Statement of Research

Graph partitioning, scientific computing and parallel processing are the principal areas of my research activities and interests. I have developed serial and parallel graph partitioning techniques, developed numerical infrastructure for the modeling of Josephson junctions, parallelized the KIVA software library, which is used for modeling internal combustion engines, and actively contributed to developing parallel algorithms for sparse iterative solvers. An overview of my research activities and interests in these areas is provided in the following sections.

I am particularly interested in continuing my research efforts at an institution that encourages interdisciplinary collaboration towards the solution of a variety of interesting scientific and engineering problems. I especially look forward to working in an environment that nurtures and rewards scientific achievement. Furthermore, the prospect of working on real world problems in engineering and life sciences is particularly appealing to me. It is my conviction that the computational challenges of our time demand a new breed of scientists who can combine rigorous mathematical modeling with advanced computer science skills.

Graph Partitioning

My dissertation work focuses on the problem of partitioning large sparse unstructured graphs, and its applications. Algorithms that find good partitionings of highly unstructured graphs are critical in developing efficient solutions for a wide range of problems in many application areas on both serial and parallel computers. These graphs arise from many diverse areas such as scientific computing, operations research, optimization, data mining, geographical information systems, VLSI design, parallel processing, and task scheduling. Depending on the application, the graph can model different quantities. For example, in a 3-dimensional finite element problem characterized by a given mesh and its connectivity, every element of the mesh can be represented by a node in the graph, and two graph nodes are connected via an edge when the corresponding finite elements share a face. To perform any type of parallel numerical computations on this mesh using $k$ processors, the mesh or the corresponding graph must be split into $k$ disconnected parts/partitions, in such a manner that the number of elements assigned to every partition is roughly the same and the number of faces split among different partitions is minimal. In other words, traditional graph partitioning algorithms compute a $k$-way partitioning of a graph such that the number of edges that are cut by the partitioning is minimized and each partition has an equal number of vertices. The task of minimizing the edge-cut can be considered as the objective and the requirement that the partitions be of the same size can be considered as the constraint.

My work focuses on multilevel graph partitioning algorithms, which are shown to produce high quality partitionings in a moderate amount of time. Multilevel partitioning algorithms consist of three distinct phases: (a) coarsening, (b) initial partitioning, and (c) refinement. During the coarsening phase, a sequence of successively smaller graphs is constructed; during the initial partitioning phase, a partition of the smallest (i.e., coarsest) graph is computed; and during the refinement phase, this initial partitioning is projected to successively finer graphs and periodically refined. Multilevel graph partitioning algorithms have proven to be very powerful for solving a wide range of problems, provided proper coarsening and refinement schemes can be developed. However, there is no universally applicable coarsening or refinement strategy. Therefore, I developed proper schemes to successfully solve two additional problems that, even though related to the traditional graph partitioning problem, are considerably harder.
The first problem is that of mesh coarsening. In particular, coarsening a mesh so that the resulting coarser mesh is optimal is a critical problem for geometric multigrid methods. Geometric multigrid methods have gained widespread acceptance for solving large systems of linear equations, especially on structured meshes. The convergence of these multigrid iterative methods strongly depends on the quality of the coarse meshes. One of the challenges in successfully extending these methods to unstructured grids is the problem of generating an appropriate set of coarse grids. I developed robust algorithms, both serial and parallel, for generating a sequence of coarse grids from the original unstructured grid. My algorithms treat the problem of coarse grid construction as an optimization problem that tries to optimize the overall quality of the resulting fused elements. These algorithms are available through the ParMGridGen library that was released in 2001.

Traditional graph partitioning algorithms try to evenly distribute the degrees of freedom (nodes) among the \( k \) processors to balance the computational load. This is not necessarily optimal because for direct linear system solvers, it is the fill-in created by the factorization and not the number of degrees of freedom that determines the computational load. The second problem I solved was how to obtain a partitioning that simultaneously balances the actual computational load and the memory use (rather than the number of unknowns), while minimizing the communication load. At the same time, I was able to compute an optimal fill-reducing ordering for each subdomain/partition. This is of importance to domain decomposition-based numerical simulations whose sub-problems corresponding to the various subdomains are solved using sparse direct factorization methods (e.g., FETI). Effective load-balancing of such computations requires that the resulting partitioning simultaneously balances the amount of time required to factor the local subproblem using direct factorization and the number of elements assigned to each processor. Traditional graph-partitioning algorithms cannot be used to load-balance these type of computations as they can only compute partitionings that simultaneously balance constraints defined a priori on the vertices and optimize objectives defined locally on the edges.

**Scientific Computing**

I believe that solving the challenging computational problems of the future will require advanced theoretical and mathematical models combined with sophisticated computational algorithms; and research will not be advanced by a single practitioner, but by a collaborative team of complementary researchers. In recent years, researchers in computer science have recognized the need for **enabling technologies software**, that is, software that allows transparent access to theoretical advances in engineering and science, combined with equally transparent access to computational platforms, especially ones that facilitate distributed computation. The idea is to develop an environment that enables practitioners to describe problems in an abstract framework without worrying about the underlying computational paradigm. The goal is to free engineers and other applied scientists from arcane theoretical and implementation details and allow them to concentrate their efforts on the actual physical problem. Also, these environments must efficiently harness the full potential of the computational infrastructure available. Automatically ensuring scalability and efficiency in a wide range of applications is a particularly exciting and significant problem that is central to modern high performance computing research.

To make myself a valuable asset in such interdisciplinary research, I have developed a fundamental theoretical and computational understanding of the numerical methods used in computational engineering applications. I have extensively studied and utilized finite difference, finite element, and finite volume techniques for solving differential equations. I have also focused on studying and implementing direct and iterative techniques for solving linear systems of equations.

My first experience in interdisciplinary research involved modeling the static properties of superconducting Josephson junctions. The main goal of this project was to characterize the influence of external magnetic fields on the maximum tunneling current for a device. Mathematically, it involves solving non-linear elliptic equations with homogeneous and inhomogeneous electromagnetic properties of the layered device. I developed efficient ELLPACK and MATLAB code to implement a split-Galerkin method to solve the equation set developed by my collaborators — Maxwell equations with quantum Josephson equations describing tunneling phenomena between two superconductors separated by a thin insulator.

I was able to further appreciate the potential impact of interdisciplinary cooperation in applied sci-
ence research through my collaboration on KIVA with researchers from the Departments of Mechanical Engineering and Aerospace Engineering at the University of Minnesota. KIVA, which is a highly complex code for modeling chemically reacting flows with sprays, such as those found in an internal combustion engine, was developed by scientists at Los Alamos National Labs. Armed with a basic knowledge of the underlying physics, I worked on parallelizing the code from a computational aspect. This is a good example of a project that is too complicated to be attacked from a purely engineering or computational perspective.

My expertise is in finding the most efficient mathematical and computational approach to solve the numerical problem derived from the theoretical model developed by my collaborators, and then implementing this approach in a fast, robust, and efficient manner. I believe that with my background and skills I can significantly contribute to attacking a lot of the scientific problems of interest to your institution, such as problems in the areas of atmospheric sciences, turbulence, and computational fluid dynamics.

Parallel Processing

Parallel processing has always been dear to my heart. The first research project I was ever involved in, in 1995, was to create a parallel search tool, similar to grep, using the Message Passing Interface (MPI), on a CRAY T3-D. Although I was still an undergraduate student at the time, it was clear to me right away that parallel processing is the only viable solution for the large class of problems for which a single processor cannot provide a solution in a reasonable amount of time. Therefore, when I decided to pursue graduate studies, I knew that I wanted to attend a university where I could become involved in research focusing on parallel computing. The University of Minnesota was a perfect match because it provided me access to the facilities of the Minnesota Supercomputing Institute and the Army High Performance Computing Research Center.

As part of my research, I developed parallel algorithms for such diverse problems as sparse iterative solvers for linear systems, graph partitioning, and chemically reacting flows with sprays (KIVA). Some of the algorithms developed have been made available to the public in the form of MPI-based libraries and are extensively used in many academic, government, and industrial projects. ParMGridGen, which provides a range of algorithms for creating coarse grids to be used in geometric multigrid methods, is such a library. These parallel algorithms incur a very small communication overhead, achieve a high degree of concurrency (minimize idle processor time), and maintain the high quality of the coarse grids obtained by serial algorithms. ParMGridGen has been downloaded approximately 4,000 times since it was released and has been used by engineers and scientists for a wide range of applications.

Due to the abundance of models and capabilities included in the previously mentioned KIVA library, KIVA is the most widely used library for engine research. Due to the moving piston, fuel droplets occupy different volumes at different times. In a parallel environment this causes additional computational load imbalance because the particles (fuel droplets) are not necessarily distributed uniformly amongst computational nodes. Due to the irregularity and complexity of the problem, as well as the high complexity of the serial code, earlier attempts to parallelize KIVA were not very successful. My parallelization, in collaboration with researchers from engineering, was able to address several issues, such as the computational load imbalance that was mentioned before. The parallel version of the KIVA software package was made available to the Army Research Office (ARO) and the Energy Research Center (ERC) at the University of Wisconsin, and is being further developed by several ERC and ARO researchers.

Parallel processing has been used successfully to solve a wide range of very important problems. However, despite significant advances, parallel processing is still far from becoming a mainstream technique. The high reengineering cost associated with parallelizing serial applications, and the continuous evolution of parallel architectures are the primary impediments to the widespread adoption of parallel computing. Parallel algorithms already developed for existing architectures often need to be reengineered for upcoming ones. Easing the transition from serial to parallel, and from current to future architectures creates enormous opportunities for parallel processing research in a number of areas. An area of particular interest to me is the design of scalable parallel algorithms required by applications that are suitable for emerging parallel architectures.