storage. *arXiv:1004.4663v1 [cs.IT]*, 2010.

[51] I. Tamo and A. Barg. A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory*, 60(8):4661–4676, 2014.

[52] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis. Optimal locally repairable codes and connections to matroid theory. In *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1814–1818, 2013.

[53] I. Tamo, Z. Wang, and J. Bruck. MDS array codes with optimal rebuilding. In *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1240–1244, 2011.

[54] I. Tamo, Z. Wang, and J. Bruck. Zigzag codes: MDS array codes with optimal rebuilding. *IEEE Transactions on Information Theory*, 59(3):1597–1616, 2013.

[55] E. K. Thomas and F. Oggier. Explicit constructions of quasi-uniform codes from groups. In *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 489–493, 2013.

[56] C. Tian. Rate region of the (4,3,3) exact-repair regenerating codes. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 1426–1430, 2013.

[57] C. Tian, V. Aggarwal, and V. A. Vaishampayan. Exact-repair regenerating codes via layered erasure correction and block designs. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 1431–1435, 2013.

[58] V. T. Van, C. Yuen, and J. Li. Non-homogeneous distributed storage systems. *arXiv:1208.2078*, 2012.

[59] J.H. van Lint. *Introduction to Coding Theory*. Springer, third edition, 1999.

[60] Y. Wu. A construction of systematic MDS codes with minimum repair bandwidth. *arXiv:0910.2486*, 2009.

[61] Y. Wu. Existence and construction of capacity-achieving network codes for distributed storage. *IEEE Journal on Selected Areas in Communications*, 28(2):277–288, 2010.

[62] Y. Wu and A. G. Dimakis. Reducing repair traffic for erasure coding-based storage via interference alignment. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 2276–2280, 2009.

[63] H. Zhang, M. Chen, A. Parek, and K. Ramchandran. A distributed multichannel demand-adaptive p2p VoD system with optimized caching and neighbor-selection. In *Proceedings of SPIE (San Diego, CA)*, 2011.

Publication I

T. Ernvall, S. El Rouayheb, C. Hollanti, H. V. Poor, Capacity and Security of Heterogeneous Distributed Storage Systems, *IEEE Journal on Selected Areas in Communications*, vol. 31 (12) 2013, pp. 2701-2709.

Publication II

T. Ernvall, Codes between MBR and MSR Points with Exact Repair Property, *IEEE Transactions on Information Theory*, vol. 60 (11) 2014, pp. 6993-7005.

Publication III

T. Ernvall, T. Westerbäck, C. Hollanti, R. Freij, Constructions and Properties of Linear Locally Repairable Codes, *IEEE Transactions on Information Theory*, submitted 2014. Available: arXiv:1410.6339 [cs.IT].

Turku Centre for Computer Science
TUCS Dissertations

1. **Marjo Lipponen**, On Primitive Solutions of the Post Correspondence Problem
2. **Timo Käkölä**, Dual Information Systems in Hyperknowledge Organizations
3. **Ville Leppänen**, Studies on the Realization of PRAM
4. **Cunsheng Ding**, Cryptographic Counter Generators
5. **Sami Viitanen**, Some New Global Optimization Algorithms
6. **Tapio Salakoski**, Representative Classification of Protein Structures
7. **Thomas Långbacka**, An Interactive Environment Supporting the Development of Formally Correct Programs
8. **Thomas Finne**, A Decision Support System for Improving Information Security
9. **Valeria Mihalache**, Cooperation, Communication, Control. Investigations on Grammar Systems.
10. **Marina Waldén**, Formal Reasoning About Distributed Algorithms
11. **Tero Laihonen**, Estimates on the Covering Radius When the Dual Distance is Known
12. **Lucian Ilie**, Decision Problems on Orders of Words
13. **Jukkapekka Hekanaho**, An Evolutionary Approach to Concept Learning
14. **Jouni Järvinen**, Knowledge Representation and Rough Sets
15. **Tomi Pasanen**, In-Place Algorithms for Sorting Problems
16. **Mika Johnsson**, Operational and Tactical Level Optimization in Printed Circuit Board Assembly
17. **Mats Aspnäs**, Multiprocessor Architecture and Programming: The Hathi-2 System
18. **Anna Mikhajlova**, Ensuring Correctness of Object and Component Systems
19. **Vesa Torvinen**, Construction and Evaluation of the Labour Game Method
20. **Jorma Boberg**, Cluster Analysis. A Mathematical Approach with Applications to Protein Structures
21. **Leonid Mikhajlov**, Software Reuse Mechanisms and Techniques: Safety Versus Flexibility
22. **Timo Kaukoranta**, Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization
23. **Gábor Magyar**, On Solution Approaches for Some Industrially Motivated Combinatorial Optimization Problems
24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming – C++ is more Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-Disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Market Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Frank Tétard**, Managers, Fragmentation of Working Time, and Information Systems

41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations Between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity – A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-Type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images
63. **Tommi Meskanen,** On the NTRU Cryptosystem
64. **Saeed Salehi**, Varieties of Tree Languages
65. **Jukka Arvo**, Efficient Algorithms for Hardware-Accelerated Shadow Computation
66. **Mika Hirvikorpi**, On the Tactical Level Production Planning in Flexible Manufacturing Systems
67. **Adrian Costea**, Computational Intelligence Methods for Quantitative Data Mining
68. **Cristina Seceleanu**, A Methodology for Constructing Correct Reactive Systems
69. **Luigia Petre**, Modeling with Action Systems
70. **Lu Yan**, Systematic Design of Ubiquitous Systems
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory

86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming

128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations
132. **Riikka Vuokko**, A Practice Perspective on Organizational Implementation of Information Technology
133. **Jeanette Heidenberg**, Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches
134. **Yong Liu**, Solving the Puzzle of Mobile Learning Adoption
135. **Stina Ojala**, Towards an Integrative Information Society: Studies on Individuality in Speech and Sign
136. **Matteo Brunelli**, Some Advances in Mathematical Models for Preference Relations
137. **Ville Junnila**, On Identifying and Locating-Dominating Codes
138. **Andrzej Mizera**, Methods for Construction and Analysis of Computational Models in Systems Biology. Applications to the Modelling of the Heat Shock Response and the Self-Assembly of Intermediate Filaments.
139. **Csaba Ráduly-Baka**, Algorithmic Solutions for Combinatorial Problems in Resource Management of Manufacturing Environments
140. **Jari Kyngäs**, Solving Challenging Real-World Scheduling Problems
141. **Arho Suominen**, Notes on Emerging Technologies
142. **József Mezei**, A Quantitative View on Fuzzy Numbers
143. **Marta Olszewska**, On the Impact of Rigorous Approaches on the Quality of Development
144. **Antti Airola**, Kernel-Based Ranking: Methods for Learning and Performace Estimation
145. **Aleksi Saarela**, Word Equations and Related Topics: Independence, Decidability and Characterizations
146. **Lasse Bergroth**, Kahden merkkijonon pisimmän yhteisen alijonon ongelma ja sen ratkaiseminen
147. **Thomas Canhao Xu**, Hardware/Software Co-Design for Multicore Architectures
148. **Tuomas Mäkilä**, Software Development Process Modeling – Developers Perspective to Contemporary Modeling Techniques
149. **Shahrokh Nikou**, Opening the Black-Box of IT Artifacts: Looking into Mobile Service Characteristics and Individual Perception
150. **Alessandro Buoni**, Fraud Detection in the Banking Sector: A Multi-Agent Approach
151. **Mats Neovius**, Trustworthy Context Dependency in Ubiquitous Systems
152. **Fredrik Degerlund**, Scheduling of Guarded Command Based Models
153. **Amir-Mohammad Rahmani-Sane**, Exploration and Design of Power-Efficient Networked Many-Core Systems
154. **Ville Rantala**, On Dynamic Monitoring Methods for Networks-on-Chip
155. **Mikko Pelto**, On Identifying and Locating-Dominating Codes in the Infinite King Grid
156. **Anton Tarasyuk**, Formal Development and Quantitative Verification of Dependable Systems
157. **Muhammad Mohsin Saleemi**, Towards Combining Interactive Mobile TV and Smart Spaces: Architectures, Tools and Application Development
158. **Tommi J. M. Lehtinen**, Numbers and Languages
159. **Peter Sarlin**, Mapping Financial Stability
160. **Alexander Wei Yin**, On Energy Efficient Computing Platforms
161. **Mikołaj Olszewski**, Scaling Up Stepwise Feature Introduction to Construction of Large Software Systems
162. **Maryam Kamali**, Reusable Formal Architectures for Networked Systems
163. **Zhiyuan Yao**, Visual Customer Segmentation and Behavior Analysis – A SOM-Based Approach
164. **Timo Jolivet**, Combinatorics of Pisot Substitutions
165. **Rajeev Kumar Kanth**, Analysis and Life Cycle Assessment of Printed Antennas for Sustainable Wireless Systems
166. **Khalid Latif**, Design Space Exploration for MPSoC Architectures

167. **Bo Yang**, Towards Optimal Application Mapping for Energy-Efficient Many-Core Platforms
168. **Ali Hanzala Khan**, Consistency of UML Based Designs Using Ontology Reasoners
169. **Sonja Leskinen**, m-Equine: IS Support for the Horse Industry
170. **Fareed Ahmed Jokhio**, Video Transcoding in a Distributed Cloud Computing Environment
171. **Moazzam Fareed Niazi**, A Model-Based Development and Verification Framework for Distributed System-on-Chip Architecture
172. **Mari Huova**, Combinatorics on Words: New Aspects on Avoidability, Defect Effect, Equations and Palindromes
173. **Ville Timonen**, Scalable Algorithms for Height Field Illumination
174. **Henri Korvela**, Virtual Communities – A Virtual Treasure Trove for End-User Developers
175. **Kameswar Rao Vaddina**, Thermal-Aware Networked Many-Core Systems
176. **Janne Lahtiranta**, New and Emerging Challenges of the ICT-Mediated Health and Well-Being Services
177. **Irum Rauf**, Design and Validation of Stateful Composite RESTful Web Services
178. **Jari Björne**, Biomedical Event Extraction with Machine Learning
179. **Katri Haverinen**, Natural Language Processing Resources for Finnish: Corpus Development in the General and Clinical Domains
180. **Ville Salo**, Subshifts with Simple Cellular Automata
181. **Johan Ersfolk**, Scheduling Dynamic Dataflow Graphs
182. **Hongyan Liu**, On Advancing Business Intelligence in the Electricity Retail Market
183. **Adnan Ashraf**, Cost-Efficient Virtual Machine Management: Provisioning, Admission Control, and Consolidation
184. **Muhammad Nazrul Islam**, Design and Evaluation of Web Interface Signs to Improve Web Usability: A Semiotic Framework
185. **Johannes Tuikkala**, Algorithmic Techniques in Gene Expression Processing: From Imputation to Visualization
186. **Natalia Díaz Rodríguez**, Semantic and Fuzzy Modelling for Human Behaviour Recognition in Smart Spaces. A Case Study on Ambient Assisted Living
187. **Mikko Pänkäälä**, Potential and Challenges of Analog Reconfigurable Computation in Modern and Future CMOS
188. **Sami Hyrynsalmi**, Letters from the War of Ecosystems – An Analysis of Independent Software Vendors in Mobile Application Marketplaces
189. **Seppo Pulkkinen**, Efficient Optimization Algorithms for Nonlinear Data Analysis
190. **Sami Pyöttiälä**, Optimization and Measuring Techniques for Collect-and-Place Machines in Printed Circuit Board Industry
191. **Syed Mohammad Asad Hassan Jafri**, Virtual Runtime Application Partitions for Resource Management in Massively Parallel Architectures
192. **Toni Ernvall**, On Distributed Storage Codes

Turku Centre *for* Computer Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www. tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Division for Natural Sciences and Technology
- Department of Information Technologies

ISBN 978-952-12-3166-7
ISSN 1239-1883

Toni Ernvall On Distributed Storage Codes

Toni Ernvall On Distributed Storage Codes

Toni Ernvall On Distributed Storage Codes