from *SIAM News*, Volume 35, Number 6, July/August 2002

# Optimization on the NEOS Server

*By Elizabeth D. Dolan, Robert Fourer, Jorge J. Moré, and Todd S. Munson*

The NEOS Server for Optimization was unveiled in 1995 as one result of an effort to develop computational servers and collaborative technologies. Today, the NEOS Server (www-neos.mcs.anl.gov) is a multi-institutional research effort providing access to more than fifty academic and commercial optimization packages through an assortment of Internet interfaces. More than seventy thousand job requests are handled annually, including optimization problems from academic, commercial, and government institutions. In this article, we give an overview of the NEOS Server, chronicle improvements found in Version 4 that make it faster and easier to use, and shed some light on actual applications that have been optimized with the NEOS Server.

## Overview

We designed the NEOS Server to automate the most tedious tasks faced by users of optimization libraries. The Server's goal is to specify and solve optimization problems with minimal user input. In particular, derivatives and other auxiliary information are computed automatically; the optimization problem is linked with software libraries by solver-specific driver scripts, and computing resources are provided for NEOS jobs. This automation of chores leaves users free to concentrate on modeling their problems, rather than on the solution process.

The basic mechanics of solving a problem with NEOS start when the user writes a specification for some problem and submits it through one of the established NEOS interfaces (see Figure 1). On receiving a problem specification, the Server assigns the job a number and password and queues it for scheduling on one of our remote machines. Once the job is received by the remote machine, the submission is unpacked, the driver processes the submission (which includes calculation of any necessary derivatives and linking of the algorithm libraries and objects), and the optimization code executes. The remote machine reports results to the Server, which sends them to the user interface. We discuss the interfaces, new tools for monitoring jobs, and available resources in the following sections.

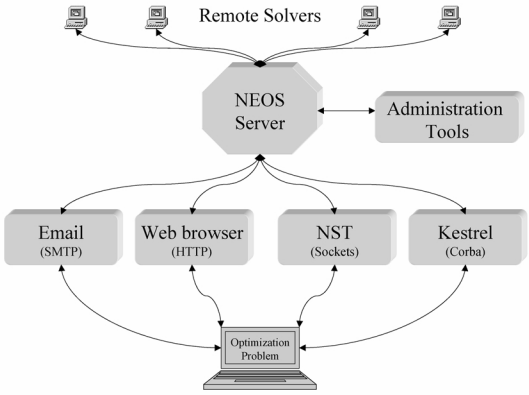


**Figure 1.** *The NEOS Server.*

Most of the NEOS solvers accept input in a programming language, a modeling language, or one of a wide variety of data-oriented formats. The solver pages and examples on the NEOS Server Web site [9] provide details on the accepted input formats. Job results also vary in structure. Typically, they consist of a listing of the optimization software output and the final solution vector.

A full description of the use, design, and implementation of the NEOS Server and administrative tools can be found in the NEOS Administrative Guide [2]. The guide covers a variety of issues, including installation of the Server, addition of solvers, and monitoring of use.

*Interfaces*

All problems are submitted to the NEOS Server by way of one of the interfaces shown in Figure 1. In each case, the Server processes problems in the same way; only the method of submission changes.

The original NEOS Server submission interfaces were based on e-mail and then Web browser technology. E-mail submission is efficient and basic. It consists of a text message specifying the input, with distinct file content delineated by prescribed tokens (e.g., `BEGIN.F, END.F`). The message is sent to neos@mcs.anl.gov, and the results are returned to the user in an e-mail message. Many users who have slow Internet connections or who do not wish to remain online throughout a long job rely on this interface.

The most popular (although not necessarily the fastest) interface for problem submission relies on a Web browser. Users access the NEOS Web site and select a solver. They then fill in a form with their problem data and click on the *submit* button. Intermediate solver out-put and the final results are sent to users' browsers for display.

We introduced the NEOS Submission Tool (NST) as an alternative to the Web browser interface. The NST GUI features fast socket connections with the NEOS Server and offers the option of saving submissions for reuse. It was initially implemented with Tcl/Tk and Perl, which worked well on Unix systems. Because many users submit jobs from non-Unix machines, however, we recently released a platform-independent NST in Java.

The NST, which runs on the user's machine, contacts the Server to generate a menu of available solvers. The user selects a solver; based on solver-specific data supplied by the Server, a submission form is built. The user fills in the form and clicks the *submit* button to send the problem specification to the Server. Intermediate output and the final results are displayed by the NST.

Our most recent addition to the NEOS Server interfaces is Kestrel [3, 5]. This CORBA-based interface is designed to send optimization problems generated in a local modeling language environment, either AMPL [1] or GAMS [6], to the NEOS Server. The user chooses a remote solver from the Server's list of appropriate solvers and issues the modeling language's *solve* command

as if the problem were 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, without parsing text results that might look different from solver to solver.

The Kestrel interface requires that the Server handle binary input files. Binary file submission is now available through the Web and NST interfaces for solvers that accept, say, Matlab binary submissions. Users can also upload compressed files. The Server can handle standard de-flation of file types `.z`, `.Z`, `.gz`, and `.zip`, although the submission specification also allows solvers capable of accessing compressed files directly to do so.

*Additional Tools*

Once a job has been submitted to the NEOS Server and a job number and password have been assigned, Web tools are available for checking the status of the problem or for terminating the job. Web access to current queues and to intermediate and final results for a job allows users of e-mail and Kestrel interfaces to view intermediate data; it also offers an alternative result-retrieval mechanism for users who lose connectivity with the NEOS site's Web server or the NEOS socket server. Kestrel users can also overcome network outages by re-connecting later with the Kestrel server.

The Kill-Job administrative solver requires a job number and password to request that a job be terminated. If the job is waiting to run, it is removed from the solver queue. If the job is already running on a remote solver station, the Kill-Job solver connects to the NEOS communications daemon on that station and asks that the job be killed.

*Resources*

Machine cycles are provided by participating institutions. At present, remote solvers are distributed at the following sites around the world:

- Aachen University of Technology
- Argonne National Laboratory
- Arizona State University
- National Taiwan University
- Northwestern University
- University of Wisconsin

Computations can be performed on dedicated machines or on general-use machines. Computing times are fast, even on general-use equipment, as the NEOS machines tend to have powerful processors and large amounts of memory. Many users prefer NEOS for solving larger versions of their problems, some of which may run for days on donated NEOS machines.

We have continued to add solvers to the Server, with a total of more than fifty now available. Specific applications solved and the most frequently used NEOS solvers are briefly discussed later. While no single site has all of the solvers installed, the Kestrel client allows some solvers to augment their abilities by accessing other solvers remote to them. One special solver harnesses the translator created by GAMS Development Corporation to solve GAMS submissions with any Kestrel AMPL solver. This translating solver internally calls on the Kestrel interface to submit the generated AMPL model. Results are returned to the user in GAMS format.

## Enhancements

We have worked steadily to improve the NEOS server since its initial release. Versions 1 and 2 were not portable. Version 3 was designed to be portable across Web and e-mail servers on the Unix system. Other computational servers are dependent on their software environments and cannot readily be moved. With the recent release of Version 4, we have also addressed performance and fault-tolerance issues.

*Efficiency*

As the number of jobs submitted to the NEOS Server has increased, we have had to become more efficient in managing the available resources.

One modification along these lines has been the replacement of *client-pull* with *server-push* Web technology. As a result, more NEOS jobs can run simultaneously, without overwhelming the Web server with requests for information that may not be available. The former client-pull structure forced clients to query the Web server for intermediate job results every few seconds to get an idea of real-time progress, resulting in tens of thousands of useless requests per month. Server-push techniques allow the Web server to transfer data as it becomes available over a sustained connection with the client. The NEOS Server now processes greater numbers of Web-interface jobs with only a fraction of the Web hits and subsequent load on the Web server machine required with the client-pull system.

To ensure that NEOS jobs do not overwhelm the systems of participating institutions, we provide numerous options to increase flexibility in setting limits on jobs. Examples include limits on file size, job time, and the number of jobs that can run on a particular workstation at one time. The NEOS communication daemons and scheduler have been rewritten to take these solver options into account. For example, jobs that will be rejected are quickly culled from the solver queues to make room for legitimate submissions.

*Robustness*

The communications package for remote solvers has also been revised to require fewer daemons. The old implementation

required one daemon for each solver on each remote machine. A machine that hosted many solvers could be inundated with daemon processes and monitoring processes. The new version requires only one daemon to service all the solvers hosted on any one machine. One benefit of this enhancement is that fewer communications daemons must be monitored and restarted in the event of failure.

An equally important task of the communications package is to maintain a level of fault tolerance. 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 users. 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.

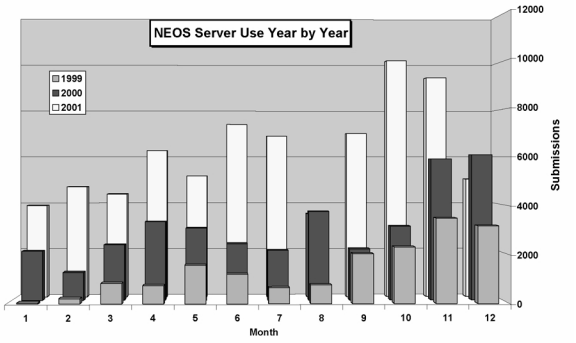


**Figure 2.** *Numbers of jobs submitted to NEOS, 1999–2001.*

## Statistics

Improvements to the NEOS Server have allowed us to handle an exponentially increasing number of submissions without a heavy investment in either maintenance or machines. This trend is clearly seen in Figure 2, which shows the number of jobs processed, by month, over the last three years. Although the numbers vary considerably on a monthly basis, our data indicates a steady increase in the average number of job submissions per month for the NEOS Server:

1999: 1400 jobs/month
2000: 3100 jobs/month
2001: 6200 jobs/month.

That is, the number of jobs roughly has doubled each year during this period.

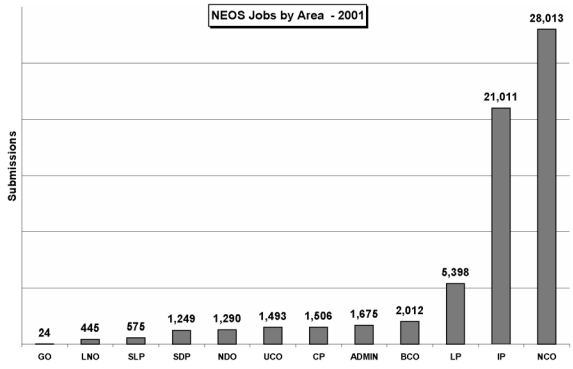


**Figure 3.** *NEOS jobs by area of optimization: GO, global optimization; LNO, linear network optimization; SLP, stochastic linear programming; SDP, semidefinite programming; NDO, nondifferentiable optimization; UCO, unconstrained optimization; CP, complementarity problems; ADMIN, NEOS administrative solvers; BCO, bound constrained optimization; LP, linear programming; IP, integer programming; NCO, nonlinearly constrained optimization.*

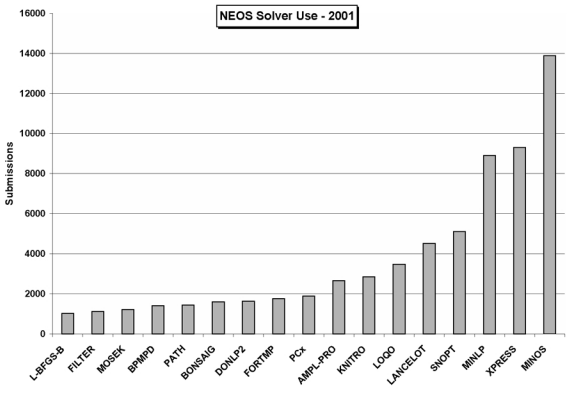


**Figure 4.** *Most frequently used NEOS solvers, 2001.*

The NEOS Server received about 74,000 job submissions in 2001. Percentages of jobs submitted by the various modes are as follows: e-mail, 11%; Kestrel, 12%; NST, 19%; and Web, 58%. As expected, most of our users relied on Web browsers for submitting their jobs. We suspect that users are reluctant to try the NST or Kestrel because of the required downloading and installation of an additional piece of software. In our view, the Java-enabled NST offers the best submission mode for general problems, but Kestrel offers the ultimate convenience for solving optimization problems within a modeling language. We suspect that the popularity of AMPL and GAMS for formulating optimization problems will translate eventually into a higher percentage of submissions with Kestrel.

### *Solver Usage*

The data collected during the life of the Server might be of interest to the optimization community at large, and in this section we present some of the more generic data. In particular, Figures 3 and 4 display sample compilations of NEOS submission characteristics for 2001.

Figure 3 shows the number of NEOS submissions per solver area (with the recently separated areas of mixed-integer nonlinearly constrained optimization and mixed-integer linear programming recombined into integer programming for the graph). This figure clearly shows that most of the jobs received by the Server were for nonlinearly constrained optimization (NCO) solvers, closely followed by those for integer programming (IP) solvers. We caution, however, that these statistics can be misleading about the problems themselves—a solution to a linear programming problem, for example, can be found with a solver from either the NCO or the IP category.

Still, we are surprised to see the relatively small number of submissions to semidefinite programming (SDP) solvers, as this is an area of significant current interest to the optimization community. We speculate that semi-definite programming is simply experiencing the fate of most new areas: Users have yet to understand how to pose their problems as semidefinite programs, and the lack of support for SDP solvers in popular modeling languages likely discourages submissions. Until these situations are addressed, we are likely to see interest center on more established areas.

Figure 4 gives an idea of the numbers of jobs submitted to some of the most popular solvers on NEOS. The numbers represent combinations of all submissions to a solver at any of its various interfaces, including different input formats and optimization areas.

## Sample NEOS Applications

Numbers of submissions to the NEOS Server reveal that the Server is being used, but they do not tell us anything about how it is being used. As it turns out, the NEOS Server handles a wide variety of optimization problems. Some of the projects we have heard about are listed below.

*Applications: 2000*

- Quality-of-service management in multimedia database systems.
- Estimating the value at risk of financial institutions.
- Image reconstruction of space objects.
- Optimal shift schedules for ground-handling activities in an airport.
- Design of a power cycle consisting of four heat exchangers, a turbine, and a pump.
- Transistor sizing for timing and noise reduction.
- Design of communication networks for large distributed networks.
- Modeling seat-assignment problems.
- Unilateral contact problems in engineering mechanics and robotics.
- Distance geometry and multidimensional scaling.
- Forecasting future purchasing behavior for telecommunications services.
- Finding energy functions for molecular structures.
- Evaluating the capacity of the binary-multiplying channel.
- Portfolio selection and scheduling.
- Designing a yagi antenna.
- Resource requirements for broadband networks.

*Applications: 2001*

- Circuit simulation with a large number of complex constraints.
- Predicting recent observations of Bose–Einstein condensates.
- Predicting globular protein folding.
- Low-power VLSI design.
- Settling conjectures for multiperson cooperation.
- Supply chain optimization.
- Limit-state analyses of large dams.
- Studying the brain's representation system.
- Modeling electricity markets.
- Designing fractional delay filters for LAN/WANs.
- Scheduling thermo- and hydro-energy resources.
- Building a crew-scheduling system for an airline in Indonesia.
- Solving the modified Cahn–Hilliard free-energy equation.
- Applying optimization to farming in Switzerland.
- Route selection in packet-switched networks.
- Finding bounds for the maximum cut value in sparse graphs.

## Future Directions

The NEOS project has evolved into an outstanding example of success in the areas of Internet-based computing, computational servers, and collaborative technology. Designed to provide optimization solutions to scientific computing problems, the NEOS Server has obviously found a niche in other areas as well. In the future, we plan to augment the NEOS capabilities based on our experiences.

Analyzing the data contained in all the problems that have been sent to the NEOS Server represents a major challenge. We have only scratched the surface with the statistics presented here. For example, we are frequently asked whether we could build a benchmarking collection from a subset of these problems. The idea of having test problems is appealing, but we still need to develop methods for examining and sorting through thousands of submissions for those with the characteristics of good test problems.

Some users are now working to benchmark optimization solvers through the NEOS Server. Because NEOS users do not choose the machine that will be used by the solver, however, any timing information is unreliable. In addition, the information returned is not consistent across solvers, so that users are not able to compare the results obtained by each solver. In response to a growing interest in benchmarking [7, 8], we are developing a tool that will facilitate solver comparisons. This interest is also partially responsible for our development of performance profiles [4] for benchmarking.

We anticipate adding solvers in areas not currently represented. We also hope to add more global optimization solvers. Global optimization, in particular, is likely to raise scheduling and communication issues because solution times can be large.

Another possible direction for research is the addition of application-specific solvers. We get many inquiries, for example, about cutting stock problems. Much is known about solving these problems, but, aside from the simplest cases, they cannot be formulated in such a way that they can be solved by standard general-purpose codes. A solver designed specifically for cutting stock problems

would facilitate the modeling and solution of these problems.

## Acknowledgments

The NEOS Server is a collaborative project of the optimization community. Researchers associated with the Optimization Technology Center—in particular Jean-Pierre Goux and Jeff Linderoth—deserve special mention. A (we hope) complete list of collaborators can be found on the NEOS Server [9] site, but several efforts stand out:

Hans Mittelmann introduced the semi-infinite optimization and semidefinite and second-order cone programming area on the NEOS Server. He is also the solver administrator for more than a third of the solvers. David Gay provides constant support for the AMPL solvers.

GAMS Development Corporation provides solver support and is also the developer of a GAMS to AMPL translator. Companies provide popular commercial solvers for access through the NEOS Server (Dash Optimization's XPRESS, EKA Consulting's Mosek, and OptiRisk Systems' FortMP). Sun Microsystems has donated seven workstations to the Optimization Technology Center.

Of course, in many ways the most important collaborators are the developers of optimization software who allow their code to be run through NEOS. Without their research, none of this would be possible.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing, U.S. Department of Energy, under contract W–31–109–Eng–38, and by the National Science Foundation (Information Technology Research), grant CCR–0082807.

## References

[1] AMPL, see http://www.ampl.com.

[2] E.D. Dolan, *NEOS Server 4.0 administrative guide*, Technical Memorandum ANL/MCS–TM–250, Argonne National Laboratory, Argonne, Illinois, 2001.

[3] E.D. Dolan, R. Fourer, J.-P. Goux, and T.S. Munson, *Calling the NEOS Server with Kestrel*, 2002, in preparation.

[4] E.D. Dolan and J.J. Moré, *Bench-marking optimization software with performance profiles*, Math. Programming, 91 (2002), 201–213.

[5] E.D. Dolan and T.S. Munson, *The Kestrel interface to the NEOS Server*, Technical Memorandum ANL/MCS–TM–248, Argonne National Laboratory, Argonne, Illinois, 2001.

[6] GAMS, see http://www.gams.com/.

[7] H.D. Mittelmann, *Benchmarks for optimization software*, see http://plato.la.asu.edu/bench.html.

[8] H.D. Mittelmann, *Benchmarking interior point LP/QP solvers*, Optim. Methods Software, 12 (1999), 655–670.

[9] *NEOS Server for Optimization Problems*, see http://www-neos.mcs.anl.gov/.

*Elizabeth D. Dolan (dolan@mcs.anl.gov) and Robert Fourer (4er@iems.nwu.edu) are members of the Department of Industrial Engineering & Management Sciences, Northwestern University. Dolan is also a researcher in the Mathematics and Computer Science Division at Argonne National Laboratory, as are Jorge J. Moré (*more@mcs.anl.gov) and Todd S. Munson (munson@mcs.anl.gov).