On the Football Pool Problem

Jeff Linderoth* François Margot† Greg Thain‡

Dept. of Industrial and Sys. Eng., University of Wisconsin, Madison, WI, USA.
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA.
Computer Sciences Dept., University of Wisconsin, Madison, WI, USA.

The Football Pool Problem is one of the most famous problems in coding theory. The problem derives its name from a lottery-type game where participants predict the outcome of soccer matches. A ticket is winning if the outcome of no more than d matches out of v are predicted incorrectly, where each match has three possible outcomes—win, lose, or draw. The goal of the Football Pool Problem is to determine the minimum-size set of lottery tickets to buy to guarantee at least one winning ticket, no matter the outcome of the matches. Mathematically, the problem is to determine the smallest covering code of radius d of ternary words of length v. For $d = 1$ and $v = 6$, the optimal code size is only known to be between the values of 65 and 73. In this work, we report experiences on using enumerative techniques in combination with integer programming as part of a large-scale computation aimed at sharpening these bounds.

An integer linear program (ILP) can be written to determine the smallest covering code, using one variable for each possible code word. This yields a symmetric ILP with 729 variables and 729 constraints extremely difficult to solve. To mitigate the difficulties associated with symmetry, we use an isomorphism pruning technique that allows for a drastic reduction in the size of the enumeration tree, combined with several enumerative tools in order to get a manageable collection of simpler ILPs.

One technique we use was introduced by Blass and Östergård. It amounts to enumerate the set S of all nonisomorphic integer solutions for a projection of the feasible set of the ILP, and then, for each $s \in S$, solve an ILP simpler than the original one. We use three additional procedures to ease the burden of solving the many ILP required of this technique: reordering, regrouping, and filtering. First, by reordering the components of a solution s before using them in the ILP, the solution time of the latter is often significantly reduced. Second, recognizing that some solutions in S are very similar to each other, regrouping them and solving a modified ILP for each group of sequences is advantageous. Finally, we use a preprocessing operation that can prove that some $s \in S$ cannot lead to a code of cardinality smaller than 73.

Our computational grid is built using the `Condor` software system for high-throughput computing and the grid-computing toolkit and ran on 19'000 computers on the Teragrid and the Open Science Grid. To date, we have been able to establish a new lower bound of $z_6 \geq 71$. To exclude a solution of value 69, we used an average of 555 workers (2038 max) and 110.1 CPU years. For 70, we used an average of 562 workers (1175 max) and 30.3 CPU years. For these two portions of the computation, over 140 CPU years were delivered by grid resources in roughly 92 days. The total number of nodes numbers in the billions, and required *trillions* of LP pivots. To our knowledge, this is the largest branch-and-bound computation ever run on a wide-area grid. By our best estimation, we are roughly 50% through the computation for 71 and hope to be able to announce that $z_6 \geq 72$ soon.

*Supported by National Science Foundations grants CMMI-0522796 and OCI-0330607 and Department of Energy grant DE-FG02-05ER25694. E-mail: `linderot@cs.wisc.edu`

†Supported by Office of Naval Research grant N00014-03-1-0188. E-mail: `fmargot@andrew.cmu.edu`

‡Supported by National Science Foundation grant OCI-0330607. E-mail: `gthain@cs.wisc.edu`