The Expand-Maximize-Compress (EMC) Algorithm for Sparse Data Analysis in Serial Crystallography: Progress on a Rust-Based Software Package

Aaron D. Finke

Max IV Laboratory, Lund University, Fotongatan 2, 224 84 Lund, Sweden
aaron.finke@maxiv.lu.se

Keywords: Serial Crystallography, EMC Algorithm, Sparse Data

The Expand-Maximize-Compress (EMC) algorithm [1] is a method that attempts to reconstruct reciprocal space models based on noisy, “sparse” diffraction data, in which few photons on a detector frame are from diffraction. Given a set of noisy diffraction patterns $K$, with unmeasured particle orientations $W$, the EMC algorithm attempts to build a 3D model $W$ by iteratively maximizing the data likelihood function, $p(K|W)$. Since this is difficult in the absence of measured $W$, the algorithm provides a way to reconstruct $W$ by alternating the estimates of $W$ and $W$ by fixing the values of one or the other until convergence.[2] This method has proven utility in tomographic and imaging applications,[3] and more recently, in crystallographic applications.[4-5] Specifically, in crystallographic methods such as serial crystallography, where each data frame corresponds with the diffraction pattern of a randomly oriented crystal, EMC can reconstruct the reciprocal space of the crystal system even when the data is sparse.[6]

An EMC software package to both set up the data and perform the EMC calculations was written in C and Python [7], with heavy emphasis on parallel computing with MPI. However, set up and use of this software is difficult, and the large memory and processing requirements for standard EMC computation for serial crystallography led to unforeseen crashes due to memory leaks. In lieu of fixing these in C, an alternative approach was developed: rewriting the compiled parts of the software package in the Rust language, a modern, compiled, low-level language with emphasis on memory safety and thread safety. Rust’s paradigm of data “ownership” ensures memory safety, which is critical to a complex, highly compute-intensive method such as EMC. I will present recent advances in redeployment of this software, toward a goal of wide-scale deployment.


The author wishes to thank Jeney Wierman and Marian Szebenyi, MACCHESS, Cornell University for helpful discuss