METACOMPUTING ENVIRONMENTS FOR OPTIMIZATION

Project Summary

This document proposes the construction of metacomputing environments for optimization. By combining distributed hardware, network hardware/software infrastructure, optimization tools, and informational material, this work will produce produce environments powerful enough to tackle problems of unprecedented scale and complexity.

The project integrates fundamental algorithmic and applications-oriented research in optimization with research in software infrastructure for distributed systems management. New algorithms that use the powerful but heterogeneous, high-latency environment of the metacomputer platform will be developed to solve key problems in targeted application areas. The optimization tools will include interfaces that link the client workstation to the metacomputer server in novel ways. The environment will be able to execute in tandem with data-gathering and modeling resources elsewhere in the network.

Focus areas for the algorithmic research include combinatorial optimization, integer programming, global optimization, and nonlinear programming, while our target application areas are transportation, telecommunications, molecular conformation, and weather forecasting. Software infrastructure research will focus on the resource allocation, scheduling, persistency, and security mechanisms needed to maximize the usefulness of the environment, from the viewpoint of the optimization tools and their users. The project will make use of the GUSTO metacomputing testbed being constructed by the Globus project.

The proposal has been prepared by a multidisciplinary team that includes the key players in three existing projects—NEOS, Condor, and Globus—from which the proposed work draws much of its inspiration. These projects have collaborated with each other on a number of pilot projects in recent months and, during that time, have formulated the ideas at the core of this proposal. The group's collective expertise in optimization, metacomputing, and parallel computing and its access to massive computational resources gives it the confidence to tackle a project of the scale and complexity outlined here.

Metacomputing environments for optimization will put leading edge optimization technology at the fingertips of a vast new community of users. They will change the way that optimization technology is used and will greatly expand its impact on science and engineering.

Project Description

1 Metacomputing and Optimization

We propose to construct metacomputing environments for solving optimization problems of unprecedented scale and complexity. These environments, which we refer to collectively by the term **metaNEOS**, will turn existing networked computing resources into productive problem-solving environments by integrating advanced distributed resource managers, novel optimization tools, and informational content into these computational platforms. They will make it possible to solve important problems in the areas that directly motivate our research: transportation, telecommunications, molecular conformation, weather forecasting. metaNEOS will put leading edge optimization technology and huge amounts of computing power at the fingertips of a vast new community of users.

Networks of computers—most visibly, the Internet—have been highly successful as communication and information-sharing devices, but their enormous potential as computational devices has yet to be realized. Metacomputing environments can be built from existing resources—workstations, PCs, and supercomputers, network hardware/software infrastructure, numerical software tools—and can make use of vast amounts of computing resources that would otherwise be wasted. They augment the scarce and expensive supercomputers that are available to some scientists by gathering together resources that are individually less powerful but relatively inexpensive. Users with compute-intensive applications can benefit from the raw power of a metacomputer, while users with smaller problems will benefit from the currency, immediacy, and scope offered by the other resources accessible to the metacomputing environment. Users at all levels can have ready access to a wide range of software tools, along with the user-friendly interfaces and informational resources needed to make productive use of these tools.

In very-large-scale optimization, many problems of great current interest are not well served by existing computational platforms. Integer programming and global optimization problems are perhaps the most obvious examples of problems for which very large amounts of computer time to solve many practical instances. In most cases, these requirements can not be met by a single computer. Fortunately, it is possible to devise algorithms for these problems that are well suited to the intrinsic characteristics of our proposed metacomputing environment, namely,

> *High latency.* Implies asynchronous parallel algorithms that do not require large amounts of data to be transferred between their component tasks.
>
> *Heterogeneity.* Tasks generated by the algorithms should be executable on (and, ideally, tailored to) the computational resources in the heterogeneous collection.
>
> *Dynamic/opportunistic environment.* If additional computational resources become available while the algorithm is executing, it should be able to exploit them. Conversely, if the available resource pool shrinks during execution, the algorithm should be able to reschedule any interrupted tasks on different resources and adapt to the smaller pool.

The algorithmic work we propose here focuses on problems and algorithms that make a good match with these and other characteristics of the metacomputing platform. In integer programming, for instance, branch-and-cut algorithms generate large task pools (each task a continuous linear program), which can in principle be scheduled on a heterogeneous pool of resources of almost arbitrary size, with limited communication or coordination required between the tasks. Naturally, a great deal of research is needed to design a truly practical solver along these lines, as we discuss later in the proposal. The goal is certainly worth pursuing, however, since many integer programming problems of great economic importance simply cannot be solved on today's uniprocessor architectures.

We also propose research in algorithms for global and combinatorial optimization. Our efforts here are motivated by important applications in protein folding/distance geometry and in telecommunications network design/routing, respectively. As we explain below, the nature of these applications and the algorithms that we propose are well suited to the abovementioned characteristics of the metacomputing environment.

Optimization problems that depend on volatile data include weather forecasting, logistics problems in industry, and management of traffic flow. These problems present a somewhat different challenge to metaNEOS: they require data gathered from around the network to be integrated seamlessly in to the optimization calculations, and the solutions to be calculated in real time. In addition, they may need to interact with specialized resources outside of metaNEOS that perform parts of the modeling and calculation. In Section 2.3, we describe the basic metacomputing paradigms and algorithmic developments needed to meet the requirements of these and similar problems.

Research in metacomputing resource management technology and user and application interfaces is as essential to the success of this project as the algorithmic research in optimization outlined above. Our software tools will be served by Web and Windows interfaces, functional interfaces that are callable from legacy C and Fortran applications, and interfaces to optimization modeling languages such as AMPL and GAMS. In resource management, we will utilize existing resource management mechanisms, communications protocols, and security systems to design resource allocation policies for a complex task mix in a dynamic environment. Current authentication capabilities will be augmented to promote maximum throughput without compromising security. The environment will meet high standards of reliability and predictability.

Research results from this project will include new algorithms for the optimization problems mentioned above that are motivated by the characteristics of the metacomputing platform—the natural platform for many very-large-scale practical instances of these problems. By using these algorithms to solve our motivating applications, we will identify and incorporate features in metaNEOS that add value to these algorithms in their typical usage contexts. In metacomputing infrastructure research, we will use the experience gained with the optimization tools and their applications to develop the resource management, resource scheduling, communications, and information directory tools needed to meet a variety of quality-of-service requirements in a computing-intensive discipline with a wide user base.

Two main deliverables will accrue from the project. The first will be the optimization components of metaNEOS—a collection of portable metacomputing-aware optimization software for solving the problems mentioned above. The second will be the metaNEOS infrastructure, which may be used to build analogs of metaNEOS in other areas of computational science and engineering. We will make available resource management tools, secure communications protocols, tools to monitor usage, and tools to install new software and to construct interfaces to this software. We anticipate that this infrastructure will become the backbone of metacomputing environments for fundamental problems classes such as PDEs and numerical linear algebra, as well as for specialized problem areas such as computational chemistry, computational crystallography, and weather forecasting. By focusing on the particular discipline of optimization, with its wide range of users with diverse needs, we will be better able to identify the broad infrastructure needs of similar metacomputing environments than if we were to build this infrastructure in a vacuum.

Research Team Background

Our multidisciplinary team includes leaders in the fields of optimization, metacomputing, and parallel computing. Their involvement in three current projects—NEOS, Condor, and Globus—demonstrates that they have the collective expertise in metacomputing needed to achieve the ambitious goals of the proposed project. Likewise, their track record in algorithms and software for optimization and parallel computing indicates success in this aspect of the project also.

NEOS—the **Network-Enabled Optimization System**—is a current project of some optimization members of the team (Ferris, Moré, Nocedal, Wright). As it stands, NEOS is a highly effective environment for solving a wide range of optimization problems remotely over the Internet, featuring leading-edge software tools, Web and Windows interfaces, integrated automatic differentiation tools, and an extensive collection of informational and educational resources. While NEOS is a forerunner for some of the concepts described in this proposal, our goals for metaNEOS go far beyond the existing features of NEOS in their incorporation of parallel software for large optimization problems, their ability to handle a high volume of single-task and multi-task jobs, their support of new server-client paradigms, and their focus on issues of resource management, security, and persistency.

The metacomputing members of our team are the leaders of two projects—Condor (Livny) and Globus (Foster)—whose goals are to develop the software infrastructure needed to solve large computational problems on distributed resources. The Condor project proposed the first paradigm for high-throughput computing on distributively owned resources in 1986, and has pioneered many advances in resource allocation since then. Globus focuses on issues of resource location and management and communications, and on security issues such as secure data access and authentication. Our project will be able to use the GUSTO metacomputing testbed being constructed by the Globus project. GUSTO integrates computers at many U.S. universities, laboratories, and international sites.

In optimization, members of our team have developed state-of-the-art software for unconstrained and bound-constrained optimization and least-squares calculations (Moré, Nocedal), linear programming (Goldfarb, Wright), combinatorial optimization and integer programming (Bienstock, Ceria, Goldfarb), and complementarity problems (Ferris).

Education

Students will be heavily involved in the design and construction of metaNEOS. They have played a key role in all three of our current projects described above, and have in all cases benefited from their experiences. Much of the design and implementation of the NEOS Server, for instance, was carried out by a talented graduate student in computer science, and construction of the NEOS Guide was performed by teams of graduate and undergraduate students at Argonne and Northwestern, working under faculty supervision. The students' assignments widened their exposure to different areas of optimization, gave them the experience of working at the intersection of two exciting technologies (optimization and the Internet) and put them in touch with applications and users. Students have also played key roles in the development of Condor and Globus,and we feel that their involvement in the project proposed here will be an important part of their education. Accordingly, we have committed a majority of our budget to the support of students and postdoctoral associates.

2 Optimization Problems and Applications

Certain problems in very-large-scale optimization are by their nature an excellent match for metaNEOS, since it is possible to design effective algorithms that exploit the high-latency, heterogeneous, opportunistic characteristics of this environment. We focus here on three such problem classes, and propose algorithmic research that is motivated both by important applications and by our metacomputing platform.

2.1 Combinatorial Optimization and Integer Programming

Many practical problems can be formulated as optimization problems with discrete decision variables. Finding optimal solutions for problems of this type has been perhaps the one of the most important challenges facing the optimization community during the past thirty years.

Discreteness enters into optimization problems in many different ways, giving rise to an extensive taxonomy of problem classes. Widely studied classes include set-covering problems, matching problems, and the traveling salesman problem, to name a few. Specialized algorithms have been devised to exploit the particular character of each class, and many of these methods have been used to solve important practical problems.

Mixed Integer Programming. The most popular way of modeling discrete decision problems is through *mixed-integer linear programming.* This type of models can be stated simply as

$$\min_{x,z} c^T x + d^T z, \qquad Ax + Cz = b, \quad l \leq \begin{pmatrix} x \\ z \end{pmatrix} \leq u, \quad z \text{ an integer vector}, \tag{1}$$

where the components of x are real, and l and u are vectors of upper and lower bounds, respectively. The term "mixed" indicates the presence of both real and integer variables. When z is absent, the problem is a (continuous) linear program. Algorithms for solving (1) proceed by solving a series of *continuous relaxations*, in which the integrality requirement on z is dropped but extra bounds and constraints are added to the formulation.

Many of the more specialized combinatorial problems can be stated in the form (1), and generations of practitioners have been trained to formulate models in this framework. However, there is currently no software for solving (1) that is capable of operating on the distributed heterogeneous platform that underlies metaNEOS.

We will develop a parallel solver for mixed integer programming that incorporates state-of-the-art research results. We will make a version available on our metacomputing environment, as well as a serial version for solving smaller-scale problems.

Computational experiments in metacomputing environments have already shown promise for other combinatorial optimization problems, such as the traveling salesman problem, but very little research has been done in using parallel computation to solve mixed integer programming problems of very large scale. metaNEOS will allow us to solve problems that cannot be solved in uniprocessor computers, or even in small-scale parallel environments. The most effective approaches for solving (1)—branch-and-bound, branch-and-cut—are in principle well suited to this environment. However, the communications requirements between the component tasks are quite complex, so a great deal of research and computational experience is needed to design an efficient solver.

Branch-and-cut is a combination of two well-known methods for mixed integer programming, namely, branch-and-bound and cutting plane algorithms. In branch-and-cut, the automatic cut generation is performed not only prior to starting branch-and-bound but also at each node of the enumeration tree. For this approach to be successful, it is important that the cuts generated at a node of the enumeration tree be valid at all the nodes. So far, this requirement has limited the types of cuts used in branch-and-cut to combinatorial cuts, but recent research has focused on two new kinds of cuts:

> *Lift-and-project cuts* have been shown to be an effective way of strengthening mixed programs in which the integer variables (the components of z) are constrained to be either 0 or 1. Not only do they dominate the classical mixed integer cuts of Gomory, but they are also much more stable from a numerical point of view.
>
> *Disjunctive cuts* are formed by dividing the problem into two subproblems, in much the same manner as a branch-and-bound algorithm. Disjunctive cuts can be generated through the solution of a linear programming problem of roughly the same size of the original problem.

Our research will encompass both the theoretical and computational issues that arise in all branch-and-cut procedures that use general cutting planes. In the first phase, we will implement and test in our metacomputing environment separation procedures for lift-and-project cuts or disjunctive cuts, and possibly some generalizations of these, paying particular attention to the opportunities for parallelism that are produced by each cutting procedure. Second, we will embed these procedures within a general branch-and-cut code and implement the resulting algorithm on our metacomputing platform.

Combinatorial Problems and Telecommunications Design/Routing. Here we describe an important application area—design of telecommunications networks and the routing of traffic on these network—in which the sheer complexity and size of the problems to be solved necessitates the use of a parallel platform. Beyond our immediate interest in solving these problems, our aim in this section is to show how important applications motivate us to include specialized combinatorial algorithms in our metacomputing environment, in addition to the general integer programming solver discussed above.

Network design/routing problems concern the installation of capacity (that is, routing equipment) and the routing of traffic demands to make use of this capacity, so that the cost of the installation is low and the quality of the routing is good. The optimization problems that arise in this context are typically very large and difficult.

A major research goal for us is to develop general-purpose, parallelizable algorithms for problems of this type. In ongoing research (see Bienstock [4]), we are studying how new advances in the approximate solution of certain linear programs—multicommodity flow problems and covering-type linear programs—can be used alongside combinatorial heuristics for set-covering problems to obtain approximate solutions to network design problems. Specifically, the two major issues are as follows:

> Approximate solutions to the continuous relaxations, each quickly obtained, can be used to separate "good" inequalities and hence to strengthen the formulation rapidly. This process then can be used to drive a cutting-plane or a branch-and-cut algorithm.

Modern combinatorial heuristics (see Ceria, Nobili, and Sassano [6]) can be brought to bear on the set (multi-)covering problems that arise in network design. Approximate solution of the continuous relaxations can be used again to speed up these heuristics; see Bienstock [4].

Several issues must be addressed in adapting the approximate solution techniques for multicommodity flow problems and covering-type linear programs (see [43], [38]) to a parallel computing platform. The most promising techniques will break a multicommodity flow problem into a single "main" problem and a large number of independent subproblems, one for each commodity. They will solve the main problem, pass some information to a subset of the subproblems, solve the subproblems, return information to the main problem, and repeat the process. Typically, the main problem takes significantly longer to solve than each individual subproblem, so the approach is suitable for a heterogeneous platform. The requests that each process will make to the metaNEOS resource broker (see Section 3.1) will differ considerably: The main process will generally require a powerful processor, while the subproblems would be assigned to less powerful workstations (possibly with multiple subproblems assigned to each workstation). The research issues that arise in implementing this approach include the following:

How will the high latency of the metacomputer affect the procedure? We can make the algorithm more asynchronous by re-running the main algorithm prior to its receiving information from *all* the subproblems, and can prove that the algorithm still converges provided that information from the subproblems is communicated "often enough."

A more complex asynchronous variant is obtained by replicating the main problem on several loosely communicating processors. Each processor solves the main problem more or less independently, and each communicates with a subset of the subproblems. Convergence of this approach is not yet established; a host of issues related to interplay of communication and aged information must be addressed.

Finally, we turn to the second major issue: combinatorial heuristics for set-covering problems and their parallel implementation. The techniques of Ceria, Nobili, and Sassano [6] can be viewed as a generalization of, and an improvement on, the standard greedy heuristics (see Chvátal [10]). Roughly speaking, these techniques fix variables sequentially, using dual variables to estimate the "goodness" of each primal variable. Again, the most promising parallel approach is to use a "main" process to fix variables and to assign the resulting subproblems to other processors. The two issues described in the previous paragraph recur in a slightly different form: asynchronicity of the information exchange between the main process and subproblems, and possible splitting or replication of the main process. As above, we will investigate different variants of these approaches from both theoretical and practical viewpoints, and study the effectiveness of each asynchronous variant on different resource mixes, so that metaNEOS can make intelligent choices about the optimal strategy to employ at any given time.

2.2 Global Optimization

The protein folding problem in biochemistry and the phase problem in crystallography require the computation of a global minimizer of a potential function. These global optimization problems are extremely difficult because the potential function is rugged with a vast number of local minima, and most practical instances simply cannot be solved because the computational demands are too great.

metaNEOS is an ideal platform for global optimization solvers. They can harness large amounts of powerful resources and, as we explain below, the most popular algorithms are an excellent match for the heterogeneous, high-latency, and dynamic characteristics of the metacomputing environment. For pragmatic reasons, too, metaNEOS has advantages over the "supercomputer center" environment in which most large global optimization problems are solved today. Currently, users need to familiarize themselves with such issues as scheduling policies, job submission methods, and software libraries, all of which are specific to each center and which make it difficult to port applications between/across centers. In contrast, metaNEOS will take care of the scheduling issues transparently to the user, provide a set of user-friendly interfaces similar to the existing NEOS web browser interfaces, and allow the user to call embed a wide range of numerical software tools in their application.

To focus our efforts in developing the algorithms and infrastructure for this aspect of metaNEOS, we will concentrate on two important, representative applications areas: distance geometry modeling from nuclear magnetic resonance data and the simulation of molecular metal clusters.

In the first phase of our work, we will incorporate our successful code DGSOL into metaNEOS. DGSOL uses MPI to implement a parallel algorithm for the problems outlined above on an IBM SP multiprocessor system and on a network of Sun workstations. DGSOL's algorithm smooths the objective function to reduce the number of local minima, and uses local minimization routines to identify the lowest-lying local minima. It then progressively reduces the amount of smoothing, using continuation techniques to trace the local minima, until eventually a set of local minima for the original problem are found. The smoothing and continuation strategies ensure that, in most practical instances, one of these local minima is in fact the global minimum. Users need only provide DGSOL with a set of distance data; the output is a set of one or more structures that minimize the potential function. Currently, DGSOL can solve distance geometry problems with 200 atoms, but we need to extend the range considerably since the smallest protein has on the order of 600 atoms. The computational power of metaNEOS will provide much of the additional capability, but we will also carry out algorithmic research into enhanced continuation techniques, the role of the starting point, sensitivity of the solutions, and the use of multistart, reduced subspace, and mutational techniques to sample conformational space. We will also investigate extensions of smoothing techniques to general potential functions that lack the structure and sparsity of the particular functions above.

The suitability of the DGSOL approach for metaNEOS is clear. Tracing of individual local minima can be carried out by individual tasks executing concurrently on a parallel platform; little exchange of information between these tasks is required. Moreover, the number of such tasks changes during execution of DGSOL, as some minima disappear and others bifurcate, so the allocation of tasks to processes can exploit a dynamically changing resource mix.

As a long term goal, we propose a metaNEOS facility for general global optimization problems in which both the objective function $f(\cdot)$ and constraints $c(\cdot)$ may be nonlinear. Users would be allowed to specify a global steering mechanism and to choose from a variety of local optimization techniques. Alternative global minimization strategies such as multistart, simulated annealing, genetic algorithms would also be considered. (Note that all these approaches can be parallelized via a decomposition of the search space.) As in the existing NEOS, automatic differentiation tools would be embedded in the environment to relieve the user of the need to supply derivative calculation code.

The global optimization facility in metaNEOS will allow scientists to solve problems that are well beyond the range of current codes. The power of the distributed computing platform, the comprehensive nature of the software resources, and the user-friendly nature of the interfaces will combine to enable important advances in many disciplines.

2.3 Problems Defined by Volatile Data

Many practical optimization problems are defined by volatile data sets. Important examples include medium-range weather forecasting via variational data assimilation, logistics problems, and management of vehicular traffic flow. The solutions of these problems depend on factors that change frequently: addition and deletion of meteorological observations (in weather forecasting), inventory levels, demand patterns, and transportation costs (in logistics), and traffic accidents and adverse weather conditions (in traffic flow problems). The challenges for our metacomputing environment are

to seamlessly integrate current data gathered from around the network into the optimization calculations;

to cooperate with specialized computing resources that perform parts of the modeling and calculation outside of the metaNEOS environment; and

to solve the problems in real time.

We focus on two applications with which we are closely involved: weather forecasting and traffic equilibrium. We show how the use of improved algorithms executing in metaNEOS will enable the solution of more realistic models in real time.

Weather Forecasting and Data Assimilation. Accurate numerical weather prediction requires accurate and detailed knowledge of the state of the atmosphere at a given point in time. The equations of atmospheric motion, which are well understood, can then be integrated forward in time from this "initial" point to produce a forecast. The problem in forecasting is primarily to initialize these equations with accurate atmospheric conditions.

Until recently, the determination of the atmospheric condition was rendered impossible by the paucity of data in many areas, particularly over the oceans. The deployment of GPS, SSM/I and other satellite systems is changing this situation by providing a wealth of new information on the ocean an atmosphere. However, much of this satellite data is distributed irregularly in time and space. To be truly useful, some of the data must either be converted into quantities such as wind velocity, moisture, and temperature (a conversion that can be ill-posed in certain situations [13]), or must be directly assimilated into the forecast model.

Four dimensional variational data assimilation (4DVAR) is the most sophisticated method for assimilation of observations [40]. It determines the initial atmospheric conditions by minimizing an objective function that measures the discrepancy between the model solution and all available observations distributed in time and space. In each iteration of 4DVAR, the objective function and its derivatives (with respect to the parameters that describe the initial atmospheric conditions) are determined by integrating the model forward in time and then integrating its adjoint backward in time. This information is input to a nonlinear minimization algorithm, which responds with a new estimate of the initial conditions of the atmosphere. The 4DVAR approach bypasses the issue of data conversion by allowing "raw" data that differs in type from the model parameters to be assimilated directly. It also allows past information to be used alongside current information.

Although it is now accepted as the most sophisticated approach to weather forecasting, 4DVAR has not yet achieved its full potential for two reasons. First, its computational requirements are large. Each of the 20–30 iterations performed in a typical run require integration of the atmospheric model and computation of the adjoint, which is about two to three times as expensive as the forward integration. The second reason is that current 4DVAR practice uses unconstrained optimization models that require the model and observational errors to be small, which in turn requires a high standard of data quality. This latter difficulty will be alleviated by the use of bound-constrained optimization models which, from an optimization software viewpoint, can in principle be implemented with relatively minor changes to unconstrained optimization software.

We propose research into design of the algorithms and the features of metaNEOS needed to make four-dimensional variational data assimilation the approach of choice for practical weather forecasting.

These features will enable us to support the many other potential applications in which model evaluations (function, gradient, and constraint evaluations) must take place on computational platforms outside of the metaNEOS environment proper, for reasons of practicality or security. In the case of 4DVAR, we will use a highly tuned code for integration of the equations of atmospheric motion, developed for the Cray 90 by our collaborators at the National Centers for Environmental Prediction. This model will continue to execute on its dedicated hardware, while we perform the other parts of the calculation (the ray tracing model, the optimization calculations to determine the next estimate for the parameters) and coordinate all aspects of the optimization within the metaNEOS environment.

Algorithmically, we will design optimization algorithms which make less stringent requirements on data quality and which find the solution in as few iterations as possible. The need to minimize the number of iterations can be seen by considering a problem of modest size: For a 6-hour assimilation on a 3D grid with $50 \times 50 \times 10$ points and a two-minute time step, about 20 minutes of computing time are required on the Cray 90 to calculate the objective function and its gradient. A complete optimization with 30 iterations requires 10 hours in evaluation time alone; 10 hours of validity of the resulting forecast is therefore lost.

Since the Hessian of the objective function is dense, Newton's method is out of the question. Candidate algorithms include the nonlinear conjugate gradient method, a finite-difference Newton-like method, and the limited memory BFGS method [44]. The limited memory method (Liu and Nocedal [41]) tends to require the fewest function evaluations and uses no tuning parameter, and it is the method of choice in current implementations of 4DVAR. However, the limited memory approach has two drawbacks: Its asymptotic convergence rate tends to be slow, and it usually fails to achieve good accuracy on ill-conditioned problems.

We propose to develop a new optimization algorithm that combines

finite-difference Newton-like steps, and

enriched limited memory steps that make use of the second-order information generated during the Newton-like step.

Our motivation for this approach is that the number of iterations of the pure limited-memory method decreases significantly when the memory is increased beyond 10 steps, suggesting that additional second-order information will help in decreasing the iteration count further. Compact representations of the limited memory matrices (Byrd, Nocedal, and Schnabel [5]) will facilitate parallel execution of the optimization task.

Traffic Flow and Guidance Information Systems. In systems for traffic flow control and guidance information, the use of realistic models is hindered by the unavailability of computing resources to perform the required calculations in real time. *We propose to incorporate features in metaNEOS and to develop algorithms that allow prediction and control of realistic traffic flow models to be performed in real time.* Obviously, given the state of the art in this area, this is a long-term goal. However, since this application is important both in its own right, and as a representative of similar applications in telecommunications and water resource distribution, we feel that its particular characteristics should strongly influence the design of metaNEOS.

Current systems for predicting and altering flow patterns predominantly use the system equilibrium paradigm based on an overseer solving large scale nonlinear programs to determine users paths through the network. These models are unrealistic in that they assume that the traffic controller is able to force motorists to take a particular route. Other models invoke the user equilibrium paradigm, that assumes users of a network compete noncooperatively for the resources of the network and choose shortest paths for travel, where the time of travel is influenced by nonlinear congestion effects with other users. The user equilibrium is solvable by nonlinear complementarity methods; realistic large scale nonlinear models of this format are currently solvable on workstations in a manner of minutes. However, both these model types are static in nature and assume for example that the state of the traffic network does not change over time and futhermore that motorists will not change routes after their paths have been assigned.

We believe that a stochastic partial user equilibrium model that incorporates dynamic events and structural changes will remove the necessity of forcing compliance, at the same time making it more realistic and implementable. In the stochastic model, flows are measured on a subset of the arcs and a large scale nonlinear program is solved to determine the current parameters of the demand. Based on these new demand functions and information about arc failures (due to traffic accidents), a new equilibrium model can be formulated and solved to determine new equilibrium flows. Information can then be provided to the motorists to enhance their likelihood of using short routes through the network. Alternatively, traffic signals can be adjusted to reduce congestion effects. In this latter case, a parallel local search heuristic could be implemented to solve an equilibrium-constrained mathematical program, to provide optimal time delays for the ramping problems in real time.

We anticipate that a flow control/guidance information system of this type will operate as a distributed but dedicated subnetwork of computers within metaNEOS. The subnetwork will gather and integrate data on current conditions, perform some of the calculations, and return the results to operators of the traffic control system. Resources of metaNEOS outside the subnetwork will be called on to perform larger, less time-critical calculations, or to obtain more comprehensive solutions of problems for which approximate solutions have already been computed on the subnetwork.

3 Software Infrastructure

3.1 Resource Management

Key to the success of metaNEOS are techniques for locating and scheduling computational resources in a dynamic and evolving environment. The issues are challenging, because the set of resource requests will change dynamically as new optimization tasks arrive, other new tasks are created by dynamically sized applications, and existing tasks are completed. Furthermore, the computational resource requirements will differ significantly between tasks. Some (for example, the integer programming solver of Section 2.1) will require relatively tight coupling between their processors, while others (for example, some applications of the global optimization solver of Section 2.2) will execute on almost any computing resource that metaNEOS can assign to it. Requirements for specific hardware or software (as in the volatile data applications of Section 2.3) can impose additional constraints.

We can also expect the number and type of the resources available to metaNEOS to change dynamically, often in unpredictable ways. The hardware resources available to metaNEOS (which will include the GUSTO metacomputing testbed) will comprise a heterogeneous mix of high-end multicomputers, multiprocessors, and workstations. Some of these resources will be dedicated to optimization, at least at certain times, while others will be available only when not required for other tasks. The dynamics and interaction between resource requests and offers we will face in metaNEOS are similar to the issues that Condor has to deal with in managing a distributively owned collection of workstations.

A critical issue in designing schedulers for metaNEOS and similar systems is the expected number of resources and tasks to be handled by a single scheduler. Our working model is that independent metaNEOS systems will be deployed by different communities and for different purposes, and each system might support $10^2 - 10^5$ processors. (Management of collections of this size is not out of range of our current systems: Some Condor pools comprise 300 processors and there are Condor flocks with 600 processors, while the Globus software deployed on the I-WAY provided a coordinated view of several thousand processors.) These distinct metaNEOS systems may be linked in a loose federation if effective mechanisms for resource trading are devised, but we do not propose to develop such mechanisms in this work. Though the workload will vary from one metaNEOS system to the next, each system should be equipped to handle optimization jobs like those described in Section 2 as well as jobs of the type handled by the existing NEOS system. As we plan for success, we must anticipate that the requests will exceed available resources, so our scheduling policies must be able to resolve potentially conflicting requests.

We propose to address these issues in the context of the architecture illustrated in Figure 1. The *metaNEOS resource broker*, which we will develop as part of this research, will be responsible for matching application requests with available resources. Not only is the broker responsible for locating suitable resources for each individual request but, unlike an application-specific scheduler [3], it will also seek to optimize overall utilization or resources. Often, the broker will be accessing resources that are not dedicated to metaNEOS, and which are themselves managed by other "local" schedulers of various types.

The basic function of the broker is to direct tasks to one or more of the several schedulers to which it has access. Its design will require the integration of techniques pioneered in Condor (which are oriented toward high-throughput computing) with techniques being developed in Globus (which has a high-performance orientation). The Globus Resource Manager (GRM) [1]

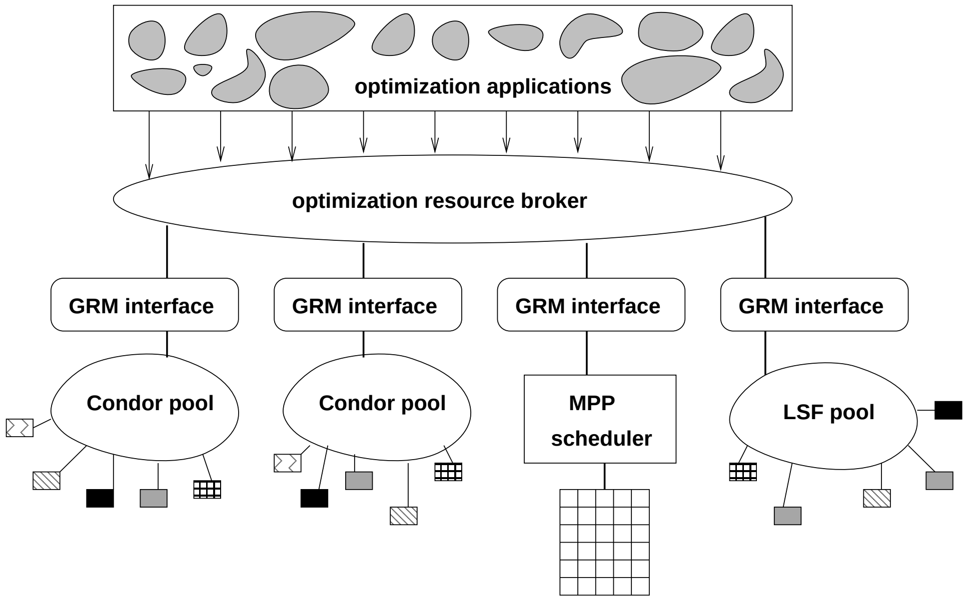

Figure 1: Architecture of our optimization resource management system

defines a resource description language and a uniform low-level API for managing specific schedulers, such as MPP queue managers, Condor, and Platform Computing Corporation's Load Sharing Facility. The GRM API makes it possible to build a single broker that interacts with and schedules to multiple schedulers. Relevant features of Condor include its Matchmaker and its Classified Ads. The specific issues that we will address in integrating these components into the metaNEOS broker are as follows:

How do we represent resource capabilities? Representations such as SNMP MIBs, X.500/LDAP object classes, Condor Classified Ads, and the Globus Metacomputing Directory Service (MDS) provide useful concepts that are able to represent complex resources such as LSF clusters and the usage policies of different system components. We propose to integrate and extend MDS and Condor concepts to provide a representation that can deal with the complex resource hierarchies illustrated in Figure 1.

How do we represent application resource requirements? Condor Classified Ads and the Globus Resource Manager (GRM) description language both provide useful concepts, but cannot currently describe the multidimensional performance objectives of metaNEOS tasks. Some applications require high degrees of reliability, while others may require rapid turnaround. The broker should take requirements such as these into account in deciding where where to place tasks. We propose to integrate and extend MDS and Condor concepts to provide appropriate capabilities.

How do we implement the metaNEOS broker? Experience with the Condor matchmaker suggests that a centralized structure can be quite scalable. However, we anticipate requiring some degree of distribution. The relative advantages of stateless and stateful brokers will be studied, as will the tradeoffs between the sophistication of the algorithms used within the broker and the associated overhead. In particular, we will explore the feasibility of using optimization techniques themselves to compute efficient schedules.

How do we meet application Quality-of-Service requirements? The broker should support both high-throughput scheduling (in which the focus is on delivering large amounts of computing cycles over long time periods, as implemented in Condor) and high-performance scheduling (in which the emphasis is on providing performance guarantees). A simple approach is to dedicate distinct resources to each type of task. Alternatively, a broker can utilize specialized strategies (for example, mirroring or hot standbys) to provide performance guarantees when using resources with less predictable performance characteristics. Expected task runtimes, which can be provided accurately for some problem types in optimization, will be used to improve the quality of our scheduling.

We provide a little more background on languages for describing resource requirements. The GRM resource description language allows users to specify boolean combinations of simple expressions that describe constraints on the number and type of the compute resources required for a particular program, as well as information about the program itself. Two very simple examples illustrate its use. The first requests that a program `myprog` execute on between 5 and 10 processors, each with at least 64MB memory and available for at least 4 hours:

```
&(count=5)(count<=10)(max_time=240)(memory=64)(executable=myprog)
```

The second requests either 5 processors with at least 64 MB of memory, or 10 nodes with at least 32 MB:

```
&(executable=myprog)(|(&(count=5)(memory=64))(&(count=10)(memory=32)))
```

The broker will also make use of Globus' Metacomputing Directory Service (MDS) [14], which provides a uniform interface to diverse information about the structure and state of the resources in a distributed computing system. MDS information will be used to locate the resources that the optimization resource broker will use as a basis for its scheduling decisions.

The Globus security service [2] provides public-key based authentication mechanisms.

3.2 Persistency

For a metacomputing environment to be adopted as a production tool by users who solve practical problems, it must provide reliable service. Providing reliability is by no means a simple task in an environment where resources are distributively owned, locally managed, and can enter and leave at will. Users expect the metacomputing environment to guarantee successful completion of each and every job, but the management software of the environment can take very little for granted. Chaos is at once the major source of strength and the enemy of metacomputing.

For more than a decade, the Condor project has been struggling with the challenges of providing reliable computing services on very large collections of distributively owned resources. Condor employs sophisticated checkpointing, logging, replication, recovery, remote execution, and tracing mechanisms, to ensure that it can deliver on its promises for high reliability. These mechanisms make use of techniques from database management systems, and were designed and implemented to in a self-contained, portable fashion. *In this project, we will develop these mechanisms further and customize them to the special needs of optimization codes.* For instance, we will exploit the fact that optimization tasks can usually be checkpointed in a compact fashion, without storing the entire state of the task.

3.3 Network-Aware Interfaces to metaNEOS

In designing user interfaces to metaNEOS, we must take into account the needs of traditional users and also provide new easier-to-use interfaces that more fully exploit the interactive, immediate nature of the metacomputing environment. We propose to focus on interfaces of the following types.

Web, Windows, and Email interfaces. The existing NEOS Server uses a variety of network-aware interfaces that allow users to submit data through a Web page, an email message, or a Tcl/Tk Windows application. These interfaces prompt users for parameter values, convergence tolerances, and the locations of code fragments and data files on their local file systems. By this process, users are not only given access to an extensive collection of software and hardware resources for problem solving, but they are relieved of the drudgery of writing driver code and of linking all the pieces of data and code on their local system. Moreover, NEOS comes with a complete set of tools that make it easy for developers of the optimization software to build these interfaces.

Functional interfaces: Procedural languages. Traditionally, optimization software is invoked by calling a subroutine library from user-written "driver" code. By relinking the user code with a communications library instead of an optimization library, the call to a local library routine can be replaced by a call to metaNEOS, transparently to the user. Legacy code therefore can be reused with little or no alteration. The communications library routines handle packing and unpacking of data, and uses elements of Globus' Nexus system to encrypt and transmit the data and results.

Functional interfaces: Modeling languages. Many operations research practitioners now develop complex models using either standard spreadsheet and simulation tools or an optimization modeling language such as GAMS and AMPL. These tools allow users to define their problem in more intuitive terms than a piece of Fortran or C code would allow. By developing interfaces between these modeling tools and metaNEOS, we will retain the ease-of-use advantages while adding greatly to the computational power available to users. It will become possible to specify more complex and realistic models and to experiment with a greater range of "what-if" variations in the limited time available to the practitioner.

In the case of AMPL, for instance, we will support and extend two modes of use. The first mode will allow a user to submit a complete AMPL model, along with data files, to metaNEOS, and receive a file containing the optimal solution in AMPL format. In the second mode, a user will execute an AMPL session on their workstation to set up the model, data, and options. However, their "solve" command will be serviced remotely by metaNEOS rather than by a local library code, similarly to the Fortran and C interfaces outlined above. We will extend this mode to allow the user to issue non-blocking calls to solvers, allowing them to spawn multiple models to be solved simultaneously by the metacomputer.

Client-Server Paradigms. The existing NEOS Server is fundamentally a batch device: After submitting a problem, a user has no further interaction with the optimization task until the results have been computed. For many applications, however, a more interactive mode of operation is more appropriate. The user could be called on to steer the optimization task while it executes (e.g., in the global optimization application of Section 2.2), or to refine the model interactively in response to provisional results, or to specify some portion of the model to be optimized while fixing the rest (e.g., optimizing just the stock portion of a portfolio model, or the front door panel of a car body design). In all cases, the user's workstation—the client—is responsible for the interface, data gathering, and possibly some of the calculation, while metaNEOS acts as the server, performing optimization tasks on request.

Most such interactive modes are application-specific, and in this project we will not generally concern ourselves with the design of client models for specific applications. We are responsible, however, for identifying the commonality between various client-server modes, and for incorporating the middleware to support these modes into metaNEOS. In fact, the main issues have already been identified. First, we need to support two-way interaction between metaNEOS resources and remote clients. This issue also arose in the volatile data applications of Section 2.3; it is best dealt with by including in metaNEOS the tools from Globus' Nexus that perform encrypted, portable communication in heterogeneous distributed environments. Second, we need to support existing specialized optimization interfaces (for example, spreadsheets, graphical interfaces), so that the calls to the optimization routines in these applications can be replaced by calls to metaNEOS. Here, the functional procedural-language interfaces will provide the means for converting these applications for use with metaNEOS tools.

3.4 Security Issues

MetaNEOS introduces a number of challenging security problems. A potentially large population of users need to access computational resources at many sites, and in general we cannot assume a prior trust relationship. In addition, users and sites may both have complex policy requirements. For example, some users may need the integrity and/or privacy of their data to be protected, and others may want to be certain that their optimization problem is solved by "certified" software. Some sites may want to restrict access to certain users, or classes of users.

We plan to address these problems by building on, and extending in significant ways, the technologies being developed in the Globus project [25]. Globus uses public key technology to permit mutual *authentication* of "users" and "servers," and addresses key management issues that arise when a single computation may involve many processes and span many computers and sites. However, Globus does not take a very sophisticated approach to authorization, assuming that this is handled out-of-band by means of user accounts established on each resource that a user may wish to use. We plan in the MetaNEOS project to remove the need for such strong prior trust relationships. In a first stage, we shall map global user ids to a set of local ids managed by MetaNEOS. In a second phase, we shall explore the use of cryptographic certificates that encode "use conditions" [39] to control access. (Similar techniques are being considered by the financial services industry.) In principle, this approach can allow sites to restrict access to users who are certified to have certain characteristics (e.g., to work for a particular institution), and for users to request that their problems only be solved at sites that are certified to have certain properties. However, there are many difficult issues to be resolved before these techniques can be used in practice.

This core public key infrastructure will also allow us to integrate integrity and encryption mechanisms for user data [24] and to support cryptographic signatures on optimization software. It also provides a basis on which to address the difficult problem of accounting, although that is not a focus of this research.

References

[1] Globus Resource Manager Specification. Globus working note, available at `http://www.globus.org/scheduler/grm_spec.html`, 1997.

[2] Globus Security Policy and Requirements. Globus working note, available at `http://www.globus.org/security/policy.htm`, 1997.

[3] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application-level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing '96*. ACM Press, 1996.

[4] D. Bienstock. Experiments with a network design algorithm using ϵ-approximate LP solutions. Submitted for publication, 1996.

[5] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited-memory methods. *Mathematical Programming, Series A*, 63:129–156, 1994.

[6] S. Ceria, P. Nobili, and A. Sassano. A Lagrangian-based heuristic for large-scale set covering problems. Submitted for publication, 1995.

[7] K. M. Chandy and I. Foster. A deterministic notation for cooperating processes. *IEEE Transactions on Parallel and Distributed Systems*, 6(8):863–871, 1995.

[8] K. M. Chandy and S. Taylor. *An Introduction to Parallel Programming*. Jones and Bartlett, 1991.

[9] K. Mani Chandy, Ian Foster, Charles Koelbel, Ken Kennedy, and Chau-Wen Tseng. Integrated support for task and data parallelism. *International Journal of Supercomputer Applications*, 8(2):80–98, 1994.

[10] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1975.

[11] The PORTS Consortium. The PORTS0 interface. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1994.

[12] T. DeFanti, I. Foster, M. Papka, R. Stevens, and T. Kuhfuss. Overview of the I-WAY: Wide area visual supercomputing. *International Journal of Supercomputer Applications*, 10(2):123–130, 1996.

[13] J. R. Eyre. Variational assimilation of remotely sensed observations of the atmosphere. In *The Second International Symposium on Assimilation of Observations in Meteorology and Oceanography*, pages 639–645. World Meteorological Organization, 1995.

[14] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proc. 56h IEEE Symp. on High Performance Distributed Computing*. IEEE Computer Society Press, 1997.

[15] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proc. 56h IEEE Symp. on High Performance Distributed Computing*. IEEE Computer Society Press, 1997.

[16] I. Foster. *Designing and Building Parallel Programs*. Addison-Wesley, 1995. Also at: `http://www.mcs.anl.gov/dbpp/`.

[17] I. Foster. Compositional parallel programming languages. *ACM Transactions on Programming Languages and Systems*, 18(4):454–476, 1996.

[18] I. Foster, B. Avalani, A. Choudhary, and M. Xu. A compilation system that integrates High Performance Fortran and Fortran M. In *Proceedings of the 1994 Scalable High Performance Computing Conference*, pages 293–300. IEEE Computer Society Press, 1994.

[19] I. Foster and K. M. Chandy. Fortran M: A language for modular parallel programming. *Journal of Parallel and Distributed Computing*, 26(1):24–35, 1995.

[20] I. Foster, J. Geisler, C. Kesselman, and S. Tuecke. Multimethod communication for high-performance metacomputing applications. In *Proceedings of Supercomputing '96*. ACM Press, 1996.

[21] I. Foster, J. Geisler, C. Kesselman, and S. Tuecke. Managing multiple communication methods in high-performance networked computing systems. *Journal of Parallel and Distributed Computing*, 40:35–48, 1997.

[22] I. Foster, J. Geisler, W. Nickless, W. Smith, and S. Tuecke. Software infrastructure for the I-WAY high-performance distributed computing experiment. In *Proc. 5th IEEE Symp. on High Performance Distributed Computing*, pages 562–571. IEEE Computer Society Press, 1996.

[23] I. Foster, J. Geisler, and S. Tuecke. MPI on the I-WAY: A wide-area, multimethod implementation of the Message Passing Interface. In *Proceedings of the 1996 MPI Developers Conference*, pages 10–17. IEEE Computer Society Press, 1996.

[24] I. Foster, N.T. Karonis, C. Kesselman, G. Koenig, and S. Tuecke. A secure communications infrastructure for high-performance distributed computing. In *Proc. 56h IEEE Symp. on High Performance Distributed Computing*. IEEE Computer Society Press, 1997.

[25] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 1997. To appear.

[26] I. Foster, C. Kesselman, and M. Snir. Generalized communicators in the Message Passing Interface. In *Proceedings of the 1996 MPI Developers Conference*, pages 42–49. IEEE Computer Society Press, 1996.

[27] I. Foster, C. Kesselman, and S. Tuecke. The Nexus approach to integrating multithreading and communication. *Journal of Parallel and Distributed Computing*, 37:70–82, 1996.

[28] I. Foster, D. Kohr, R. Krishnaiyer, and J. Mogill. Remote I/O: Fast access to distant storage. In *Proc. IOPADS'97*, 1997.

[29] I. Foster, D. Kohr, R. Olson, S. Tuecke, and M. Xu. Point-to-point communications using migrating ports. In *Languages, Compilers, and Run-time Systems for Scalable Computers*, pages 199–212. Kluwer Academic Publishers, 1995.

[30] I. Foster, D. R. Kohr, Jr., R. Krishnaiyer, and A. Choudhary. Double standards: Bringing task parallelism to HPF via the Message Passing Interface. In *Proceedings of Supercomputing '96*. ACM Press, 1996.

[31] I. Foster, D. R. Kohr, Jr., R. Krishnaiyer, and A. Choudhary. MPI as a coordination layer for communicating HPF tasks. In *Proceedings of the 1996 MPI Developers Conference*, pages 68–78. IEEE Computer Society Press, 1996.

[32] I. Foster, D. R. Kohr, Jr., R. Krishnaiyer, and A. Choudhary. A library-based approach to task parallelism in a data-parallel language. *Journal of Parallel and Distributed Computing*, 1997. To appear.

[33] I. Foster and R. Olson. A guide to parallel and distributed programming in nPerl. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1995. `http://www.mcs.anl.gov/nexus/nperl/`.

[34] I. Foster, R. Olson, and S. Tuecke. Productive parallel programming: The PCN approach. *Scientific Programming*, 1(1):51–66, Fall 1992.

[35] I. Foster, M. Papka, and R. Stevens. Tools for distributed collaborative environments: A research agenda. In *Proc. 5th IEEE Symp. on High Performance Distributed Computing*, pages 23–28. IEEE Computer Society Press, 1996.

[36] I. Foster and S. Tuecke. Enabling technologies for web-based ubiquitous supercomputing. In *Proc. 5th IEEE Symp. on High Performance Distributed Computing*, pages 112–119. IEEE Computer Society Press, 1996.

[37] I. Foster and D. W. Walker. Paradigms and strategies for scientific computing on distributed memory concurrent computers. In *Proc. of the High Perf. Comp. 1994 Conf.*, La Jolla, CA, April 1994.

[38] M. D. Grigoriadis and L. G. Khachiyan. Fast approximation schemes for convex programs with many blocks and couping constraints. *SIAM Journal on Optimization*, 4:86–107, 1994.

[39] W. Johnston and C. Larsen. A use-condition centered approach to authenticated global capabilities: Security architectures for large-scale distributed collaboratory environments. Technical Report 3885, LBNL, 1996.

[40] F. X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects. *Tellus*, 38A:97–110, 1986.

[41] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[42] S. More, A. Choudhary, I. Foster, and M. Xu. MTIO: A multi-threaded parallel I/O system. In *Proceedings of the 11th International Parallel Processing Symposium*, pages 368–373. IEEE Computer Society Press, 1997.

[43] S. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proc. 32nd Annual Symp. On Foundations of Computer Science*, 1991.

[44] X. Zou, I. M. Navon, M. Berger, P. K. H. Phua, T. Schlick, and F. X. Le Dimet. Numerical experience with limited-memory quasi-Newton and truncated Newton methods. *SIAM Journal on Optimization*, 3:582–608, 1993.