Parallelism as a “Trade-off” Dimension in Algorithm Design (6/28/2019)

Parallelism can be meaningfully considered as a "trade-off" dimension in addition to time, space, and correctness dimensions in algorithm design.

Parallelism provides time speedup, asymptotically bounded by input size. Having reserved enough parallelism in an algorithm design, exceeding practical settings at a large input size, it can be traded against space by redesigning the algorithm while keeping the actual speedup.

An expectation bound of a randomized algorithm or data structure is often reached by rerunning/reconstructing. A subroutine in a randomized algorithm can also be a (unfair) coin flip designed to be rerun (e.g. primality testing). Parallelism provides coin flip “scale-up”.

Design with Expectation Properties as Analysis Blueprint (6/24/2019)

Arriving at an expectation recurrence, practically solvable by substitution, is often central to analyzing randomized algorithms and data structures. Arguably, the richness of expectation properties is why a loose expectation bound can often be a low-hanging fruit. Considering the properties of expectation during design may provide a blueprint for analysis. A few examples:

i) Expressing “in algorithm/data structure” choices as indicator r.v.s:
whether a path is chosen is independent of random choices along the path, thus expectation of product = product of expectations.

ii) Taking max over r.v.s:
a) look for symmetry to simplify, b) otherwise summing the terms in max provides a potentially loose upper bound but with the benefit of linearity of expectation.

iii) Trying different analysis tracks by transforming sum into product or product into sum:
exponential (convex) or log (concave) functions maintain inequality throughout analysis by Jensen inequality.

Runtime and Correctness Approximations (6/18/2019)

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 (5/24/2019)

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.