Runtime and Correctness Approximations

Competitive analysis of on-line algorithms is an amazing runtime approximation analysis (and design) method. As with correctness approximations, the idea is to “embed” OPT and then get an upper bound on the difference to OPT, in most cases within a constant. In correctness approximations, the correctness proof often gives a constant. In competitive analysis, given a potential function for amortized cost, the potential difference can accumulate an upper bound on runtime difference to OPT. Competitive analysis is increasingly applied to the randomized case.

Matrix Factorization as Effective Representation of Independencies

Matrix factorization (fft, multiplication, computing svd, eigenvalues...) has been and will likely continue to be a key part of the most impactful DC/DP algorithms. If factorization can provide an effective representation of independencies, I am curious about randomization. Given inexpensive independencies, randomization can sometimes lead to gains in time/space (e.g. expectation bounds from random variable recurrences) with correctness guarantees. It is often practical to fix time/space or correctness and get an expectation bound on the other, based on independencies, and then use independencies to reach the bound.