THE STRUCTURE AND STATISTICS OF REED–MULLER CODES

Allocation: Exploratory/20 Knh
PI: Iwan Duursma[1]
Collaborator: Hsin–Po Wang

[1]University of Illinois at Urbana–Champaign

MI

FS

EXECUTIVE SUMMARY

This project focuses on computing the Tutte polynomial of the Reed–Muller (RM) codes of block length 64. RM codes are among the earliest linear block codes whose construction is easy to describe, and yet their properties have not been fully understood. They are related to polar codes, a class of codes that was introduced in the last 10 years and that is known to have good asymptotic properties. It is conjectured that RM codes also share those properties. The Tutte polynomial is an invariant of linear block codes that captures the probability of error when correcting erasures in codewords. The Tutte polynomial is used more generally in graph theory, knot theory, and physics.

RESEARCH CHALLENGE

The Tutte polynomial is a two-variable generating function for the corank-nullity distribution over all subsets of columns of a given matrix. It can be computed exhaustively or recursively using a sequence of deletion–restriction operations. It is known that computing the Tutte polynomial for general block codes is sharp-P hard (the quantitative version of nonpolinomial hardness). For Reed–Muller codes, Tutte polynomials are known up to length 32. The research team extended this to length 64 and, moreover, computed a newly defined invariant that provides more detailed information than the Tutte polynomial.

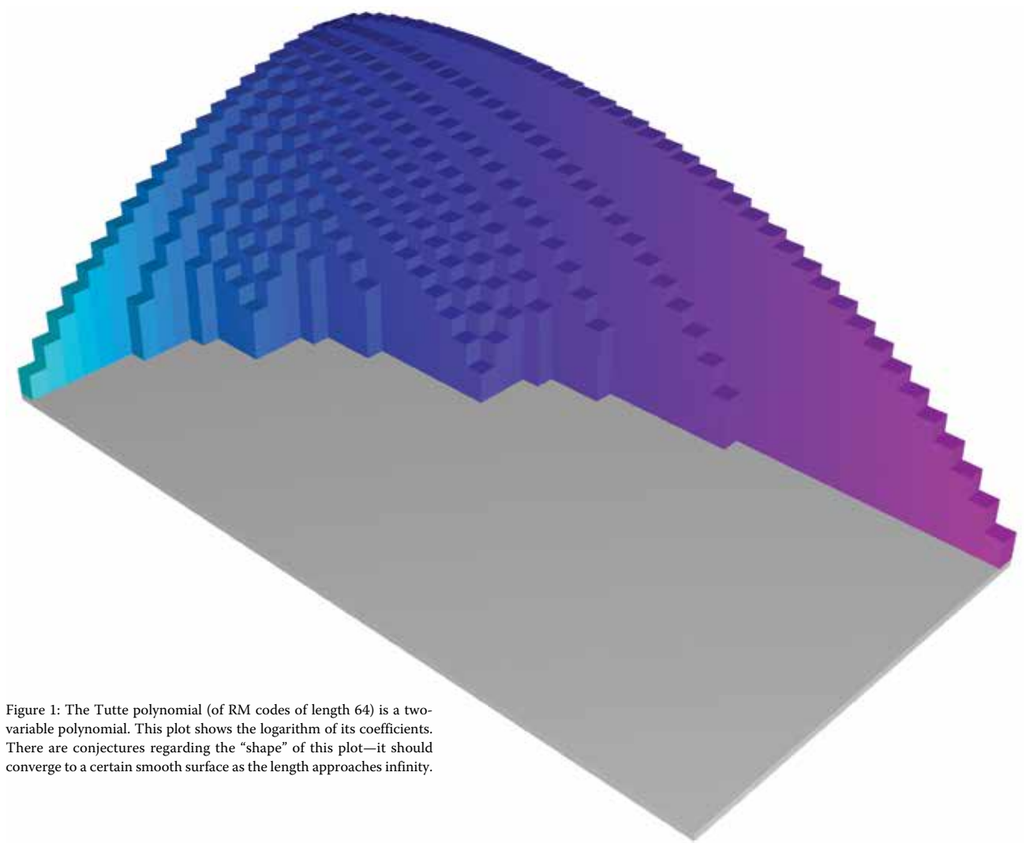

Figure 1: The Tutte polynomial (of RM codes of length 64) is a two-variable polynomial. This plot shows the logarithm of its coefficients. There are conjectures regarding the "shape" of this plot—it should converge to a certain smooth surface as the length approaches infinity.

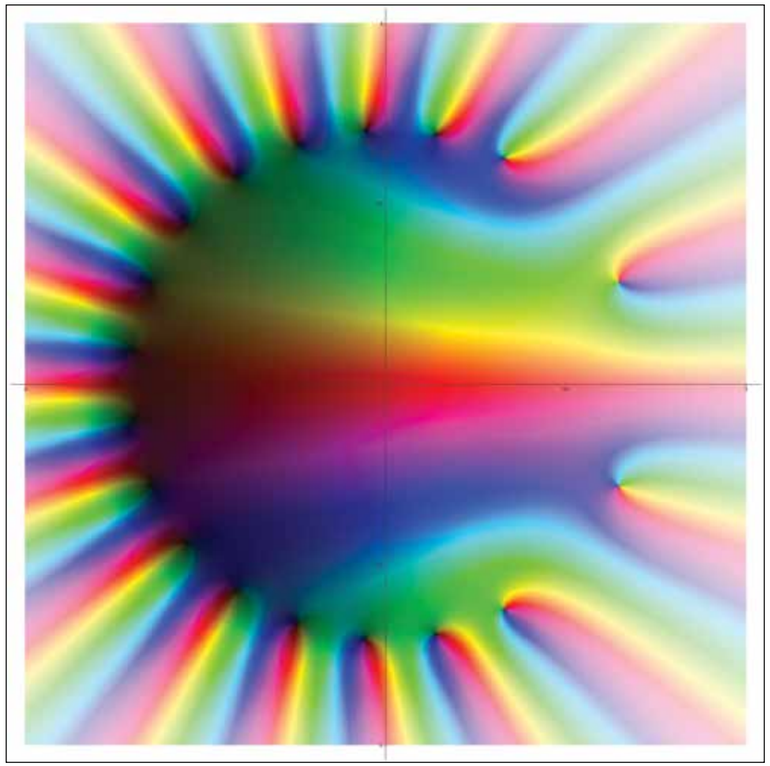

Figure 2: The zeta function of the [32,16,8] RM code using domain coloring. The zeta function in coding theory is an analog to the Riemann zeta function and the Hasse–Weil zeta function. It can be seen that the zeros all lie on a circle: the "critical line."

METHODS & CODES

The reduction that essentially brings this problem within a feasible scope is the Plotkin structure of RM codes. The fact that the construction of RM codes is recursive suggests that an invariant can be computed along the recursion. However, this fact only reduces the time cost by a factor of 60,000. The remaining work still costs Blue Waters 2,000 node-hours to complete. All the codes employed were produced by the team.

RESULTS & IMPACT

The researchers computed the Tutte polynomial along with extra data that allow followers to double-check whether the computation is correct. The exact polynomial may be found at https://github.com/Symbol1/BlueWaters-RM64/blob/master/rm64tutte.txt. The extra data with a full explanation of how they were computed and how they can be verified may be found at https://github.com/Symbol1/BlueWaters-RM64.

WHY BLUE WATERS

The role of Blue Waters is twofold. First, this computation is CPU-intensive, with a great deal of conditional branching and bitwise operators, and Blue Waters provides a parallel context to accelerate the computation using all 32 cores in a node. Second, the team generated an abundance of intermediate data that should be kept in RAM to avoid the hard disk's I/O, because otherwise the I/O becomes the bottleneck. The team expected more, but the given 64 GB of memory per node barely fits. (The researchers tailored the MPI reduce function to save space.) In addition, the computation does not write any error-handling codes, but apparently Blue Waters was all green when this job was running, thanks to its sustainable design.

PUBLICATIONS & DATA SETS

Computation of Reed–Muller codes of length 64 on Blue Waters. [Online]. Available: https://github.com/Symbol1/BlueWaters-RM64