Graph Compression in a Distributed Memory Environment

Cleve Ashcraft, LSTC * Roger Grimes, LSTC †

March 20, 2013

Abstract

LSTC produces the multiphysics simulation package LS-DYNA. Internally, we need to solve large real symmetric sparse linear systems $K\,u = f$ in a distributed memory parallel environment. At present, our typical large stiffness matrices K have 60 million rows and columns, and this size is growing rapidly.

In our current production environment we perform a graph compression of the matrix K to get a compressed graph that is usually 3 to 6 times fewer vertices and 9 or 36 times fewer edges. We then order the compressed graph in some fashion, and uncompress the ordering to the original graph.

For the past twenty years, our algorithm to perform this graph compression is as follows. We start with a with a strict partition of the lower triangle of K across processes. We generate three stimulus vectors, each filled with a random permutation of $\{1, \cdots, n\}$. We take the adjacency matrix of K, (same nonzero structure and filled with one's), and multiply the stimulus vectors to form three response vectors. Two vertices are equivalent if they have the same response values in their rows for each of the response vectors.

This algorithm suffers from overflow on today's larger matrices, particularly where (partially) dense rows are present, or large nonzero blocks, e.g., from superelements.

Multiplying by the adjacency matrix is equivalent to summing the labels of adjacent vertices of a given vertex. In our new implementation we replace the label summation by "label XOR". A stimulus vector is a bit pattern of a certain length, and to get a response value, we XOR the bit patterns of the adjacent vertices. Two vertices are equivalent if they have the same values in their rows for each response vector.

False positives are possible, where we incorrectly recognize two vertices as indistinguishable if their response values are the same. We iterate the process (create stimulus vector, compute response vector) until there is no change in the equivalence map. There is an interesting relationship between the width of the bitmap and the number of false positives.

A robust distributed memory implementation is a challenge. At the heart of the algorithm is a matrix-vector multiply (with XOR instead of addition). We cannot rely on any pre-existing domain decomposition to order the degrees of freedom to reduce communication.

*Livermore Software Technology Corporation, 7374 Las Positas Road, Livermore, CA 94550. cleve@lstc.com

†Livermore Software Technology Corporation, 7374 Las Positas Road, Livermore, CA 94550. grimes@lstc.com