Research Statement

Pramod Ganapathi

My major research interests include parallel algorithms, cache-efficient algorithms, distributed algorithms, automatic algorithm discovery, and machine learning.

Past Research. My Ph.D. work, supervised by Prof. Rezaul Chowdhury, took a significant step toward giving computers the abilities of algorithm design and problem-solving. The work focused on automation of algorithm discovery for important classes of algorithms. We designed an algorithm called ***Autogen*** [1] that can automatically discover recursive divide-and-conquer algorithms for solving a wide class of dynamic programming (DP) problems given problem specifications. The output algorithms are efficient (parallel and cache-efficient i.e. minimizing the number of data transfers between different levels of memories) and portable (across shared-memory multicore computers with different cache and processing parameters). Autogen is the first algorithm to automatically discover nontrivial divide-and-conquer algorithms.

We designed several related algorithmic frameworks for DP problems, which take as input the Autogen-discovered algorithms, and they are: [***Autogen-Wave***] [2], to derive recursive wavefront algorithms with improved parallelism. [***Autogen-Fractile***], to discover tiled/blocked algorithms having excellent cache locality on multi-core/many-core/GPU machines. [***Autogen-Tradeoff***], to automatically discover algorithms with space-parallelism tradeoff (i.e. more parallelism with more memory). Furthermore, we designed an efficient and portable algorithm for solving the complicated *Viterbi DP problem* [3].

DP problems are typically solved using simple iterative looping algorithms that are portable but inefficient or using complicated blocked/tiled algorithms that are efficient but not portable (without code modification). DP problems are also solved using divide-and-conquer algorithms that are efficient and portable. However, designing such algorithms is time-consuming even for experts. This means that it is difficult for computational scientists working in other fields of sciences (e.g.: bioinformatics, mathematical optimization, and physics) without any formal training in computer science to design efficient and portable algorithms for DP problems they encounter frequently. Autogen addresses this issue in practice so that it can be executed by ordinary programmers to automatically generate a pseudocode of an efficient and portable provably-correct DP algorithm within a fraction of a second.

We also designed several algorithms and data structures [4] for solving ***range queries*** in a bit matrix occupying sublinear space (in bits) and having constant query times.

Future Research. I would like to work on the following high-impact research projects:

[1. Parallel & Distributed Algorithms.] The two key factors to achieve high performance are parallelism and cache locality. Furthermore, two key factors that are important in the ever-changing and data-intensive world are portability and scalability, respectively. Designing, proving, analyzing, implementing, and debugging highly parallel algorithms are nontrivial due to factors such as race conditions, load balancing, indeterminacy, and deadlocks. We would like to design *nontrivial, provably-correct, portable, highly parallel, scalable, cache-efficient (and/or* *communication-efficient) algorithms to solve important problems in a wide variety of fields including computational sciences, traditional algorithmics, and machine learning.*

A few fundamental questions we would like to address in the long-term are:

(1) Is there a generic approach to design a good parallel algorithm for any given problem?
Several parallel algorithm design techniques have been developed to design parallel algorithms for different problems. However, it is unclear which techniques are suitable to solve which classes of problems. The question aims to unify different parallel design techniques into a single framework to solve any given problem.

(2) Are there generic approaches to automatically port/transform parallel/distributed algorithms from one computing model to another?
Currently, there are a wide variety of computing models (e.g.: PRAM, DAG, fork-join, BSP, PMH, multicore, pthreads, logP) and algorithms. Developing techniques to port algorithms from one model to another saves a huge amount of time for researchers and developers.

A few problems we would like to target in the short-term are: *(i) Automatic visualization of parallel algorithms in the fork-join model, (ii) Designing efficient algorithms exploiting both intra-node and inter-node parallelization*, and *(iii) Designing communication-efficient algorithms for solving matrix problems.*

[2. Automation of Algorithm Discovery.] My Ph.D. work focused on the automatic discovery of efficient algorithms for a wide class of DP problems. It will be interesting to design algorithms for automatic discovery of efficient algorithms based on various algorithm design techniques such as divide-and-conquer, greedy, and backtracking.

How are problems solved? The algorithmic problem of generating a random maze has multiple solutions based on backtracking. The famous k-eggs algorithmic problem has multiple solutions based on DP. The algorithmic problems of sorting or median finding in the union of two sorted lists have multiple solutions based on divide-and-conquer. Generalizing for all algorithm design techniques, we can ask:

(1) What mathematical properties must a problem satisfy for a particular algorithm design technique to be applicable?
(2) Among many ways in which a particular technique can be applied to solve a problem, how to know which is the best way?
(3) How can we apply an algorithm design technique to solve problems (semi-)automatically? A

Answering such extremely fundamental questions helps in understanding how humans solve problems so that we can transfer those abilities to unintelligent machines to make them intelligent and solve complex problems faster.

We have a few preliminary results to simplify human algorithm design. We designed an algorithmic framework that uses backtracking to simplify human algorithm design for combinatorial problems. The combinatorial algorithms that can be generated by the framework include permutations, combinations, subsets, Catalan numbers, partitions, compositions, gray codes, palindromes, derangements, and solutions to Diophantine equations, sudoku, and the n-queens problem. We also designed an algorithmic framework that simplifies the human design of permutation algorithms. The framework can be used to generate 21 permutation algorithms including the well-known permutation algorithms of Wells, Langdon, Zaks, Tompkins, Lipski, and Heap.

[3. Machine Learning.] In the last decade, research and development in machine learning has exploded. Machine learning algorithms that learn hidden functions, relations, and patterns from right data are being widely deployed in industry due to their practicality and versatility.

A few fundamental questions we would like to address are:

(1) Can machine learning algorithms solve/improvise the traditional problems/algorithms?

There are several important problems from computational sciences and linear algebra that are solved by traditional algorithms having super-quadratic time complexity. If machine learning algorithms can solve such problems correctly/probabilistically/approximately and run faster than the traditional algorithms, then machine learning algorithms can be used to either complement or completely replace certain important traditional algorithms.

(2) How can we improve the performance of deep neural networks and other machine learning algorithms?

Deep neural networks and other machine learning algorithms (e.g.: EM) are based on DP. The training time required for deep neural networks ranges from days to weeks depending on the complexity of the problem, data, and algorithm. We designed an efficient (highly parallel, cache-efficient) and portable (cache- and processor-oblivious) algorithm for solving the complicated Viterbi DP problem [3]. Similarly, we would like to design high-performing deep neural networks and other machine learning algorithms exploiting cache locality and (non-trivial) parallelism to reduce the training and testing times.

Based on my past accomplishments and future visions and directions, I strongly believe that I can contribute to society significantly through research and education. It would give me unparalleled joy to get an opportunity in making the world a better place to live.

References:

[1] Rezaul Chowdhury, Pramod Ganapathi, Stephen Tschudi, Jesmin Jahan Tithi, Charles Bachmeier, Charles E. Leiserson, Armando Solar-Lezama, Bradley C. Kuszmaul, and Yuan Tang. "Autogen: Automatic Discovery of Efficient Recursive Divide-&-Conquer Algorithms for Solving Dynamic Programming Problems." (Invited Paper) *ACM Transactions on Parallel Computing* **(Special Issue for Top Papers from PPoPP 2016)**. 2017. 4(1):4.

[2] Rezaul Chowdhury, Pramod Ganapathi, Yuan Tang, and Jesmin Jahan Tithi. "Provably Efficient Scheduling of Cache-Oblivious Wavefront Algorithms." *29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 2017. pp. 339-350.

[3] Rezaul Chowdhury, Pramod Ganapathi, Vivek Pradhan, Jesmin Jahan Tithi, and Yunpeng Xiao. "An Efficient Cache-Oblivious Parallel Viterbi Algorithm." *22nd International European Conference on Parallel Processing (Euro-Par)*. 2016. pp. 574-587.

[4] Michael A. Bender, Rezaul Chowdhury, Pramod Ganapathi, Samuel McCauley, and Yuan Tang. "The Range 1 Query (R1Q) Problem." (Invited Paper) *Theoretical Computer Science (TCS)* **(Special Issue for Top Papers from COCOON 2014)**. 2016.