Finding the Maximum Cardinalities of Condorcet Domains in Voting Systems

Background

The problem of rank aggregation appears in many contexts, such as social choice decisions, web search algorithms, and voting theory. The typical paradigm in voting theory involves n voters and m candidates. Every voter ranks the candidates, which results in a permutation of the m candidates, that is, a permutation σ in $\mathbb{S}_m$, the set of all $m!$ permutations. The vector of all rankings is denoted by $\Sigma = (\sigma_1, \ldots, \sigma_n) \in (\mathbb{S}_m)^n$ and is called a ***profile***. For example, assume there are three candidates $1, 2, 3$ and five voters, then two examples of profiles are $\Sigma_1 = (123, 321, 213, 312, 123)$ and $\Sigma_2 = (123, 231, 312, 123, 321)$.

A key problem in voting is to derive the aggregate result of the voting and choose a winner. A popular method for vote aggregation is based on the ***Condorcet criterion***. The ***Condorcet winner*** is the candidate who wins every other candidate by pairwise majority. However, the main disadvantage of this approach, known as the ***Condorcet paradox***, is that such a winner does not necessarily exist, since this criterion does not necessarily admit transitivity. For example, the first candidate in the profile Σ_1 is the Condorcet winner as he wins the other two candidates, however the profile Σ_2 does not have a Condorcet winner since the first candidate wins the second, the second wins the third and lastly, the third wins the first.

A ***Condorcet domain*** is a set $S \subseteq \mathbb{S}_m$ of permutations such that every profile $\Sigma \in \mathbb{S}^n$ has a Condorcet winner. For example, for three candidates it is possible to verify that $S = \{123, 132, 213, 312\}$ is a Condorcet domain. The problem of finding the largest size of a Condorcet domain is a fascinating problem which, even though it was studied for a long time, is still an *open problem* for arbitrary m.

Objectives

The main goal in this project is to find the maximum cardinalities of Condorcet domains. If $f(m)$ denotes the maximum cardinality of a Condorcet domain with m candidates, then it was found that $f(3) = 4, f(4) = 9, f(5) = 20, f(6) = 45, f(7) \geq 100, f(8) \geq 222$. Our aim is to shed more light into these findings and derive results for $f(7)$ and $f(8)$ and/or any other findings.

Methods to meet objectives

This project will first require the understanding of some of the existing results in the area of Condorcet domain. In particular, understanding the criteria to check whether a set $S \subseteq \mathbb{S}_m$ is a Condorcet domain. According to these findings we will build a computer program that will carefully check and look for Condorcet domains, with the intention of deriving specific results for $f(7)$ and $f(8)$.

Error Behavior and Characterization of Flash Memories

Background

Flash memory has become the storage medium of choice in portable consumer electronic applications, and high performance solid state drives (SSDs) are also being introduced into mobile computing, enterprise storage, data warehousing, and data-intensive computing systems. On the other hand, flash memory technologies present major challenges in the areas of device reliability, endurance, and energy efficiency.

Flash memory chip is built from floating-gate cells which are organized in blocks. Every cell can store one bit, known as single-level cell (SLC), or multiple bits, known as multi-level cell (MLC). Each block typically contains between 64 pages and 384 pages, where the size of a page can range between 2KB and 8KB.

Error-correcting codes (ECCs) such as BCH and LDPC codes have typically been used in commercial flash memories in order to correct cell errors in the memory. However, the ever-shrinking feature size in flash memory devices and the recent introduction of multi-level flash technology has created the need for more powerful ECC methods. This task requires the comprehensive understanding of the error mechanisms and error characteristics of flash memory chips, which will be the main focus of this project.

Objectives

The main goal in this project is to enhance the understanding of the physical channel model of flash memories, which serves as the basis for signal processing and coding algorithms. The objective will be specifically focused on the effect of inter-cell and inter-page interference phenomena during program and read operations.

Methods to meet objectives

We will use in this project an existing extensive empirical database of errors, which were observed during erase, program, and read operations on different flash memory chips. A key objective in these experiments is to mimic the degradation in the reliability of flash memory devices. This is done by iteration of the process of erasing a flash memory block, then writing pseudo-random data into it, and finally reading it in order to compare and find errors.

We will build upon existing work which analyzed the recorded database of errors on the block and page levels and gave a characterization of the observed error types. The main focus of the analysis in this project will be dedicated towards understanding the error behavior in the cell level, with particular attention to the data dependent nature of the errors arising from inter-cell interference between different cells.

Implementation of WOM Codes in Flash Memories

Background

Write-Once-Memory (WOM) codes were first introduced in 1982 by Rivest and Shamir. They make it possible to record information more than once in a so-called write-once storage medium. These media can be represented as a collection of binary cells, each of which initially represents a bit value 0 that can be irreversibly overwritten with a bit value 1. A WOM code allows the reuse of the cells and write information mutiple times without violating the asymmetry writing constraint.

A simple example, presented by Rivest and Shamir in Table 1, enables the recording of two bits of information in three cells twice. It is possible to verify that after the first 2-bit data vector is encoded into a 3-bit codeword, if the second 2-bit data vector is different from the first, the 3-bit codeword into which it is encoded does not change any code bit 1 into a code bit 0, ensuring that it can be recorded in the write-once medium.

Table 1: A WOM code Example

Data bits	First write	Second write
00	000	111
10	100	011
01	010	101
11	001	110

Flash memories impose programming constraints that are similar to write-once memories. Flash memories contain floating gate cells which are electrically charged with electrons to represent the cell level. While it is fast and simple to increase a cell level, reducing its level requires a long and cumbersome operation of first erasing its entire containing block and only then programming the cells. Such block erasures are not only time consuming, but also degrade the lifetime of the memory. A WOM code can be applied in this context to enable additional writes without first having to physically erase the entire block. The cost associated with this increase in the endurance is the redundancy and the additional complexity associated with the encoding and decoding processes.

Objectives

The focus in this project will be dedicated to efficient algorithms for the implementation of WOM codes in flash memory chips. Our main goal is to show that the proposed algorithms achieve better performance when comparing with the current existing ones.

Methods to meet objectives

This project will first require the understanding of the operation, advantages, and constrains of WOM codes. Then, we plan to study approaches that avoid the capacity lost when implementing WOM codes in flash memories. In particular, we will investigate codes where flash memory cells will be written twice. On the first write all pages will be written and on the second write we will combine between several physical pages in order to write a logical page.