

SOME RESEARCH TOPICS FOR STUDENTS

Quentin F. Stout

Computer Science and Engineering, University of Michigan

qstout@umich.edu

Here's a quick overview of some research topics for students. I've also worked with many students that initiated their own project. If you have something interesting to pursue come and discuss it. I've emphasized PhD thesis topics below, but for some of the areas there are undergraduate research projects available. Note: two of my students, Russ Miller and Philip MacKenzie, won awards for *Best Thesis* within the entire university.

Much of the research we do involves PARALLEL COMPUTING. I became interested in this when I read descriptions of how the mind works and the inherent parallelism. I started thinking about how using many computers working together required fundamentally new approaches. Beyond intellectual curiosity, there are critical practical reasons for studying parallel computing:

1. People continually need to solve harder problems: vastly more data, more complexity, faster response, ...

Historically people solved 1) by better algorithms and the fact that computers kept getting faster. However,

2. Suddenly, due to pesky physical limits, computers aren't getting any faster.

People continue to develop better algorithms, but in some areas this is getting diminishing returns. For example, a new algorithm may be better in O-notation, but have constants so large that it is impractical. Fortunately:

3. Computers are increasingly parallel.

Laptops have graphics processors (GPUs) with ≈ 100 processors, IBM's Watson computer won Jeopardy by using $\approx 3,000$ cores in parallel, and in 2019 there will be a supercomputer with $\approx 100,000,000$ cores, capable of a billion billion operations per second. However, a serial algorithm would use only 0.00000001 of it.

For many important problems, to solve 1), fact 2) shows that now you must exploit 3). However, there are relatively few efficient parallel programs. It is an active, broad, research area that people will care about for many years, e.g., when you finish your thesis in 4 or 5 years.

USING SUPERCOMPUTERS FOR COMPUTATIONAL SCIENCE: This typically involves being part of large multidisciplinary teams: recent ones include climate modeling, a NASA National Grand Challenge team for Sun-Earth modeling, and a DOE Center of Excellence in Predictive Science modeling high-energy laser experiments and improving computer simulation methodology. NASA uses our software to predict the Earth's magnetic fields (get the NASA SWAN Mac widget to see the predictions) and NOAA (the national weather forecasting agency) uses our Earth Systems Modeling Framework. All US weather forecasts are based on NOAA's simulations. Bob Oehmke worked on aspects of this as a grad student, and is now playing a major role in ESMF development.

Computer science aspects include developing algorithms and data structures that scale to thousands of processors, and software frameworks that combine multiple components (such as models of the atmosphere, ocean, and land) into an integrated simulation. We do performance analyses of the system. This sometimes leads to discovering unexpected behavior, which has led to some theses. Some of these were focused on improving performance, while others were very theoretical, focusing on the causes and asymptotic effects.

PARALLEL ALGORITHMS: Pick your favorite area: combinatorial optimization and graph algorithms (shortest path, optimal flow, etc.), pattern matching, computational geometry (nearest points, intersecting polygons, etc.); clustering, ... You know some efficient serial algorithms. How useful are they for parallel computers?

Parallel computers offer several interesting/difficult challenges. For example, no single processor has all of the data, and perhaps has only a tiny fraction. Thus individual processors might take actions, such as selecting an edge of a graph, that seem appropriate given their data, but are globally wrong. For example, it might think it has not created a cycle, but when all of the processors' choices are combined there is a cycle. Thus parallel algorithms have to insure that globally consistent choices are made, but the processors have to do this with only local data. Think globally, act locally.

We look at numerous algorithmic problems for both real machines and abstract models. For real systems we invent an algorithm, implement the algorithm, do performance analysis, improve the algorithm and/or implementation, repeat. Depth-first is bad, breadth-first is OK: often you need to completely rethink how to solve the problem. That makes it more interesting to work on.

For more theoretical problems we often look at an elegant model of parallelism where tiny processors work together to solve a problem, each communicating only with the processors adjacent to it. This is a fundamental model of information and computation spread out in space, showing the limits of classical physics due to the finite speed of light. How should they move information around in space to efficiently solve the problem? Storing and moving information so that all the processors can simultaneously access what they need is the parallel equivalent of a data structure.

One can build thousands of simple processors per chip. This gives you massive parallelization, but exacerbates the problem mentioned above, namely, each processor has only a tiny fraction of the data. Hence an algorithm has to heavily rely on massive cooperation, yet needs to minimize communication in order to keep the time reasonable. It is somewhat like a committee meeting: you need a certain number of members to get the task done quickly, but if you have too many then too much time is spent in talking and the task takes longer.

In our model, a 2-dimensional grid of n processors can sort n items in $\Theta(\sqrt{n})$ time and a 3-d grid can sort them in $\Theta(\sqrt[3]{n})$ time. These results also hold for problems such as finding a minimal spanning tree or intersecting polygons. This is all using classical physics, and is far faster than serial or quantum computing. For some problems, such as matrix multiplication, optimal results are known for 2-d, but not for 3-d, and the 3-d algorithms are fundamentally different. Strassen matrix multiplication cannot speed up 2-d, but in 3-d can be implemented and reduce the time. However, the fastest known serial algorithms do not help 3-d. It is somewhere in-between, but no one knows where.

Parallel Algorithms Minimizing Energy: A new area of research is designing algorithms to minimize energy consumption. People want to maximize the battery life of cellphones, and want to keep supercomputer electric bills at only a few million \$/year. Power consumption is viewed as the most important, and difficult, constraint on developing exascale computers. Serial algorithms optimize time, parallel algorithms optimize number of processors vs. time tradeoffs. Now we're starting to consider energy vs. time vs. number of processors tradeoffs.

There are also some intriguing possibilities involving optical connections, which you can think of as wormholes connecting processors far apart or acting like dendrites in your brain. Algorithms can be faster and/or use less energy, and experimental chips already incorporate optical connections. Patrick Poon is working on this.

ADAPTIVE LEARNING, CLINICAL TRIALS: Standard clinical trials put half the patients on one treatment, half on the other, collect the results, and then analyze them. Wouldn't you prefer to get the better treatment, rather than having only a 50/50 chance of getting it? In adaptive trials, new patients are assigned to treatments based on the previous results. There are competing goals: to learn the best treatment for patients after the trial is over, and to do the best for the patients in the trial. The FDA has recently decided to encourage adaptive clinical trials, though they note that more needs to be done to computationally design and analyze them. The EPA uses adaptive toxicology tests on rats, and adaptive online GREs are used on students.

We look at tradeoffs, various models, etc., designing new algorithms and ways to approach stochastic optimization problems. There are papers where people said it was too difficult (or impossible) to do what we now compute easily. This is a "bandit" model of optimization, a very general model in decision theory, machine learning, etc. Bob Ohemke's work was finalist for Best Paper at the Supercomputing conference.

MISCELLANEOUS PROJECTS WITH STUDENTS: Andy Poe wrote a serial algorithm for the Ham Sandwich theorem and applied it to domain decomposition for car crash simulation, and I've developed algorithms for fitting optimal stair-step functions to data, motivated by adaptive learning. Rona Kopp computed the largest Busy Beaver value known, Julia Lipman analyzed mathematical models of how fast groups get work done when individuals are randomly interrupted (motivated by Ted Tabe's timings of communication in parallel computers that didn't match what we expected), Phil MacKenzie developed lower bounds based on shooting at targets (and its relevance to a $\Theta(\log^* n)$ expected time parallel algorithm for finding Hamiltonian cycles), Reiko Tanese developed the "island model" of optimizing via the genetic algorithm, etc. Often we didn't realize that these algorithms or models were important until after we started working on a bigger problem for which they turned out to be a key aspect. Several projects involved the study of complex systems and the effect of many objects interacting in parallel.

A new area of interest is data mining of various forms on large data sets. I'd like to work on this with people interested in finding complex patterns and/or apply machine intelligence. Another area of interest involves exploiting the fact that some data sets are generated by an underlying random process. Web links, social networks, etc., are generated dynamically by various processes, and many people are interested in understanding and exploiting this. Often they believe they understand the processes in a vague, general way, but the details are unknown. Nonetheless, perhaps this is enough information to develop algorithms that have a faster expected case.

Also, as I mentioned, I'll also listen to ideas about quite different projects.

CONTACT: The easiest contact is via email: qstout@umich.edu, or drop by my office: 3605 BBB.