

Sanjukta Bhowmick: Statement of Research

My interests are in high performance computing, specifically in areas that involve a liaison between non-numerical (graph theory, machine learning) and numerical (large scale sparse linear solvers) algorithms. My research includes algorithm development, software design, and their evaluation in context of large scale parallel applications.

With computation power increasing to petascale dimensions, high performance computing is poised to open unique opportunities for scientific discovery that were beyond the scope of theory and experiments. This, however, requires a close collaboration between computer scientists, numerical analysts, and researchers from basic sciences and engineering, who will be designing the experiments. I have had experience of working in computer science (The Pennsylvania State University) and applied mathematics (Columbia University) departments, as well as in a national laboratory (Argonne National Laboratory). This makes me very well suited to the interdisciplinary research that will be the norm of future scientific computing.

I believe that the full potential of cyber-infrastructure can be realized by focusing on the following four objectives: **(i)** optimizing programs to achieve near-maximum utilization of computational resources, **(ii)** designing a robust, reliable, easy-to-use problem-solving environment, **(iii)** developing infrastructure to access and analyze the explosion of data generated by the experiments, and **(iv)** enabling insights to scientific phenomena through mathematical models. My research focuses on welding computer science and computational science algorithms to bring us closer to attaining these goals.

Core aspects of my research involve combinatorial scientific computing and solution of large scale sparse linear systems. A short descriptions of my projects are given below, as a representation of my research philosophy.

Combinatorial Scientific Computing: *When the arithmetic is easy and the challenge lies in efficient ordering of a sequence of operations, computational science turns to graph theory [11].*

Combinatorial scientific computing uses graph theory to optimize computational science algorithms, for example, in reducing the fill-in of direct solvers, load balancing, and automatic differentiation (AD). The challenge in this area, apart from identifying combinatorial methods that are applicable to the scientific computing problems, is that, either these algorithms have non-polynomial complexity and therefore we have to design heuristics for near-optimal performance, or there is no perfect match, and the algorithms have to be modified beyond their original context to fit the problem requirements.

Automatic Differentiation: Automatic differentiation [9] is based on evaluating exact derivatives of functions by using the chain rule, in order to avoid the inevitable numerical errors of Taylor's method. Function sequences in the chain rule are represented as directed acyclic graphs. I worked on this problem during my postdoctoral appointment with the AD group at the Argonne National Laboratory under Dr. Paul Hovland.

In context of calculating sparse Jacobian derivatives, we designed a *backtracking correction heuristic* [4] to improve results of graph coloring algorithms. Graph coloring problem is NP-hard. Our heuristic dynamically rearranges the colors assigned by a top level heuristic to a more favorable permutation, thereby improving the performance of the original coloring algorithm.

Another problem that I worked on was optimization of Hessian computation. We can exploit the symmetric structure of a Hessian computational graph to potentially reduce the number of operations by 50%. Detection of symmetry of a general graph is NP-complete. But computational graphs in automatic differentiation have special properties, such as being a directed acyclic graph (DAG) with even number of vertices. Taking advantage of these properties, we have designed a polynomial time algorithm that can detect symmetry in Hessian computational graphs. As an interesting byproduct, we were able to extend our algorithm to detect symmetry for any DAGs [5].

Applications of Graph Embedding: Graph embedding is used to arrange vertices and edges in an aesthetically pleasing pattern on a two dimensional space. One of my ongoing projects in collaboration with researchers at the Pennsylvania State University (Dr. Padma Raghavan, Dr. Mahmut Kandemir and Anirban

Chatterjee) is to improve clustering of massive databases, such as those related to textmining, by extending embedding concepts to placement in an n-dimensional space. We use clustering to optimize data layout in caches. This is very important for sparse matrix vector multiplications. The access pattern of the vector depends on the non-zero pattern of the matrix, and clustering the non-zeros in the matrix improves the cache locality of the vector.

Automatic Tuning of Linear Solvers: *The ultimate success [of linear solver software]...is the freedom of computational scientists to move ahead, without bumping into the solver at every turn [2].*

Linear solvers are at the core of most scientific and engineering codes and are designed to cater to different solution requirements. Matching suitable linear solvers to problem characteristics can produce near optimal performance. However, given the variety of linear solvers, ranging across different methods (direct, iterative, multigrid), preconditioners (incomplete LU, sparse approximate inverse) and other parameters (matrix ordering, grid size), the algorithmic options explode combinatorially. The task of evaluating the different options for a suitable linear solver proves daunting to end users of the application code, most of whom are not trained primarily as computational scientists.

Therefore, scientists designing linear solvers, or for that matter any such core scientific computing method, should look beyond algorithms. They should also focus on software that would allow end users to efficiently explore the choices, in order to obtain the maximum benefits that such variety provides. In course of my research, I have investigated two possible solutions to automatic tuning of linear solvers, as described below.

Multimethod Solvers: My doctoral dissertation concerned the development of robust and scalable algorithms for sparse linear solution by leveraging the strengths of existing linear solvers. This work was done in collaboration with my thesis advisor, Dr. Padma Raghavan and Drs. Lois Curfman, Boyana Norris and Dinesh Kaushik at the MCS Division of Argonne National Laboratory. We developed the composite multimethod solver, where the linear system is applied to a sequence of solvers, and the adaptive multimethod solver, where linear solvers are dynamically selected to match evolving linear system properties. These multimethod solvers were tested on computational fluid dynamics applications, and the results demonstrated that they lead to robust solution and faster execution time [6, 7].

Machine Learning for Solver Selection: Given the extensive use of linear solvers, there exists a readily available wealth of data concerning the performance of solvers with respect to system characteristics. During my postdoctoral tenure, I was involved in a project where we used machine learning techniques to analyze such data sets. We were able to predict the most suitable linear solver based on characteristics of a given linear system. This work was done in collaboration with Dr. David Keyes, my advisor at Columbia University, Dr. Victor Eijkhout from Texas Advanced Computing Center, Dr. Yoav Freund from University of California, San Diego, and Erika Fuentes from University of Tennessee. Based on experiments on applications, such as the M3D plasma simulation code from Princeton Plasma Physics Laboratory, we demonstrated that the most efficient solver varies across linear systems generated during different stages of the same simulation, and machine learning techniques, like boosting, can be used to select these “good” solvers [3].

Future Research Plans: My research to date has focused on optimization (graph coloring, automatic differentiation, improving locality in caches), software design (multimethod solvers) and data management (textmining, machine learning). I plan to build on this research and utilize these computational algorithms in new domains, such as computational biology.

Scientific Computing in Biology: Examples of areas in biology which provide interesting platforms for applying scientific computing tools, and in particular, algorithms from my areas of expertise are disease dynamics, and modeling of the condensed chromatin fiber.

Disease Dynamics: Modeling and simulation of the evolution of infectious disease is an important research problem, both in medical (epidemiology) and engineering (computer virus) fields [8]. For computational scientists, disease dynamics present a wealth of opportunities for applying combinatorial and scientific computing techniques. Here are some of them:

Social networks over which the disease spreads are generally assumed to be scale-free networks. However, it is very difficult to determine whether a network is scale-free. Since these networks are self-similar, like fractals, variation of algorithms for detecting isomorphism might be effective. This strategy is similar to my earlier work on symmetry detection in DAGs. I think those algorithms, or variations thereof, might be used in identifying self-similar networks. Another method of detecting these networks would be to locate highly connected nodes. Clustering algorithms, such as the ones on which I have been working, can be used here.

The rate of spread of disease is generally modeled as partial differential equations. However, there is not much research on efficient discretization and solution of these equations. Using techniques such as multi-method solvers, we can generate simulations at faster rates.

Modeling Chromatin Fiber: This is a potential collaborative project with scientists in the Molecular Biology Department at the Pennsylvania State University. The structure of a chromatin fiber is a relatively uncharted territory. The beads-and-string model of the fiber can be modeled as a graph, with nucleosomes being vertices and the linker-DNA being edges [12]. On the basis of these models, we can design computational chromatin reconfiguration strategies, and verify them against experimentally obtained results.

Quasi-Symmetry Detection Algorithms: Inherent symmetry in a graph can be used for optimizing calculations in applications such as computer aided design (CAD) and electrical networks. However, graphs with partial symmetry occur more frequently. Detecting partial symmetry is more difficult, because we seek answers related to the “degree of symmetry”, rather than a simple yes or no. So far as I know, this problem is yet unexplored. I think that Quasi-Symmetry detection algorithms are fascinating to investigate, because it would allow us to extend beyond heuristics for isomorphism. Potential collaborators in this project are members of the SciDAC Institute for Combinatorial Scientific Computing and Petascale Simulations (CSCAPES) [1], with whom I have worked during my postdoc.

Autotuning Scientific Computing Algorithms: We are currently facing an “embarrassment of riches”, with extensive computational memory and power, algorithms tailor-made to suit specific applications, and architectures ranging from uniprocessors to multiprocessors to CMPs. For maximum utilization of this wealth, it is imperative to focus on autotuning. Based on my earlier research on linear solvers, I would like to extend these results to other scientific computing algorithms, like eigen value solvers, and mesh generators. I envision designing an interactive software that can take an application code prototype as user input, intelligently select optimal algorithms, and output a computer generated code, tuned to user specifications. Though it seems a very ambitious project, I have the necessary pieces of software, mostly generated from my past research. Our research on linear solver selection has already generated interest of the application community, including researchers working on the SciDAC-funded ground water project at Oak Ridge National Laboratory [10]. We are currently working on submitting a NSF proposal to fund this enterprise.

Final Remarks:

To summarize, my research philosophy is to expand and diversify. During my professional career I have not only worked on diverse scientific computing areas, such as sparse linear solvers and automatic differentiation, but also explored the use of computer science algorithms, such as, combinatorics and machine learning in high performance computing.

I enjoy finding connections between apparently diverse subjects, and applying algorithms of one area to another. It is also very gratifying that the current trend of funding opportunities coincides with my research interests. Apart from the ‘Faculty Early Career Development’ (CAREER) Program of NSF, many recurring NSF calls target multidisciplinary collaborations, for example, the ‘Cyber-Enabled Discovery and Innovation (CDI)’, and the ‘Expeditions in Computing’ calls. Autotuning and improvements to cyber-infrastructure is also the focus of many funding opportunities, such as the National Science Foundation(NSF) calls for ‘Software Development for Cyberinfrastructure’ (SDCI), and ‘Strategic Technologies for Cyberinfrastructure’ (STCI). The Department of Energy (DOE) through its ‘Advanced Scientific Computing Research’ (ASCR)

program and ‘Scientific Discovery through Advanced Computing’ (SciDAC) institutes also promotes scientific computing research. In this era of burgeoning interest in interdisciplinary research, it is indeed exciting to be a computational scientist.

References

1. CSCAPES: Combinatorial Scientific Computing and Petascale Simulations. www.cscapes.org.
2. A SCIENCE-BASED CASE FOR LARGE-SCALE SIMULATION Volume 2, 2004. www.pnl.gov/scales/docs/SCaLeS_v2_draft_toc.pdf.
3. S. Bhowmick, V. Eijkhout, Y. Freund, E. Fuentes, and D. Keyes. Application of machine learning to the selection of sparse linear solvers. *submitted to IJHPCA*, 2006. <http://www.cse.psu.edu/~bhowmick/>.
4. S. Bhowmick and P. Hovland. A backtracking correction heuristic for graph coloring algorithms. www.cerfacs.fr/algor/CSC05/Slides/bhowmick.pdf.
5. S. Bhowmick and P. Hovland. A polynomial time algorithm for detection of axial symmetry in directed acyclic graphs. Preprint ANL/MCS-P1314-0106.
6. S. Bhowmick, D. Kaushik, L. McInnes, B. Norris, and P. Raghavan. Parallel adaptive solvers in compressible PETSc-FUN3D simulations. In *Proceedings of the 17th International Conference on Parallel Computational Fluid Dynamics, University of Maryland, College Park, MD, May 24–27, 2005*. [ftp://info.mcs.anl.gov/pub/tech_reports/reports/P1279.pdf](http://info.mcs.anl.gov/pub/tech_reports/reports/P1279.pdf).
7. S. Bhowmick, L. McInnes, Norris B, and P. Raghavan. Robust algorithms and software for parallel PDE-based simulations. In *Proceedings of the Advanced Simulation Technologies Conference, ASTC’04, April 18 - 22, 2004*. Society for Modeling and Simulation International (SCS), 2004. High Performance Computing Symposium, Arlington, Virginia.
8. B. T. Grenfell, O. N. Bjornstad, and B. F. Finkenstadt. Dynamics of Measles Epidemics: Scaling Noise, Determinism, and Predictability with the TSIR Model. *Ecological Monographs*, 72, May 2002.
9. A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2000.
10. P. C. Lichtner R. T. Mills, C. Lu and G. E. Hammond. Simulating subsurface flow and transport on ultrascale computers using pflotran. *Journal of Physics: Conference Series, (Proceedings of SciDAC 2007)*, 78, 2007.
11. N. L. Trefethen. Maxims about numerical mathematics, computers, science, and life. *SIAM News*, Jan/Feb 1998.
12. G. Wedemann and J. Langowski. Computer simulation of the 30-nanometer chromatin fiber.