Project Description

Next-Generation Servers for Optimization as an Internet Resource

Large-scale optimization has been a subject of investigation for over 50 years, but the challenge of making it useful in practice has continued to the present day. Initially the primary difficulties were posed by *computation,* but breathtaking increases in computer power and algorithm sophistication combined to allow for routine solution of large problems arising in practical applications [3]. As computational needs were addressed, the more serious difficulties came to be posed by *representation,* as modelers found that they could solve larger problems than they could manage or understand [15, p. 169]. This challenge, too, was eventually met, by increasingly sophisticated modeling languages and systems for describing and working with optimization problems [12, 26].

The primary difficulty of large-scale optimization has now shifted again, to one of *communication.* Increasing numbers of optimization algorithms are implemented increasingly well, but prospective users are unaware of these "solvers" or do not see the potential benefit that would justify obtaining and installing them. Only certain combinations of solvers and modeling systems work with each other, moreover, and modeling language support is slow to keep up with solver extensions to new problem types.

The Internet is now providing an increasingly practical way of addressing communication problems in large-scale optimization [19]. Websites offer abundant solver information [16], to be sure, but the more significant advance is the ability to send optimization problems over the Internet for submission to a solver at some remote site. The remote optimization "server" can address numerous problem types and can provide varied solvers for problems of each type, giving modelers much more of a choice than they could hope to have locally. In previous work under the auspices of the Optimization Technology Center of Northwestern University and Argonne National Laboratory, we have studied and experimented with the concept of an optimization server through the creation of the NEOS Server [6, 9, 24], which makes nearly 50 solvers available via a broad variety of network interfaces.

The current NEOS Server only begins to address the communication difficulties of large-scale optimization, however. The Server cannot tell users which solvers are appropriate for a problem that has been submitted, or choose a solver host based on the expected resource needs of a problem. Connections from modeling languages to solvers are still incomplete, and support for benchmarking is limited. Because NEOS has evolved along with the Web and the Internet — its first interface, through e-mail, dates back to 1996 — it is limited to some degree by early design decisions.

The research that we propose is thus motivated by our vision of a next-generation NEOS Server that addresses outstanding challenges of communication in large-scale optimization. This work will address design as well as implementation issues posed by standardizing problem representations, automating problem analysis and solver choice, working with new web-service standards, scheduling computational resources, benchmarking solvers, and verification of results — all in the context of the special requirements of large-scale computational optimization. Our research in these areas is timely, being motivated by new standards for web services and by the recent success of the NEOS Server itself, and will build on the considerable expertise in optimization servers already in place at the Optimization Technology Center.

The remainder of this introduction addresses the *broader impact* of the Optimization Technology Center, the NEOS project, and specifically the NEOS Server. The four major aspects of the proposed research are then discussed in four sections, roughly according to the order in which they would affect a typical new user:

- §1. ***Representing:***
  new standard forms for optimization problem instances
- §2. ***Analyzing and categorizing:***
  procedures for guiding prospective users in their choice of solvers
- §3. ***Scheduling:***
  new web service standards and their use in distributed, intelligent assignment of optimization requests to resources
- §4. ***Benchmarking and verifying:***
  services for automatically making benchmark runs and comparing results

The work in each of these areas will help to advance the others as well, as we will indicate. Other key researchers in these areas will collaborate on some aspects of the research, as noted at the end of each section.

A final section (§5) presents the standard Results from Prior NSF Support for the principal investigators.

The Optimization Technology Center. The proposed work will be carried out under the auspices of the Optimization Technology Center (`www.ece.northwestern.edu/OTC`), a joint endeavor of Northwestern University and Argonne National Laboratory. The Center is devoted to research in numerical optimization, in Internet and distributed computing, and in problem-solving environments, and to the study of optimization in a wide range of applications.

The two principal investigators for the proposed project are the current Northwestern and Argonne co-directors of the Center. We expect that extensive collaboration between Northwestern and Argonne members of the Center will continue in carrying out this research, with members of each institution spending some time at the other. Because Argonne is a national laboratory, however, this proposal's budget does not include direct support for Argonne researchers; instead the supplementary documentation provides a statement of participation on behalf of Argonne.

The NEOS Project. The Network Enabled Optimization System project has been a major activity of the Optimization Technology Center since the Center's founding in 1994. The continuing goal of the NEOS project is to make optimization a part of the worldwide software infrastructure that supports science and commerce. To this end, the NEOS Guide (`www.mcs.anl.gov/otc/Guide`) includes online examples of optimization problems, listings of test problem collections, and surveys of publications and software. The complementary NEOS Server (`www-neos.mcs.anl.gov`) provides remote access to solvers and so is the focus of the proposed research.

The NEOS Server currently supports nearly 50 solvers. Collectively these solvers accept about a dozen different kinds of input, ranging for example from function definitions in programming languages (Fortran, C, Matlab) to explicit problem instance descriptions (MPS, LP, sparse SDPA) to symbolic modeling language descriptions (AMPL, GAMS). A callable interface, Kestrel [8], also permits direct access to many of the NEOS solvers from within modeling systems' environments.

Usage of the NEOS Server (Figure 1) has grown to an average level of about 6500 submissions per month; peak loads of 4000 in a week have been handled without difficulty. Submissions have leveled off in the past year; this has motivated us to direct

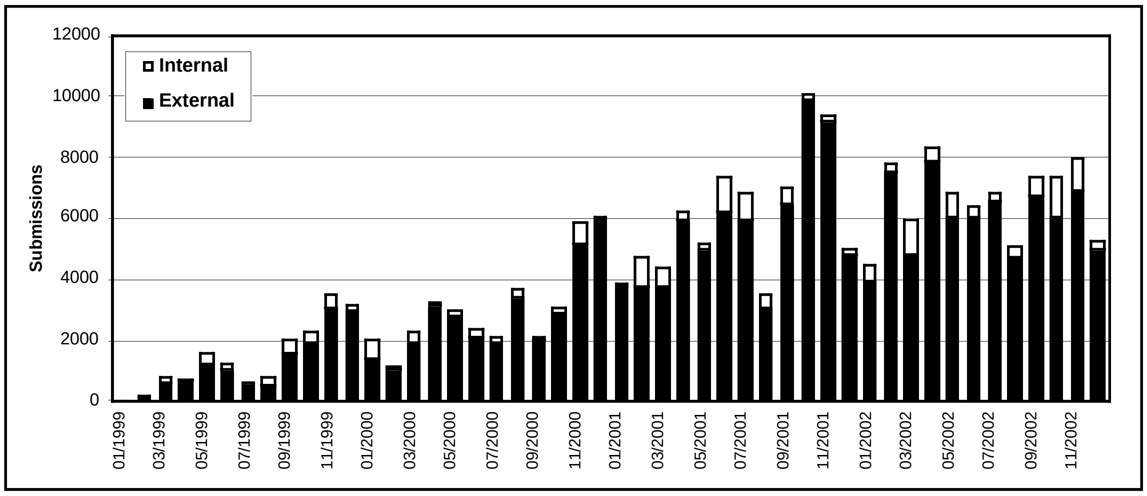


Figure 1: Monthly total submissions to the NEOS Server since 1999. "Internal" submissions are those from the domains of Argonne (`anl.gov`) and Northwestern (`nwu.edu`).

some of the proposed research, particularly in §2, toward making the Server easier to use for those who are not solver experts. We have also arranged to give NEOS tutorials this May at the INFORMS Conference on OR/MS Practice and the Annual Conference of the Institute of Industrial Engineers.

The ready availability of optimization tools has broad impact both directly through its role in a great variety of business and scientific applications, and indirectly by improving the quality of research and education in optimization modeling. Comments from existing NEOS Server users, excerpted in Figure 2, testify to the Server's appeal to potential practitioners of many kinds. The Server's varied offering of solvers and interfaces also tends to ensure that it addresses a broad base of needs.

The idea of an optimization server is beginning to have an influence beyond the NEOS project, moreover. A copy of the NEOS software has been set up at Sandia National Laboratory, for example, and a Northwestern graduate student has been hired as an intern to help with development of an internal optimization server at Motorola, Inc.

1. Representation standards for optimization problem instances

Even a cursory look at the NEOS Server's list of solvers (Figure 3) reveals the babel of input formats recognized by current optimization software. There are about 10 different low-level formats — ones that describe problem instances — recognized by one or another solver in the NEOS lineup, including MPS and LP formats for linear and integer programming, SMPS extensions to the MPS format for stochastic programming, formats such as SDPA specific to semidefinite programming, and DIMACS min-cost flow and other formats for network linear programming. Other solvers recognize input programmed as functions in various languages including Fortran, C, C++, and Matlab.

To the extent that there is any greater degree of standardization, it is through the use of input written in higher-level optimization modeling languages. Although NEOS works with the GAMS [2, 4] and AMPL [17, 18] languages, however, each of these supports only some of the available solvers. An arrangement that applies AMPL solvers to GAMS models is at best a stopgap, requiring execution of both the AMPL and GAMS compilers.

I have been working on a system for protein structure prediction. As part of my system, I had need to incorporate a nonlinear programming solver to handle packing of sidechain atoms in the protein. ... I was able to evaluate solutions provided by different nonlinear solvers ... in effect testing the code before deciding which to acquire and attempt to link into my system.

I am regularly suggesting my students to use NEOS as soon as their projects in AMPL cannot be solved with the student edition. So they debug their AMPL models locally ... and then they run their real-life projects thanks to NEOS.

Our idea is trying to design antennas by using the computer ... We have tried various solvers on NEOS to see if this is possible at all ... The NEOS server ... gives us the opportunity to find out which approach works best. Without this server it would probably be impossible to find out what strategy is most likely to succeed.

This is a great resource for those of us who need to demonstrate large scale optimization to students, but who do not have enough need for this (or funds!) to justify the purchase of a large scale optimization package.

It has greatly helped in the work I am doing here at General Motors. I have been able to build and solve a prototype combinatorial auction MIP model using AMPL and NEOS in a fraction of the time it would have required me to do this had I needed to requisition a solver and install it locally. Because of this, internal GM customers have been able to see the benefits of optimization in this business context, and will most likely give the go ahead for a full scale development project.

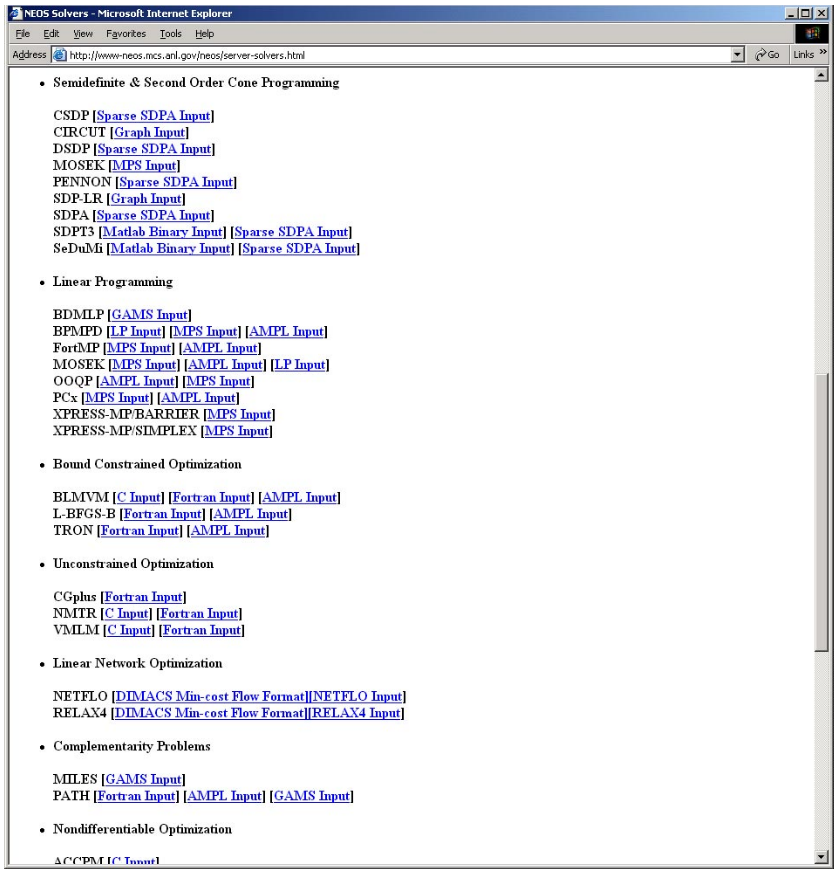
NEOS has been a very valuable tool in the two graduate optimization courses that I regularly teach. NEOS allows students to see a broader variety of solvers than we have available ...

I have been using the mixed-integer solvers at NEOS to study the complexity of phase retrieval, a problem encountered in several disciplines, principally crystallography and astronomy. My NSF grant has a budget for computer software, but not in the amount required to purchase some of the high end optimization software at NEOS. Another benefit provided by NEOS is that I am able to do serious number crunching from my humble laptop.

I didn't even know what nonlinear programming was and after I discovered what it was, it became clear how enormous a task it would be to solve the problems assigned to me. ... I had extremely complicated objective functions, both convex and nonconvex, an armload of variables, and an armload of convex, nonconvex, equality and inequality constraints, but when I sent off the information via the web submission form, within seconds I received extremely accurate and consistent results. The results were used for verifying a certain theory in my professor's research and so accuracy was extremely important.

I am attempting to find solutions to the modified Cahn-Hilliard equation, a fourth order partial differential equation by techniques to minimize the total free energy of the system. ...The submission process is straightforward and the availability of the NEOS solver has allowed me to concentrate more on the physics of our problem than the mechanics of writing optimization codes.

Figure 2. Users' descriptions of their applications of the NEOS Server, excerpted from a much more extensive list at `http://www-neos.mcs.anl.gov/neos/stories.html`.


NEOS Solvers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www-neos.mcs.anl.gov/neos/server-solvers.html
Go
Links

- Semidefinite & Second Order Cone Programming

  CSDP [Sparse SDPA Input]
  CIRCUT [Graph Input]
  DSDP [Sparse SDPA Input]
  MOSEK [MPS Input]
  PENNON [Sparse SDPA Input]
  SDP-LR [Graph Input]
  SDPA [Sparse SDPA Input]
  SDPT3 [Matlab Binary Input] [Sparse SDPA Input]
  SeDuMi [Matlab Binary Input] [Sparse SDPA Input]

- Linear Programming

  BDMLP [GAMS Input]
  BPMPD [LP Input] [MPS Input] [AMPL Input]
  FortMP [MPS Input] [AMPL Input]
  MOSEK [MPS Input] [AMPL Input] [LP Input]
  OOQP [AMPL Input] [MPS Input]
  PCx [MPS Input] [AMPL Input]
  XPRESS-MP/BARRIER [MPS Input]
  XPRESS-MP/SIMPLEX [MPS Input]

- Bound Constrained Optimization

  BLMVM [C Input] [Fortran Input] [AMPL Input]
  L-BFGS-B [Fortran Input] [AMPL Input]
  TRON [Fortran Input] [AMPL Input]

- Unconstrained Optimization

  CGplus [Fortran Input]
  NMTR [C Input] [Fortran Input]
  VMLM [C Input] [Fortran Input]

- Linear Network Optimization

  NETFLO [DIMACS Min-cost Flow Format][NETFLO Input]
  RELAX4 [DIMACS Min-cost Flow Format][RELAX4 Input]

- Complementarity Problems

  MILES [GAMS Input]
  PATH [Fortran Input] [AMPL Input] [GAMS Input]

- Nondifferentiable Optimization

  ACCPM [C Input]

Figure 3: Part of the NEOS Server's list of solvers and problem formats.

As part of our research, we propose to design a new low-level format that will be flexible enough to represent a broad variety of the optimization problems currently handled by the NEOS Server. Our representation will address problems that are not application-specific, but that are as specialized as network linear programs or as generalized as nonlinearly-constrained nonlinear programs. The adoption of such a format by solvers will make them more universally available through internet services. The adoption of the same format by modeling languages will enable solvers to more readily support many languages, moreover; the overall effect will be to decouple language and solver choice, letting the user pick the best tools for any project.

Current circumstances are particularly favorable for a study of this sort. It is not only that services such as the NEOS Server demand more standardization. New principles and tools have emerged over the past few years to guide the design of standard forms for Internet communication of all kinds.

Forms based on XML, in particular, are being used for a wide variety of purposes, and we propose to investigate their application for communicating instances of optimization problems. An XML representation consists of data delimited by `<tags>`, much like an `html` representation of the content of a web page. New collections of XML tags can be defined for any specialized purpose, however, by specifying a *schema* (Figure 4). Given a schema, standard tools are available for parsing files that correspond to it, and for building libraries to display and manipulate the contents of these files [29, 30].

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://COIN/XML˙LPschema.xsd" ... >
 <xs:element name="mathProgram">
  ...
    <xs:element name="sparseVector" type="sparseVector"
           minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="sparseMatrix" type="sparseMatrix"
           minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="linearProgram" type="linearProgram"
           minOccurs="0" maxOccurs="unbounded">
     <xs:key name="columnKey">
      <xs:selector xpath=".//NS:columns/NS:column"/>
      <xs:field xpath="@columnIdx"/>
     </xs:key>
     <xs:key name="rowKey">
      <xs:selector xpath=".//NS:rows/NS:row"/>
      <xs:field xpath="@rowIdx"/>
     </xs:key>
    </xs:element>
    <xs:element name="linearProgramSolution" type="linearProgramSolution"
           minOccurs="0" maxOccurs="unbounded"/>
...
</xs:element>
...
<xs:complexType name="linearProgram"> ...
```

Figure 4. Excerpts from a prototype *schema* (see www.w3.org/XML/Schema) that defines an XML representation of linear programming problem instances. Ellipses indicate lines omitted.

Additional schemas can be included to provide optional extensions. Thus a standard for optimization can be enforced and can grow in a well-defined way to accommodate new problem types. This contrasts with the current situation, where for example parsers for the MPS standard [28] vary in details between implementations, interpreters of the SMPS standard [1] are even more varied, and no proposal for nonlinear extensions (see, for instance [25]) has caught on at all. One perceived disadvantage of XML is its verbosity — the considerable file space taken up by tags — but in fact the tags only increase file size by a constant factor, which can be considerably reduced by use of optional alternatives to an ASCII representation.

A preliminary phase of this work is already being undertaken in collaboration with Northwestern University graduate student Leonardo Lopes and Prof. Kip Martin of the University of Chicago. We are building a complete XML-based specification for instances of linear programs, together with a library of support routines, which we will interface with at least one modeling language and several solvers. The example in Figure 4 is taken from an early prototype.

Building on this initial experience, the proposed research will undertake a more ambitious project to design a standard representation that addresses all of the problem types supported through the NEOS Server, with sufficient flexibility to be extended to new types. This is an undertaking of a breadth and difficulty not undertaken previously in the area of optimization, and as a result we see it as a proper object of a research project. It complements our interest in extending modeling languages to new problem forms, by focusing on the design of a new low-level standard that can provide diverse higher-level modeling languages with a standard way of reaching solvers.

This work is also complementary to the design of OSI, a standard procedural interface to solvers currently being implemented under the auspices of the COIN-OR project (see `www-124.ibm.com/developerworks/opensource/coin/`). OSI provides a way of calling optimizers directly from applications, whereas our standard form is to be a representation of the *content* of optimization problem instances, which could be communicated to solvers in a variety of ways. We intend to use COIN-OR to publicize our work on this project, to attract additional collaborators and reviewers, and to distribute the interface library for our XML-based standard. We have already been instrumental in creating a COIN-OR mailing list for standard forms in optimization and in organizing (with the help of Robin Lougee-Heimer of IBM) an organizational meeting on this topic attended by about two dozen participants at the INFORMS November 2002 meeting.

2. Analyzing and categorizing optimization problems prior to solving

A new NEOS Server user typically begins at the website index screen, which presents a list of 13 problem types:

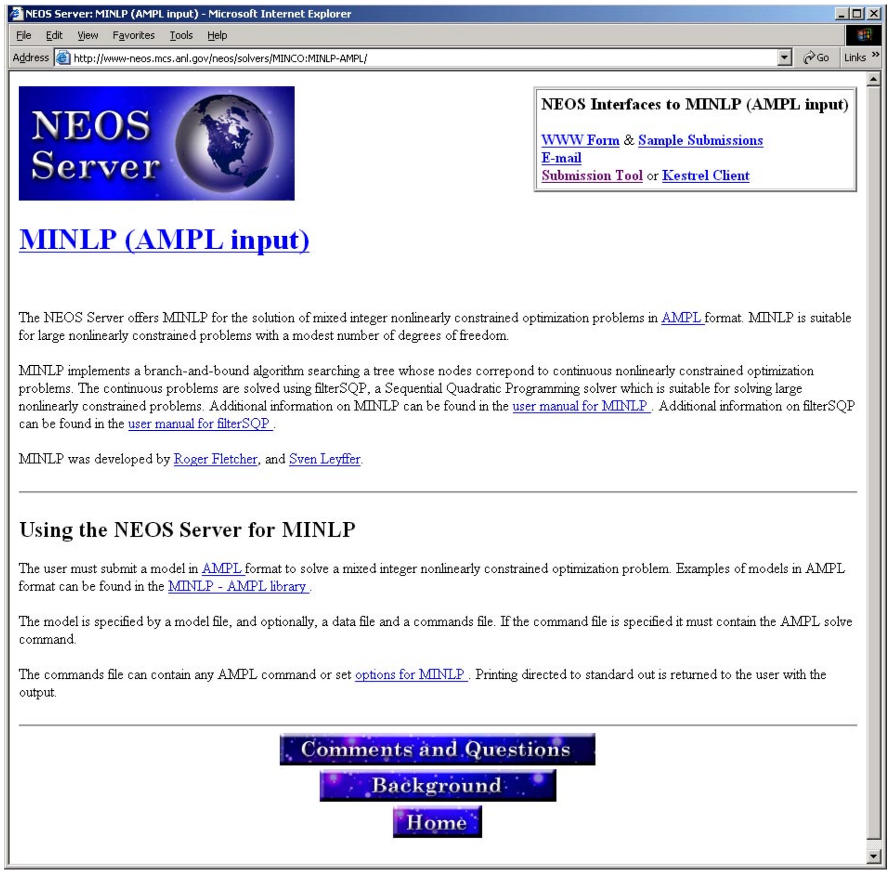
Semi-infinite Optimization	Unconstrained Optimization
Mixed Integer Nonlinearly Constrained Opt.	Linear Network Optimization
Mixed Integer Linear Programming	Complementarity Problems
Nonlinearly Constrained Optimization	Nondifferentiable Optimization
Linear Programming	Stochastic Linear Programming
Bound Constrained Optimization	Global Optimization
Semidefinite & Second Order Cone Progr.	

Each type links into a list of solvers and input formats (Figure 3). The choice among solvers is then up to the user. To provide some assistance in the choice, each solver has a main page with links to the NEOS Guide and to solver-specific documentation (Figure 5).

Although this arrangement has proved adequate for many purposes, unavoidably it burdens users with the job of determining a problem type and choosing a solver. Requests to our help line (`neos-comments@mcs.anl.gov`) suggest, in particular, that many potential users are analysts who have the training to build a model using a high-level modeling language, but who do not have the expertise to determine what category of model they have produced and what solvers are appropriate for it. The previously remarked leveling off of NEOS Solver requests (Figure 1) may reflect the difficulty of broadening the user base to include modeling and application domain experts who are not also algorithm and solver experts.

A description of an optimization problem instance already contains, at least implicitly, all of the information needed to properly categorize the problem. This principle underlies the design of *interactive* problem analyzers such as ANALYZE [23] for linear problems and MProbe [5] for nonlinear problems. Interactive analyzers rely on fairly sophisticated users, however, who are looking to better understand their problems with the aim of making their own determination of how best to solve them. In the context of the NEOS Server, we cannot be sure of as high a level of sophistication on the user's part, nor can we assume that the user is available to interact with the system online. We want to make an automated determination of problem characteristics, and of solver choice based on those characteristics.

The first part of our research in this area will thus concentrate on design of a problem analyzer for the NEOS Server. Initially we will base the analyzer on the "`.nl`" format of


NEOS Server: MINLP (AMPL input) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www-neos.mcs.anl.gov/neos/solvers/MINCO:MINLP-AMPL/
Go Links
NEOS Server
Comments and Questions
Background
Home

NEOS Interfaces to MINLP (AMPL input)

WWW Form & Sample Submissions
E-mail
Submission Tool or Kestrel Client

MINLP (AMPL input)

The NEOS Server offers MINLP for the solution of mixed integer nonlinearly constrained optimization problems in AMPL format. MINLP is suitable for large nonlinearly constrained problems with a modest number of degrees of freedom.

MINLP implements a branch-and-bound algorithm searching a tree whose nodes correpond to continuous nonlinearly constrained optimization problems. The continuous problems are solved using filterSQP, a Sequential Quadratic Programming solver which is suitable for solving large nonlinearly constrained problems. Additional information on MINLP can be found in the user manual for MINLP . Additional information on filterSQP can be found in the user manual for filterSQP .

MINLP was developed by Roger Fletcher, and Sven Leyffer.

Using the NEOS Server for MINLP

The user must submit a model in AMPL format to solve a mixed integer nonlinearly constrained optimization problem. Examples of models in AMPL format can be found in the MINLP - AMPL library .

The model is specified by a model file, and optionally, a data file and a commands file. If the command file is specified it must contain the AMPL solve command.

The commands file can contain any AMPL command or set options for MINLP . Printing directed to standard out is returned to the user with the output.

Figure 5: An example of a NEOS Server web page for a particular solver. with links to the NEOS Guide and to solver-specific documentation. The box at the top right provides links to the web interface and to instructions for other interfaces.

AMPL [22], which is already recognized by two dozen varied NEOS solvers. Later we will switch to the universal format that will be developed as described in §1.

The envisioned analyzer is actually a collection of analysis routines, each deriving some information of interest. In many cases, the information is in fact explicit in the problem instance description; the AMPL `.nl` format, for example, gives the numbers of variables figuring nonlinearly in the objective and in the constraints, so that nonlinear problems are immediately distinguishable from linear ones. About 15 values of this sort are provided in all [22]. Other problem characteristics can be unambiguously determined by fast algorithms applied to the problem instance. For example:

- ▷ Pure network flow problems can be detected by a scaling algorithm that alternately scans rows and columns of a linear constraint matrix.
- ▷ Quadratic objectives can be found by an algorithm that recursively walks the objective function expression tree to extract the coefficients of the Hessian matrix.
- ▷ Convex and concave quadratic expressions can be identified by a numerical test for positive semi-definiteness of the Hessian, such as an elimination algorithm.

Structure detection algorithms of these sorts need only process the whole problem when they are successful; they stop immediately when a failure occurs. Also some can be bypassed when not needed, so that for example a problem that has variables occurring nonlinearly in the constraints need not be tested for pure network constraints.

In addition to compiling a library of algorithms such as those above, we propose to tackle a more difficult analysis: the distinction of convex from non-convex problems. Because no algorithm for this purpose is both efficient and completely reliable, we must have efficient algorithms that approach the analysis from opposite sides:

- ▷ Convexity can be disproved by finding a counterexample. Randomly generated lines can be tested (as in MProbe) for violation of $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$, or randomly generated points can be tested for indefiniteness of the Hessian $\nabla^2 f(x)$.
- ▷ Convexity can be proved by recursively applying known properties — the sum of convex functions is convex, an increasing convex univariate function of a convex function is convex, and so forth — to an expression tree in the problem instance representation.

Both of these problems are harder than they seem at first, because it is usually desired to test convexity only over some convex constraint set. Many refinements to successive versions of MProbe's disprover, for example, have dealt with ways of sampling effectively over the feasible set. A convexity prover will also have to recursively propagate bound information in order to effectively apply the convexity properties. No effective prover yet exists, to our knowledge.

Applying both a prover and a disprover of convexity to a benchmark problem collection, we expect to find a small subset of test problems that appear convex on the basis of sampling but that cannot be proved convex given the properties our prover is able to apply. These will be candidates for further analysis, which will likely suggest further refinements to the sampling strategies for disproof or further properties to be tested for proof. In addition to hundreds of problems in existing nonlinear test libraries available in AMPL (see `www.sor.princeton.edu/~rvdb/ampl/nlmodels`), the NEOS Server receives a continuing stream of new trial problems.

The second part of the proposed research in this area concerns the determination of appropriate solvers, given a list of problem properties from the analyzer. We require a flexible approach that allows for adding properties and changing the mix of solvers without revising some program to reflect each change. This is a concern that has not been addressed by previous research, because dozens of solvers were not available from a single source prior to the advent of the Internet, the Web, and the NEOS Server.

Initially, we intend to experiment with a straightforward scheme that relies on a database that pairs solvers with problem types they can handle. Characteristics of a problem instance, determined from the analysis phase, will be used to automatically generate a query on the database that will return a list of appropriate solvers. Once this is running, we will consider extensions to generate lists ranked by degree of appropriateness. In a subsequent stage of the research we envision a more sophisticated mechanism that takes account of additional information that would be used by a solver expert; in this work we will aim to answer questions such as the following:

- ▷ How should solver recommendations deal with problem types that are subsets of other problem types? A general nonlinear optimization code might be a reasonable choice for bound-constrained optimization, for example, but not for linear programming.

▷ How can recommendations be extended to solvers' settings? Many solvers that offer significant algorithmic alternatives expect the user to decide between them, leaving the non-expert user in need of advice.

For these purposes, a straightforward database approach may not be adequate. The proposed research will consider more sophisticated ways of determining recommendations, such as through business rules systems (see `www.businessrulesgroup.org`).

We have made informal arrangements to collaborate in this work with the developer of MProbe, Prof. John Chinneck of Carleton University. Initial work on the convexity detection and solver recommendation projects is being undertaken by Dr. Dominique Orban, who has been appointed a postdoctoral fellow at Northwestern University through January 2004. We have also heard from Prof. Arnold Neumeier of some work underway to collect convexity properties, in a project at the University of Vienna, which may lead to some collaboration.

3. Web services for optimization

When the NEOS project was begun in 1995, the Web was just beginning to come into widespread use. At first the NEOS Server supported only low-level file formats or Fortran programs, and input only via e-mail; successive enhancements provided the much more powerful and convenient communication options available today (and described in the introduction to this proposal). To ensure reliability of the Server, this work used early and relatively mature standards, such as web forms, TCP/IP sockets for the NEOS Submission Tool (see `www-neos.mcs.anl.gov/neos/server-submit.html`) and CORBA for the Kestrel interface [8] (see also `www-neos.mcs.anl.gov/neos/kestrel.html`).

We are now seeing a new generation of standards that are designed to make *web services* more flexible in design and easier to build and maintain. One arrangement that has received particularly strong interest comprises three essential components (see also `www.enterprise-component.com/docs/cxsLesson5.pdf`):

◇ SOAP (Simple Object Access Protocol) allows calls to remote objects' methods and access to remote objects' data using standard web servers, the standard HTTP protocol for those servers, and XML to describe the call. SOAP is intended to serve as a more general and flexible successor to DCOM and CORBA.

◇ UDDI (Universal Description, Discovery, and Integration) is a specification for an online registry of web services. Providers can list their services in this registry, and users can seek out services by searching the registry in a standard way.

◇ WSDL (Web Services Description Language) defines the XML tags to be used in accessing a web service. Links to WSDL descriptions can be given through UDDI listings.

With tools like these, we can start to think about a more general and flexible optimization service environment. Developers and researchers might make their solvers available through the UDDI registry, using WSDL to define the problem information required. An XML-based standard for describing optimization problem instances (as we propose in §1) would fit nicely here. Only the XML tags for solver-specific options would need to be defined by individual providers of solver services.

Such an arrangement has the potential to substantially decentralize the registry of solver characteristics currently maintained for the NEOS Solver at Argonne National Laboratory. The remaining work of the centralized NEOS Solver would be focused on activities not specific to individual solvers, such as analyzing problems and recommending solvers (§2) and on providing multi-solver services such as benchmarking (§4 below) and translation (as with the current GAMS-to-AMPL modeling language translator).

This vision of a next-generation NEOS Server leaves open the question of how optimization "jobs" will be scheduled to run on available workstations. The current centralized scheme maintains one queue for each solver/format combination, along with a list of the workstations on which each solver can run. We will want to maintain this scheduling control, while at the same time making the scheduling decisions more distributed (like the solver services). We will also investigate extending the power of the NEOS scheduling schemes to take advantage of Grid computing [14], both in making use of idle computing power (as provided, for instance, by Condor [27, 13]) and in supporting the use of multi-processor optimization methods. In the case of the latter our work has especially great potential to stimulate new applications, by saving potential users the considerable difficulty of setting up the required hardware and networking software.

The special features of optimization serve to distinguish our research in this area from the routine design of new web services. Optimization runs are characterized by their huge and hard-to-predict consumption of processor time and memory space; only a modest increase in the instance size generated from an integer programming model, for example, can cause the solution time to increase from minutes to days, with a corresponding increase in the maximum size of the branch-and-bound tree. Predictions of resource requirements must take account of problem characteristics, since for instance a continuous linear program in hundreds of thousands of variables is generally much more tractable than an integer or nonlinear program of the same size.

We propose to study how categorization of optimization problem instances (as outlined in §2) together with statistics from previous runs can be used to improve upon the current scheduling decisions of the NEOS Server. As just one example, an intelligent scheduler should not assign two large jobs to a single-processor machine, since they will only become bogged down contending for resources; but a machine assigned one large job could also take care of a series of very small jobs without noticeable degradation to performance on either kind of job. Both the kind and size of optimization instances must be assessed in order to determine which should be considered "large" and which "very small" for purposes of this scheduling approach.

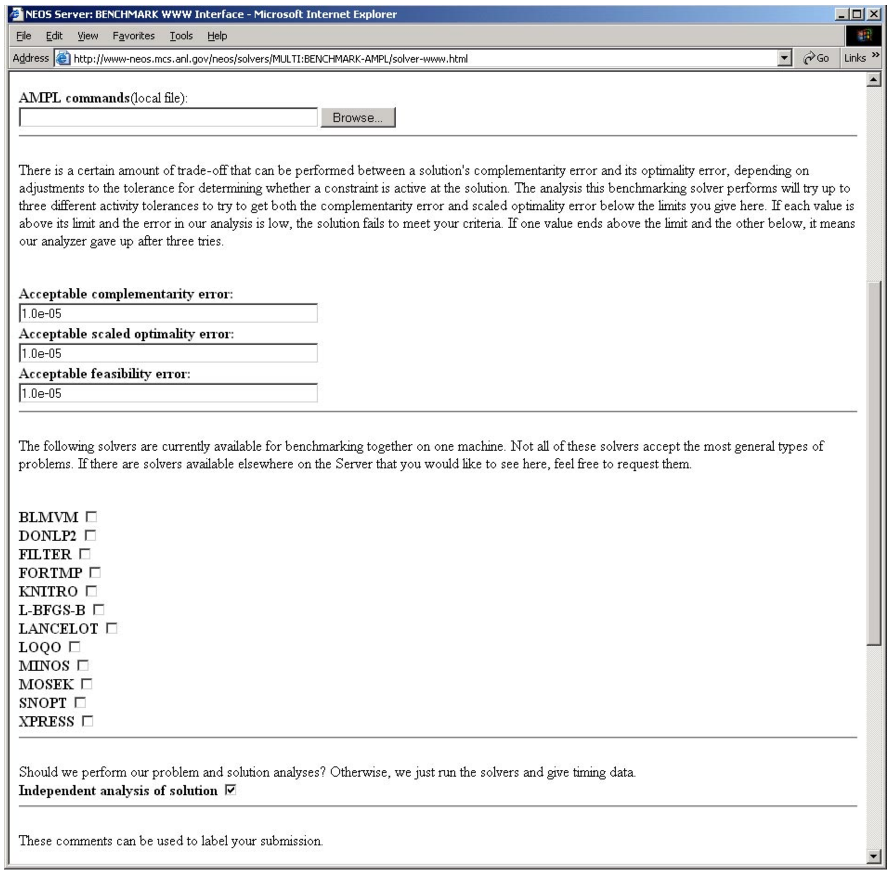
Northwestern graduate student Jun Ma is beginning dissertation research in this area. He has spent 9 months on an internship with an optimization group at Motorola, Inc., where he has built parts of an optimization server scheme for internal use. We are also discussing collaboration with Prof. Jeffrey Linderoth of Lehigh University, who was a major contributor to studies of optimization methods on computational grids in the earlier MetaNEOS Project (see `www.mcs.anl.gov/metaneos`).

4. Benchmarking and verification

The availability of more than one solver for many classes of problems makes the NEOS Server an obvious choice as a benchmarking tool. In fact the Server is potentially useful both in choosing a solver for a particular application and in comparing solvers generally. There are significant barriers to achieving these potentials, however, which motivate this part of the proposed research.

Someone who has developed a new model, but who is not sure which of the several applicable solver packages to apply, is often advised that the only way to be sure is to carry out some test runs on typical problem instances. The straightforward way to do this is to send each test instance to each candidate solver. But as NEOS makes no guarantee that separate runs will be done on comparable machines under comparable conditions, the results may say little about the relative efficiency of the solvers. The results may say more about the reliability of the solvers, but even so they may be distorted by differences in the memory available on the workstations devoted to different solvers, or by differences in time limits imposed by the owners of different workstations on which NEOS Server jobs run. There is not necessarily any obvious way to compensate for the differences between runs, moreover, because in general each solver is available on any of a number of dissimilar workstations, among which one is selected by the Server according to the load at the time a job is submitted.

As a first step in addressing these difficulties, we have added to the NEOS server a kind of "benchmarking solver." A user tells this benchmarker which solvers are to be compared (Figure 6) and which problem (in AMPL or GAMS) they are to be compared on. The benchmarker then applies all of the requested solvers — on the same computer — and returns concatenated listings of their results, along with a summary of problem statistics. For the case of smooth nonlinear problems, the benchmarker also optionally


NEOS Server: BENCHMARK WWW Interface - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www-neos.mcs.anl.gov/neos/solvers/MULTI:BENCHMARK-AMPL/solver-www.html
Go
Links
Browse...

AMPL commands(local file):

There is a certain amount of trade-off that can be performed between a solution's complementarity error and its optimality error, depending on adjustments to the tolerance for determining whether a constraint is active at the solution. The analysis this benchmarking solver performs will try up to three different activity tolerances to try to get both the complementarity error and scaled optimality error below the limits you give here. If each value is above its limit and the error in our analysis is low, the solution fails to meet your criteria. If one value ends above the limit and the other below, it means our analyzer gave up after three tries.

Acceptable complementarity error:
1.0e-05
Acceptable scaled optimality error:
1.0e-05
Acceptable feasibility error:
1.0e-05

The following solvers are currently available for benchmarking together on one machine. Not all of these solvers accept the most general types of problems. If there are solvers available elsewhere on the Server that you would like to see here, feel free to request them.

BLMVM ☐
DONLP2 ☐
FILTER ☐
FORTMP ☐
KNITRO ☐
L-BFGS-B ☐
LANCELOT ☐
LOQO ☐
MINOS ☐
MOSEK ☐
SNOPT ☐
XPRESS ☐

Should we perform our problem and solution analyses? Otherwise, we just run the solvers and give timing data.
Independent analysis of solution ☑

These comments can be used to label your submission.

Figure 6: Part of the web interface for the special benchmarking solver of the NEOS Server.

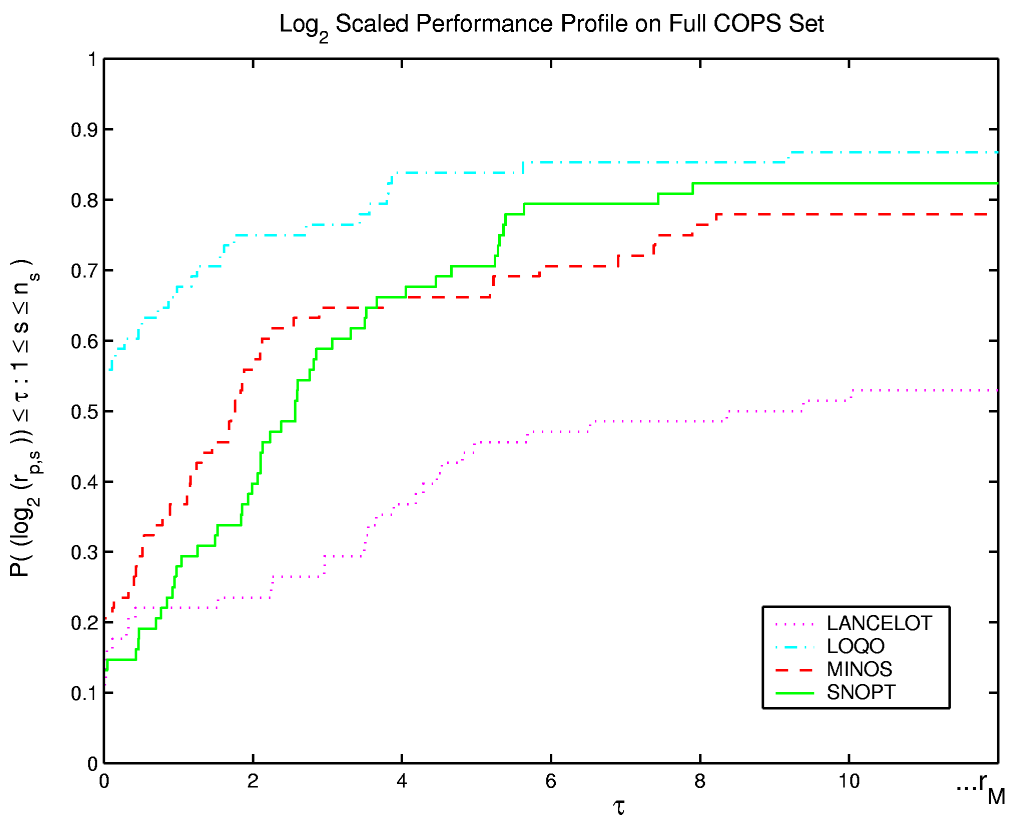


Figure 7: A performance profile [10] summarizing benchmark results from four solvers on a variety of test problems. Toward the left the curves emphasize speed of the solvers, while toward the right they place greater emphasis on reliability.

assesses the quality of each solver's solution with respect to complementarity, feasibility and optimality tolerances (which may be adjusted by the user) [11]. This innovative approach to solution verification is independent of any correctness claims or statistics made by individual solvers. In the proposed research, we will investigate connecting the analyzer described in §2 to the current benchmarker, so that the user is asked to choose only among solvers that are appropriate for the problem to be solved. Concurrently, we will further test and refine the verification methods in [11] and will extend them to handle a broader variety of problems and situations.

Benchmarking on only one problem can be misleading, so a number of sample problems from an application are often tested at the same time. Benchmark tests on large sets of problems from diverse applications are also common, for purposes of comparing the overall quality of different solvers. For this purpose we have developed the concept of a *performance profile* [10], which clearly shows the tradeoffs between speed and reliability of alternative solvers applied to a test problem set (Figure 7). This device has been favorably received and is being increasingly adopted by researchers for their computational comparisions of new algorithmic ideas. We will investigate the incorporation of this approach into the NEOS Server environment, with the aim of producing a benchmarker that takes a set of problems as input and produces statistics and performance profiles for appropriate solvers.

We intend our benchmarking tool to accept but not require guidance from the user, so that it is appropriate for use by practitioners as well as researchers. The measures of reliability reflected in the resulting performance profiles will make use of our verification approach to ensure that consistent standards are applied in comparing of solvers. The NEOS Server might then be able to automatically maintain benchmark results on available solvers for public test problem sets, re-running the benchmarker periodically to take account of updates or newly available solvers.

We have discussed this aspect of the proposed research with Hans Mittelmann of the University of Arizona, who has maintained an extensive website of optimization benchmarks (`plato.la.asu.edu/bench.html`) and has contributed a substantial number of the solvers currently accessible through the NEOS Server. We envision the possibility of considering at a later point a more formal collaboration.

5. Results of prior NSF support

Robert Fourer and Jorge Moré. "ITR: Advanced Application Service Provider Technologies for Large-Scale Optimization": *Grant CCR-0082807, $468,359 for September 2000 through August 2003.*

The NSF award most closely related to this proposal in the past five years is the same for both co-Principal Investigators. Thus their results of prior NSF support are given here together.

The NEOS Server. The greatest part of our research under this project has centered on the development of the NEOS Server as an application service provider for large-scale optimization problems. As a result of this work, today's NEOS Server (`www-neos.mcs.anl.gov`) is a collaborative project that provides access to dozens of academic and commercial optimization packages through an assortment of Internet interfaces. Over seventy thousand job requests are handled annually, including optimization problems from academic, commercial, and government institutions. Recent NEOS applications [9] include circuit simulation, protein folding, VLSI design, brain modeling, airport crew scheduling, and modeling of electricity markets. A full description of the use, design, and implementation of the NEOS Server and administrative tools is provided by the NEOS Administrative Guide [7].

Version 4.0 of the NEOS Server has an improved scheduling algorithm, and communications daemons that can track jobs for possible termination or verification. In particular, we ensure that NEOS jobs do not overwhelm the systems of collaborative institutions by providing numerous options to increase flexibility in setting limits on jobs. Examples of these options include file size limits, job time limits, limits on the number of submissions from one address per unit of time, and limits on the number of jobs that may run concurrently on a particular workstation.

We have also increased the level of fault tolerance for our communications package. We now make better use of the Internet Protocol to detect communication errors between the NEOS Server and remote solver stations and to return information to the user. Responding to communication faults, we can overcome a temporary loss of connectivity between the Server and solver communications handler and return completed results to the user as if no lapse had occurred.

The most recent of our studies of NEOS Server interfaces to bear fruit is Kestrel [8]. This CORBA–based interface is designed to send optimization problems generated by a local modeling language environment, AMPL [18] or GAMS [4], to the NEOS Server. The user chooses a solver from the Server's list of appropriate solvers and issues the modeling language's *solve* command as if the problem were going to be solved locally. Once the problem is solved on the remote system, the results are returned in the original (native) modeling format. The key feature of the Kestrel interface is that the native result format lets the user's modeling language interpret and manipulate the results directly, rather than parsing text output listings that vary greatly from one solver to the next.

Performance profiles. In considering the problems associated with an automated choice of algorithm, we developed the concept of performance profiles [10] for evaluating and comparing optimization software. The performance profile for a solver is the cumulative probability distribution function for a performance metric. We have shown that performance profiles provide a means of visualizing the expected performance difference among many solvers, without any of the deficiencies of previous approaches.

The concept of performance profiles has been accepted by the optimization community as the method of choice for presenting comparative results among solvers or among variations of a solver. We expect that other communities will also welcome this approach in the near future.

Representation of stochastic programming models. The NEOS Server offers a few solvers for stochastic programming problems, but their usefulness is limited by the lack of good tools for describing the underlying models symbolically. Some aspects can be described through conventional algebraic modeling languages, but others — notably the multistage *scenario trees* — are not inherently algebraic and require substantially different interpretations.

Northwestern graduate student Leonardo Lopes is writing his PhD dissertation on this topic. He has shown how a simple, innovative extension to the AMPL modeling language can allow for *simplified* models of individual stages, which need not have explicit indices for either stages or scenarios. A working prototype of a system using this extension has been built, and will be used in subsequent empirical tests of people's ability to write stochastic programming models. A draft description of this work [21] is nearly ready to be submitted for publication.

An earlier paper in this project [20] describes a related system for transforming, aggregating, and relaxing stochastic programming problem instances to meet the needs of a variety of solvers. To be fully general, these aggregation routines require a fast heuristic for the minimum supertree problem. This kind of facility is ideal for use by an optimization server to make available a variety of solvers while permitting each user to employ only one input form.

Design of a server for interdisciplinary optimization. With additional support from internships at Motorola Inc., graduate student Jun Ma has investigated the design of a central optimization server for models whose functions must be evaluated at several remote sites. Individual sites may differ greatly in their efficiency and reliability, posing difficult problems for the optimizing algorithm.

As part of this work, Ma has done considerable investigation of web service protocols, gaining experience that will be highly useful in the proposed research.

Development of human resources. The prior NSF support has funded some preliminary research in analysis of optimization problems before solution, carried out by postdoctoral associate Dominique Orban and graduate student Jennifer Strehler. The work of Dolan, Lopes, and Ma cited above has also been supported in part by the prior grant.