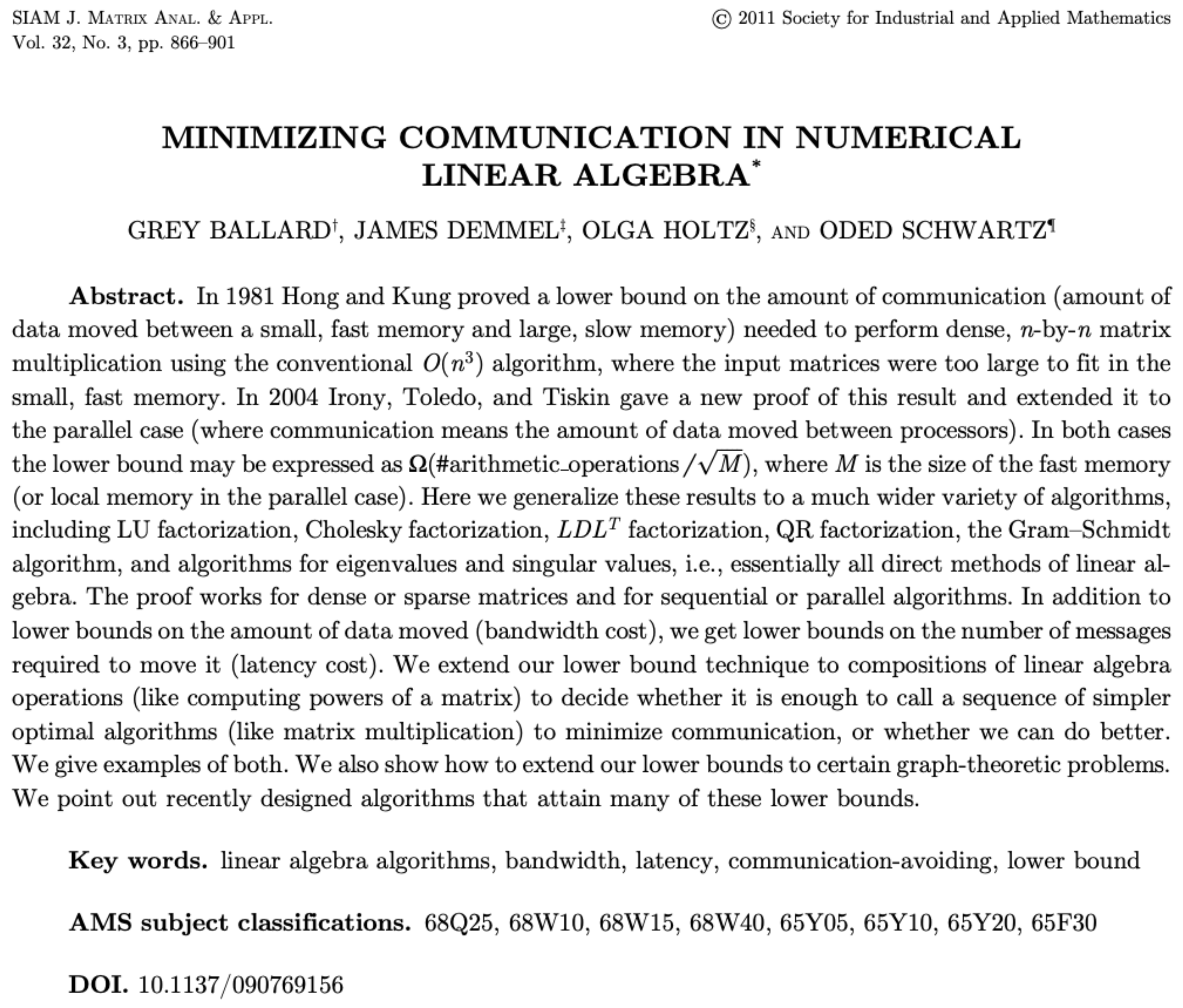
Reading and Thinking



SIAM J. MATRIX ANAL. & APPL.
Vol. 32, No. 3, pp. 866–901

MINIMIZING COMMUNICATION IN NUMERICAL LINEAR ALGEBRA*

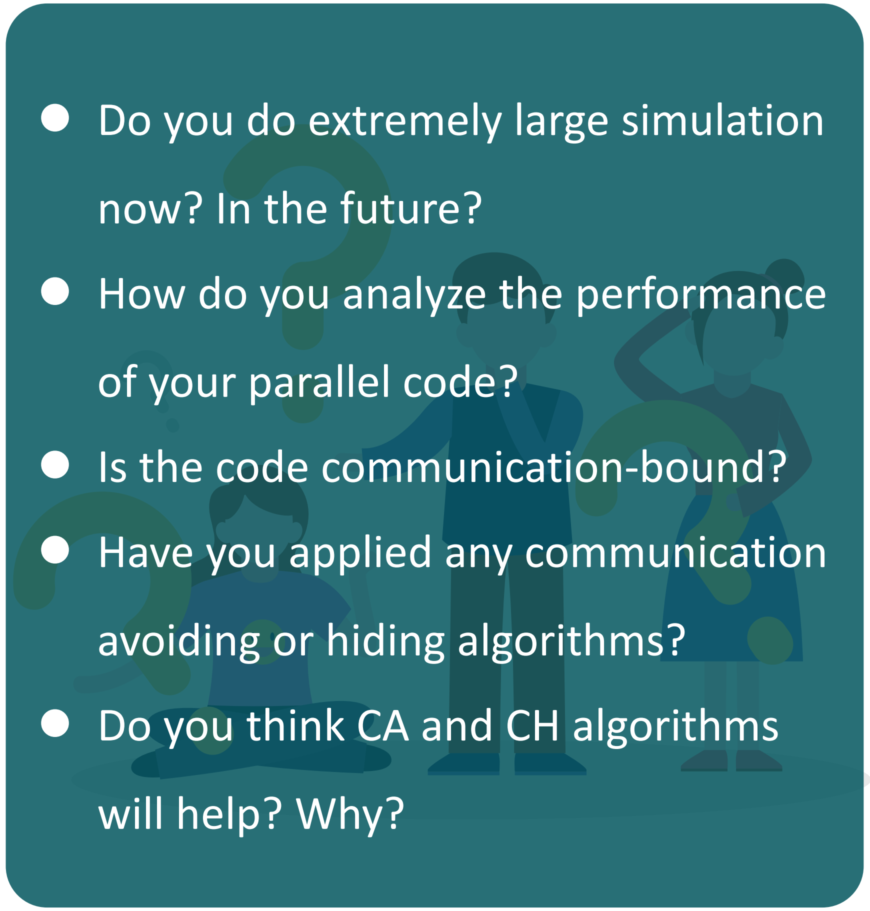
GREY BALLARD†, JAMES DEMMEL‡, OLGA HOLTZ§, AND ODED SCHWARTZ¶

Abstract. In 1981 Hong and Kung proved a lower bound on the amount of communication (amount of data moved between a small, fast memory and large, slow memory) needed to perform dense, n-by-n matrix multiplication using the conventional $O(n^3)$ algorithm, where the input matrices were too large to fit in the small, fast memory. In 2004 Irony, Toledo, and Tiskin gave a new proof of this result and extended it to the parallel case (where communication means the amount of data moved between processors). In both cases the lower bound may be expressed as $\Omega(\#\text{arithmetic\_operations}/\sqrt{M})$, where M is the size of the fast memory (or local memory in the parallel case). Here we generalize these results to a much wider variety of algorithms, including LU factorization, Cholesky factorization, LDL^T factorization, QR factorization, the Gram–Schmidt algorithm, and algorithms for eigenvalues and singular values, i.e., essentially all direct methods of linear algebra. The proof works for dense or sparse matrices and for sequential or parallel algorithms. In addition to lower bounds on the amount of data moved (bandwidth cost), we get lower bounds on the number of messages required to move it (latency cost). We extend our lower bound technique to compositions of linear algebra operations (like computing powers of a matrix) to decide whether it is enough to call a sequence of simpler optimal algorithms (like matrix multiplication) to minimize communication, or whether we can do better. We give examples of both. We also show how to extend our lower bounds to certain graph-theoretic problems. We point out recently designed algorithms that attain many of these lower bounds.

Key words. linear algebra algorithms, bandwidth, latency, communication-avoiding, lower bound

AMS subject classifications. 68Q25, 68W10, 68W15, 68W40, 65Y05, 65Y10, 65Y20, 65F30

DOI. 10.1137/090769156



- Do you do extremely large simulation now? In the future?
- How do you analyze the performance of your parallel code?
- Is the code communication-bound?
- Have you applied any communication avoiding or hiding algorithms?
- Do you think CA and CH algorithms will help? Why?