Optimization as an Internet Resource

Robert Fourer *Department of Industrial Engineering and Management Sciences, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208-3119, U.S.A.*

Jean-Pierre Goux *Artelys S.A., 215, rue Jean-Jacques Rousseau, 92130 Issy-les-Moulineaux Cedex, France*

The rise of e-commerce promises particularly great benefits for the practice of large-scale optimization. The World Wide Web already offers information, advice, and remote access to software for solving optimization problems. A variety of client programs are helping to increase the scope and convenience of these tools. More sophisticated application service providers will further disseminate optimization modeling environments and solvers, making their power and variety readily available to a broader range of customers and applications.

July 21, 2000

Breakthroughs in computing have frequently led to advances in large-scale optimization, and we can expect the rise of e-commerce to continue this trend. Optimization is proving to be particularly well suited to Internet innovations, mainly for two reasons.

First, there is no one way to solve optimization problems. Hundreds of *solvers* have been developed to take advantage of the features of particular problem types. For many problem types, varied implementations of diverse methods compete on speed, reliability, cost, and convenience.

Second, new optimization applications typically involve building new models — as evidenced, for example, by the optimization model statements in appendices to many *Interfaces* papers. To support this model-building activity, specialized *modeling systems* have been developed for building, analyzing, and maintaining optimization models. These systems handle models independently of the choice of solver, and so can be hooked to a variety of solvers. Conversely, popular solvers work with a selection of modeling systems.

The analyst who wants to build an optimization application must consequently sort through quite a tangle of software. This is in contrast to the situation to such areas as statistics or simulation, where the software principally consists of integrated packages of modeling tools and model-analysis methods. And this is where Internet services enter the picture, as providers of guidance and access to the daunting variety of optimization software.

Preliminaries

By *optimization* we mean any desired minimization or maximization of some objective function of numerical decision variables, subject to constraints on the values of the variables. Optimization also encompasses the special cases in which there are no constraints or no objective, and admits the possibility of approximating the minimum or maximum as well as computing it exactly or to a tight tolerance. Optimization is synonymous with the more traditional but less descriptive *mathematical programming,* which gives rise to such terms as integer programming and nonlinear programming that we use occasionally for common problem types.

Optimization software includes solvers or codes for finding optimal (or at least very good) values for decision variables, as well as modeling systems that help to prepare problems for solvers and to deal with the solutions that solvers return. Solvers are specialized to many different mathematical problem types, and modeling systems are designed for certain broader categories of problems. The software that we consider is primarily *general-purpose,* however, in that it is not tailored to a particular application or even to an application class (such as crew scheduling or supply-chain management). To date, most of the interesting developments in optimization on the Internet have concerned general-purpose software, but because many application-specific packages are built on top of general-purpose codes or systems, the same developments have implications for application-specific optimization software as well.

We distinguish three categories of customers for general-purpose optimization software. *Modelers* work directly with solvers and modeling systems to build optimization models and to find ways of getting acceptable solutions. *Application developers* create software that runs solvers, but as part of larger packages that take care of such generic functions as managing data and presenting a graphical interface. *Users* run application packages that perform optimization at some stage. Modelers benefit most immediately from innovations that help people to choose and experiment with optimization software. Some application developers are also modelers, while others deal mainly with the inputs and outputs of optimization models set up by modelers. Users may not even realize that they are running solvers, though they are often aware of optimization goals such as minimizing costs or maximizing profits. Enhanced optimization services ultimately affect users and application developers as well as modelers, though in a less direct way.

Many of the services we describe were developed under the auspices of the Optimization Technology Center of Argonne National Laboratory and Northwestern University. Reconceiving optimization as an Internet resource is a major mission of the center's Network-Enabled Optimization System (NEOS) project.

Online Optimization Resources

The earliest use of the Internet in optimization was to provide software for downloading. Thus many noncommercial solvers have long been available via the `ftp` protocol, generally from sites maintained by their developers. The premier central site for downloading mathematical software, the Netlib repository [Dongarra and Grosse 1987], was begun in the early 1980s and includes a variety of solvers. Netlib is also the home of the `lp/data` collection of benchmark linear programs, compiled by David Gay [1985] from diverse sources. Analogous collections, such as MIPLIB, MINLPLIB, TSPLIB, SDPLIB, and OR-Library, have since been compiled for other problem types.

The advent of the World Wide Web has encouraged more extensive online resource collections, incorporating lists of hypertext links to downloads but also much other information and advice. Thus several websites now cover the field of optimization generally, and numerous others address specialized optimization topics.

The Decision Tree for Optimization Software, developed by Hans Mittelmann and Peter Spellucci, organizes non-commercial optimization software by problem type. This site also lists test problems, books, tutorials, modeling systems, automatic differentiation packages, and model analysis tools; a particularly impressive compilation of Benchmarks for Optimization Software includes dozens of tables, each comparing a selection of solvers on test problems of a particular type. The ZIB MATHPROG site also offers pointers to many classes of public-domain optimization codes and to related information.

The Optimization Technology Center's NEOS Guide [Czyzyk, Owen, and Wright 1997] incorporates three overviews:

— the Optimization Tree, a thumbnail sketch of the best-known classes and subclasses of optimization problems;

— the Optimization Software Guide, a categorized listing of codes and packages updated from the survey by Moré and Wright [1993]; and

— Frequently Asked Questions for linear and nonlinear programming, including links to noncommercial codes and to commercial codes that offer freely downloadable demo versions.

The NEOS Guide Case Studies [Czyzyk, Wisniewski, and Wright 1999] provide elementary descriptions and interactive demonstrations of the diet problem, portfolio optimization, quadratic assignment, stochastic programming, the minimal surface-area problem, and the cutting-stock problem.

The e-Optimization.community website aims to be a “web meeting place” for optimization users and developers. It features a “who’s who” of people in optimization as well as listings of optimization resources, applications, vendors, case studies, and news. Much of this information is stored in a database that can be searched on a variety of criteria and can be updated or extended automatically through web form submissions. Optimization Online offers a similar kind of service specialized to research reports and papers on optimization.

Harvey Greenberg’s Mathematical Programming Glossary defines hundreds of terms relating to optimization problems and methods. Many entries include examples and hyperlink cross-references.

Among the specialized sites are ones for global optimization, cutting and packing, semidefinite programming, and complementarity problems.

Optimization Servers

In the mid-1990s, developers of optimization software began to conceive of World Wide Web services that would allow prospective users to try their software without having to download or install it. The initial *optimization servers* tended to use e-mail or `ftp` to move problem files in one or both directions, with the associated web pages advertising and explaining the service. Designs soon evolved, however, to make use of web forms on pages that were integral part of servers’ operations.

As of mid-2000, at least 10 websites can be identified as providing optimization services. All are free, in that they charge nothing to those who submit requests; demand is kept under control by a variety of explicit or implicit nonmonetary limitations. OptiW, MILP, and the NEOS Guide Interactive Simplex Tool focus on small-scale demonstrations of standard methods, mainly for educational purposes. Six others (described below) have been created by developers of particular modeling languages or solvers to encourage familiarity with their products. A final example, the NEOS Server, provides a single mechanism for access to a variety of solvers and representations.

The solvers behind these servers can also be purchased to support more intensive or commercial use. Yet all have originated from academic or research organizations. Companies that sell optimization software have put up similar servers from time to time, but have been quicker to discontinue them when development or marketing priorities have changed.

Solver servers. Four servers are dedicated to particular solvers. They recognize algebraic expressions, such as `x^2 + 6.0*x = y - 2^k` or `IF x1 > 0 THEN g = 0.5*p*arctg(v)`, as input via web forms, and return results in web pages. Each solver handles nonlinearities that in some respects go beyond the traditional local numerical optimization of smooth (differentiable) nonlinear functions. Three are hosted in Russia and Finland, perhaps reflecting servers’ usefulness for publicizing research outside major centers of optimization software development.

BARON, developed by Nick Sahinidis’s optimization group in the Department of Chemical Engineering at the University of Illinois, combines interval analysis and a branch-and-bound framework to seek globally optimal solutions to nonlinear optimization problems in continuous and integer variables [Tawarmalani and Sahinidis 1999].

A dozen modules provide related solvers specialized to specific problem types. Input can be copied to a web form or supplied from a local file; output is returned on a subsequent web page. Submissions require a password, which can be obtained by writing to an address provided on the web input page.

HIRON, developed by the Russian VasBo (or Practical Optimization) Club, solves nonlinear problems by means of a hybrid strategy of local and global methods. Its server makes available a subset of its computational options, for unconstrained, possibly nondifferentiable optimization problems of two to five variables. Input is through a specialized web form with a large window for nonlinear expressions and small ones for variable bounds and algorithmic parameters.

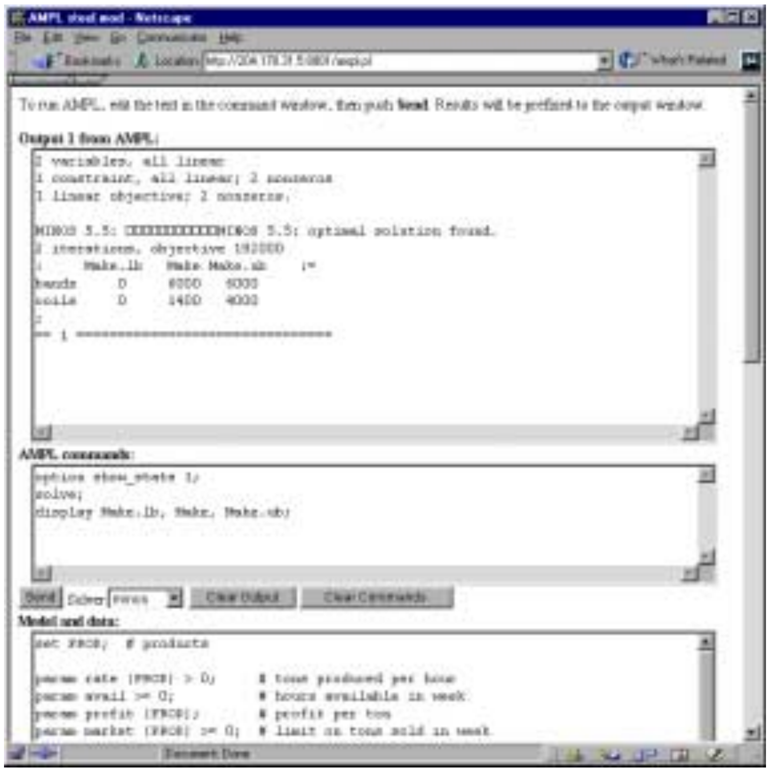
NIMBUS, developed by Kaisa Miettinen and Marko M. Mäkelä [2000] at the University of Jyväskylä, Finland, is designed for multiobjective optimization problems that may incorporate nondifferentiable functions and integer variables. The modeler interacts with the solver to establish priorities for different objectives and to decide between alternative solutions. The server accepts problem specifications through a highly structured series of web forms, one for each nonlinear objective, nonlinear constraint, variable bound, linear constraint coefficient, and so forth. Problem size is limited implicitly by the inconvenience of maintaining many variables, objectives, and constraints in this way, though users can register to save problems at the server site.

UniCalc, developed by a team led by Alexander Semenov at the Russian Research Institute of Artificial Intelligence [Babichev et al. 1993], applies interval arithmetic to the analysis of possibly nondifferentiable equations and inequalities, optionally with integer variables. The emphasis is on finding solutions to constraints, but there are limited facilities for optimization as well. The server accepts problems of up to 10 variables and 20 constraints, with some provision for arrays and indexing in the algebraic expressions.

Modeling language servers. Two servers are dedicated to particular modeling languages for optimization problems. Both are based on algebraic notation, like the languages used by the solver servers, but are able to express much more complex models and generate much larger problems, because they incorporate the concepts of sets and indexing in a more comprehensive way.

LPL, developed by Tony Hürlimann [1999], is a modeling language for linear and integer programming. It also accepts constraints connected by logical operators, such as `AND` or `NOT`, which it translates to linear constraints in terms of additional zero-one variables. Modelers can submit LPL formulations and data to the LPL server by typing or pasting into a web form or by specifying a local filename; the server applies an internal solver and returns results on a subsequent web page. Problems are limited to 100 variables and 100 constraints. (A PC version for larger problems, with hooks for other solvers, can be downloaded free of charge.)

AMPL, developed by Robert Fourer, David Gay and Brian Kernighan [1990], is a modeling language for linear and nonlinear optimization, with special features for integer, network, and complementarity problems. Users of the Try AMPL! server can request any example from the AMPL book [Fourer, Gay, and Kernighan 1993], or may provide their own model, data, and commands by typing or pasting into a web form or by specifying a local filename. To encourage experimentation, the server interface employs web forms (Figure 1) that display results of previous submissions and that allow the input to the most recent submission to be changed and resubmitted. A pull-down menu offers a



To run AMPL, edit the text in the command window, then push **Send**. Results will be prefixed to the output window.

Output 1 from AMPL:

```
2 variables, all linear
1 constraint, all linear; 2 nonzeros
1 linear objective; 2 nonzeros.

MINOS 5.5: □□□□□□□□□□□□MINOS 5.5: optimal solution found.
2 iterations, objective 192000
:      Make.lb   Make Make.ub    :=
bands     0      6000   6000
coils     0      1400   4000
;
== 1 ==================================
```

AMPL commands:

```
option show_stats 1;
solve;
display Make.lb, Make, Make.ub;
```

Model and data:

```
set PROD;  # products

param rate {PROD} > 0;      # tons produced per hour
param avail >= 0;           # hours available in week
param profit {PROD};        # profit per ton
param market {PROD} >= 0;   # limit on tons sold in week
```

AMPL steel.mod - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://204.178.31.5:0801/ampl.pl
What's Related
Send
Solver
MINOS
Clear Output
Clear Commands
Document: Done

Figure 1: The “Try AMPL!” server’s result screen includes an output window (at top) and a solver menu and other controls. The contents of the commands and model/data windows may be modified and resubmitted.

choice of eight solvers. Problems are limited to 300 variables and 300 constraints plus objectives, and one minute of computer time. (Versions without the time limitation are bundled with the AMPL book and can be downloaded free of charge.)

The NEOS Server. The NEOS Server for Optimization [Czyzyk, Mesnier, and Moré 1997; Gropp and Moré 1997] is the most ambitious realization to date of the optimization server idea. A cooperative effort of over 40 designers, developers, collaborators, and administrators at the Optimization Technology Center, it provides access to over two dozen solvers of many kinds (Figure 2). Modelers can submit problems to be solved

— by entering local filenames into web forms;

— by sending an e-mail message in a specified format;

— by using a TCP/IP socket-based submission tool, a specialized interface for problem submission available in Java and in Unix tcl/tk versions.

The server returns an output listing by whichever mechanism was used for the input.

The NEOS Server project has taken several steps to encourage a steady growth in the number of connected solvers. The project solicits solvers of all kinds, including ones that are proprietary to varying degrees (but whose owners have been willing to permit their use at no charge through the server). Although the server’s location is fixed, it is able to connect to solvers anywhere on the Internet, so that available computing resources can grow along with the number of solvers. Finally, the server offers standard and documented procedures for adding solvers, permitting new solvers to be

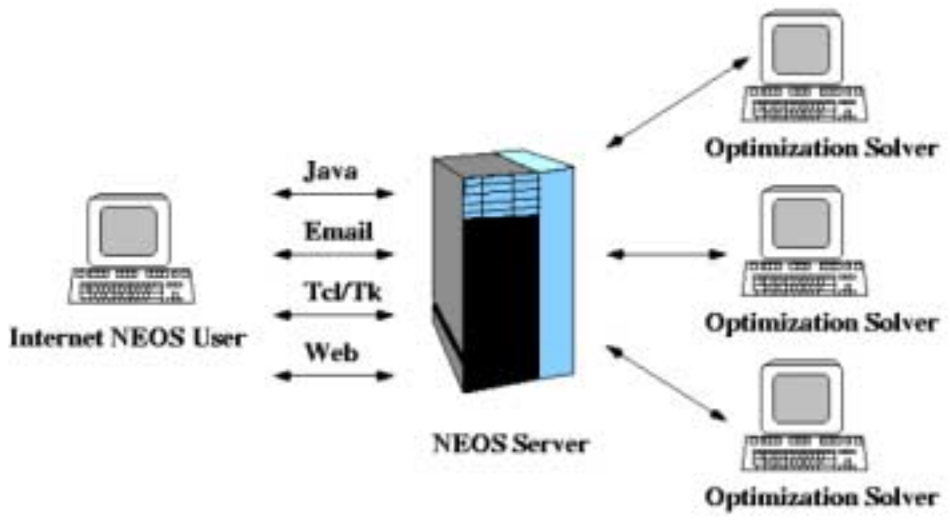


Figure 2: The NEOS Server accepts optimization requests in several ways and distributes them to requested solvers that may reside anywhere on the Internet.

registered automatically through the same mechanism that is used for ordinary optimization requests.

The large number of registered solvers precludes standardization on any one input format. The server design instead allows any text file to be passed through to a solver, so that the formats currently recognized have been determined largely by what the solvers are able to accept. These formats are of three main types:

— Low-level formats explicitly describe every constraint and objective. They include MPS for linear programs, SIF for nonlinear programs, SMPS for stochastic programs, and sparse SDPA for semidefinite programs. To accommodate the large file sizes these formats often require, the server recognizes several common compression schemes.

— C or Fortran programs represent constraints and objectives by computing function values at points that solvers specify. For solvers that require derivatives, the server can run these programs through the automatic differentiation tools ADIFOR and ADOL-C to add code that computes exact derivatives efficiently.

— High-level algebraic formulations describe optimization problems in concise, symbolic formats, using modeling languages such as AMPL [Fourer, Gay, and Kernighan 1993], GAMS [Brooke, Kendrick, and Meeraus 1992], and MP-MODEL. An accompanying data file specifies the model instance to be solved. The server runs the language processing software that converts models and data to low-level forms that solvers require.

All of these cases can be seen to involve certain preprocessing steps that the server must undertake before routing submissions to solvers and results back to users.

In 1999–2000, the NEOS Server typically received several hundred submissions each week, with peak loads over a thousand (Figure 3). In any system of this complexity, varied problems arise: machines go down, software crashes or terminates in unexpected

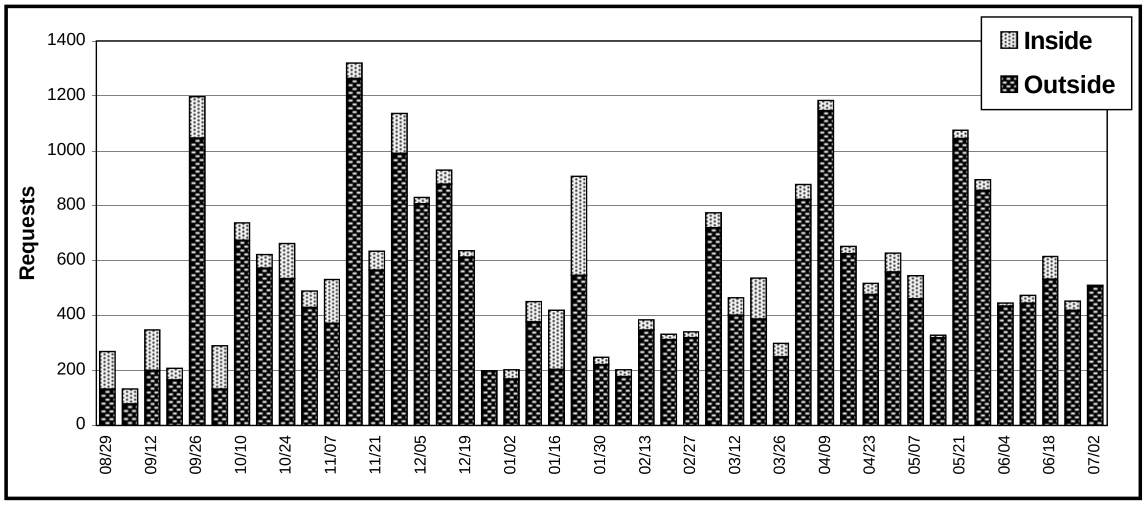


Figure 3: Bars show weekly optimization requests handled by the NEOS Server from August 1999 to July 2000. *Inside* refers to requests from domains associated with the NEOS project, and *Outside* to all other requests.

ways, and solvers run much longer than expected. Questions from users are automatically logged and distributed to a mailing list of Optimization Technology Center members and affiliates, who work out informally who answers which queries. The server's developers have used this experience to implement refinements that improve reliability.

The NEOS Server abandons any submission not finished after some specified period, currently one week. It also keeps a separate list of tighter restrictions for each registered solver. Solver-specific limits may be placed on execution time and on hourly, daily, or monthly submissions — from all sources, from any one domain or user, or from addresses matching a given regular expression. Some solvers also impose their own restrictions, such as limits on the numbers of variables of certain kinds.

Submission limits are one of several factors that have combined to keep demand for the NEOS server to the levels seen in Figure 3:

— Successful operation of solvers often requires expertise beyond the information available at the NEOS website.

— Absence of a service guarantee limits most use to prototyping, experimentation, and educational projects.

— Format and location of the output listings are inconvenient for high-volume or large-scale applications.

These limitations are addressed by several aspects of the @NEOS project described later in this article.

Optimization Clients

In any use of a remote server, some local *client* program is invoked to manage the communications. For the optimization servers described so far, the main interest is in what happens at the server side, while the client is generally a browser that sends problems and receives results via ordinary web pages. Client-server interactions are mediated by the most basic of web components: browser-based interpreters for Hypertext Markup Language (HTML) pages, servers for Hypertext Transfer Protocol (HTTP) requests and responses, and server-side connections to solvers via Common Gateway Interface (CGI) scripts.

These universally recognized standards do not offer sufficient speed and interactivity for many optimization applications, however, as Bhargava and Krishnan [1998] make clear in their survey of Web technologies for operations research and management science. More convenient and powerful uses of the Internet in optimization take advantage of newer technology to better balance the work between client and server while maintaining or improving the quality of client-server communication.

Alternative client-server arrangements. Two spin-offs of the NEOS Server project, iNEOS and AMPL Remote Access, have experimented with configurations that permit a greater amount of the work to be done on the client side, with the effort partitioned in such a way that speed of communication is not essential. Prototypes employed the experimental Nexus communication library [Foster, Kesselman, and Tuecke 1994]. More recent versions conduct communications via CORBA, which offers the greater stability and portability of an established standard, together with the advantages of an object-oriented design. The client program gains access to CORBA services through a unified C/C++ application programming interface.

The iNEOS project [Good et al. 2000] is motivated by diverse situations in which an optimization problem's objective and constraint functions cannot be sent to a remote solver. The steps for computing these functions may not be expressible in a form that the server recognizes — such as an AMPL model or a single C or Fortran program — or may require greater computational resources than the server can provide. Each function evaluation may involve a simulation run, for example. In other cases, the modeler may not want to risk revealing propriety information to a remote site.

Fortunately, many solvers for nonlinear problems require only that the objective and constraint function values — and perhaps gradients — be computable at points that they generate. These solvers need no details of the function computations. The iNEOS client takes advantage of this characteristic by arranging for a remote solver to return

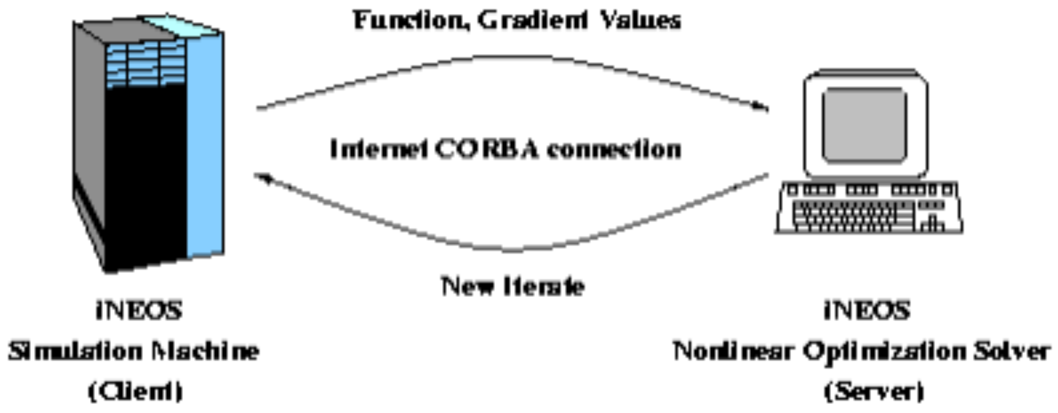

Function, Gradient Values
Internet CORBA connection
New Iterate
iNEOS
Simulation Machine
(Client)
iNEOS
Nonlinear Optimization Solver
(Server)

Figure 4: The iNEOS client enables a remote solver to return function evaluation requests to a local computer.

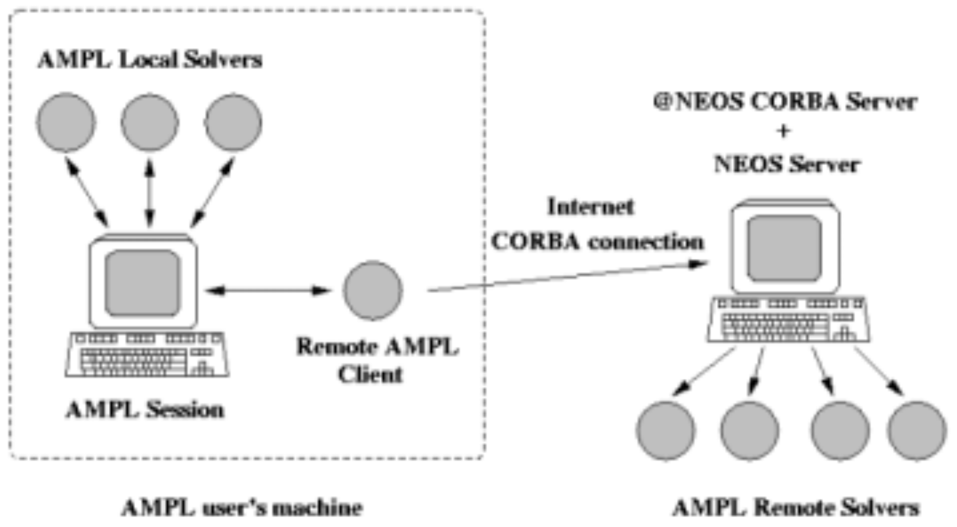


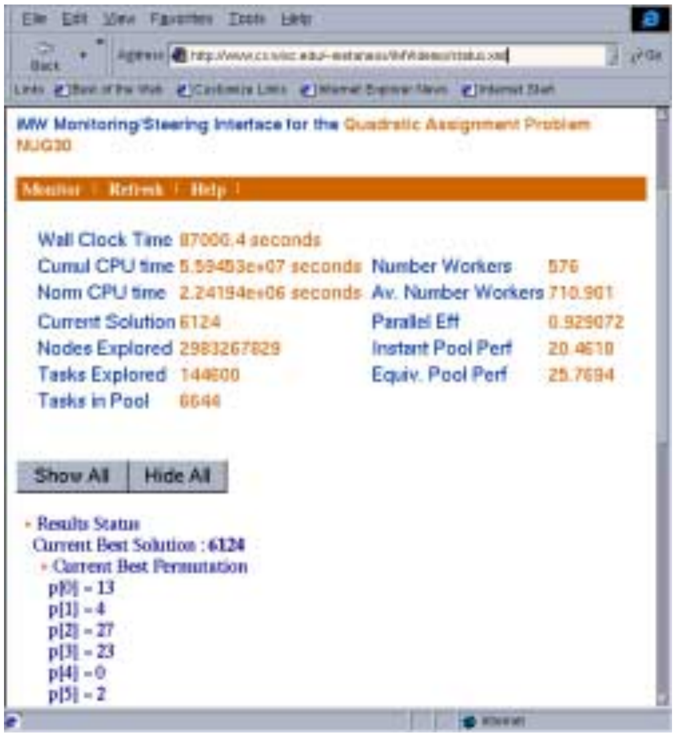
Figure 5: The AMPL Remote client lets a local AMPL session use remote solvers via the NEOS server.

function evaluation requests to the local computer (Figure 4). The extra communication takes only a small fraction of the total time required, particularly when the function evaluations are time-consuming. This approach has been applied experimentally to problems of parameter estimation in materials science [Carcione et al. 1999] and fluid dynamics.

AMPL Remote Access addresses another situation in which the server cannot do all of the desired computing. Users of advanced optimization modeling languages have access to rich interactive environments for managing projects, analyzing models, displaying results, and stepping through iterative schemes that require a series of optimizations. Much of this is lost when models and data are sent as jobs to be queued for translation and solution at a server. The NEOS Server, in particular, returns only a text-format listing that is not well suited to further manipulation and analysis.

The AMPL Remote Access client remedies this situation by allowing local AMPL interactive sessions to directly invoke remote NEOS solvers (Figure 5). From the perspective of the AMPL system and its users, the Remote Access client looks like any other solver; its invocation requires only a minor change to the usual solver option settings. The translation of an AMPL model and data to an explicit problem file is done locally, however, rather than on the server as in the current NEOS setup. Behind the scenes, the client takes care of sending the problem file to the requested solver, retrieving the results, and leaving a result file where AMPL expects to find it.

Metacomputing clients. Certain computational methods for very large problems lend themselves to decomposition into pieces that that are sufficiently independent to be run on separate workstations, coordinated via standard network connections. Creators and users of such an arrangement, a so-called metacomputer [Smarr and Catlet 1992] or computational grid [Foster and Kesselman 1999], can be assisted by specially designed metacomputing software that automatically addresses many issues of fault tolerance, task scheduling, and interprocess communication. Still, the operational details are sufficiently complex to make client front ends attractive. Systems such as


File Edit View Favorites Tools Help
Back
Address
Go
Links
Monitor | Refresh | Help |
Show All
Hide All

iMW Monitoring/Steering Interface for the Quadratic Assignment Problem NUG30

```
Wall Clock Time  87000.4 seconds
Cumul CPU time   5.59453e+07 seconds  Number Workers      576
Norm CPU time    2.24194e+06 seconds  Av. Number Workers  710.901
Current Solution 6124                 Parallel Eff        0.929072
Nodes Explored   2983267829           Instant Pool Perf   20.4610
Tasks Explored   144600               Equiv. Pool Perf    25.7694
Tasks in Pool    6644
```

- Results Status
  Current Best Solution : **6124**
  - Current Best Permutation
    p[0] = 13
    p[1] = 4
    p[2] = 27
    p[3] = 23
    p[4] = 0
    p[5] = 2

Figure 6: This web browser display, managed by iMW, reports the progress of a metacomputer solver for a quadratic assignment problem. The metacomputer has been running for about 24 hours to this point, and is currently coordinating computations on 576 workstations.

NetSolve [Arnold et al. 2000] and Ninf [Nakada, Sato, and Sekiguchi 1999], for example, offer client-server interfaces for scientific computing tools on the computational grid provided by the Internet.

An optimization solver running on a metacomputer could be made available through the NEOS server like any other solver running on a remote machine [Ferris, Mesnier, and Moré 2000]. This mode of operation is problematical, however, because it gives users no way to see what is happening during the often lengthy, complex runs. The development of clients better suited to this purpose has been a major goal of the MetaNEOS project, which has pursued approaches relying on web browsers and on optimization modeling systems.

For the underlying management of the metacomputer, the metaNEOS approaches rely on Condor [Litzkow, Livny, and Mutka 1988; Epema et al. 1996], a tool for managing large heterogeneous collections of workstations as computing resources. Condor allocates jobs to unused machines within a designated *pool*, and saves the status of a job at periodic *checkpoints* so that it may restart from the most recent checkpoint if a machine becomes unavailable before the job's completion. For metaNEOS applications, each job is typically an optimization subproblem of some kind.

The iMW package [Good and Goux 1999] manages metacomputer runs through a web-based tool for submitting, monitoring, and steering solvers that run on metacomputers. A CORBA connection, in this case between the web server and the metacomputer controller, periodically refreshes a browser display (Figure 6) that reports such progress information as the amount of time and work expended, the number of Condor pool machines currently being used, and the best solution found so far. On the basis of this information, the modeler can also convey instructions through the browser interface to the solver during its run.

To use iMW for a particular optimization method applied to a particular problem, one first implements the method for Condor using the MW development framework [Goux et al. forthcoming]. MW, also developed in the metaNEOS project, is a C++ class library that facilitates implementation of algorithms on a Condor metacomputer; it must be tailored to a particular application by reimplementation of three base classes, specifying the activities of the master process, the inputs and outputs of each worker process, and the activities of each worker. The iMW tool, implemented as a Java servlet, can be viewed as a middle tier between users and MW-based solvers. It handles communication in both directions throughout a run and can be customized in various ways [Good and Goux 1999]. MW/iMW solvers are under development for problems in integer linear and nonlinear programming, stochastic optimization, and quadratic assignment.

The Condor/modeling language interface [Ferris and Munson 2000] addresses implementation of an iterative optimization scheme on a metacomputer at a much higher conceptual level than iMW, using the commands of an optimization modeling language. Once an ordinary command script for some method has been implemented, conversion to the Condor interface can be as simple as replacing a `solve` command by an invocation of a `condor_spawn` command file, and adding an appropriately placed corresponding `condor_retrieve`. Specifics have been worked out for the AMPL [Fourer, Gay, and Kernighan 1993] and GAMS [Brooke, Kendrick, and Meeraus 1992] languages, and have been applied to a problem in feature selection that requires many independent integer programs to be solved simultaneously. Because this arrangement is much easier to set up than an MW/iMW-based solver, it makes metacomputing much more accessible to someone who has a straightforward use for optimizations carried out in parallel. It does not currently support the sophisticated communications options of iMW, however, or MW's facilities for programming master-worker schemes in detail.

Applets. The servers and clients described so far return their results as files or displays of text. Yet graphical displays can be essential to the effective use of optimization in many applications. For optimization on networks, in particular, displays of node and arc structures can be tailored to aid in comprehension and analysis of diverse problems, algorithms, and results.

When the client is a web browser, the limitations of current designs derive from their use of the `html` format for defining web pages. One widely supported way of circumventing these limitations is by writing an application in the Java programming language and compiling it into a machine-independent code file known as a web *applet*. An applet can be embedded in a web page much like an image; but when the page is viewed, the applet code runs on the browser's built-in Java Virtual Machine as a program that can create its own windows and produce graphical displays of considerable generality. This provides a way to make a graphically sophisticated application available via the Internet without requiring its installation on the local machine.

The Java applet framework has inspired web-based graphical demonstrations of diverse optimization models and methods:

— TSPfast and TSPx illustrate elementary methods for the "traveling salesman" problem of finding short tours through given locations.

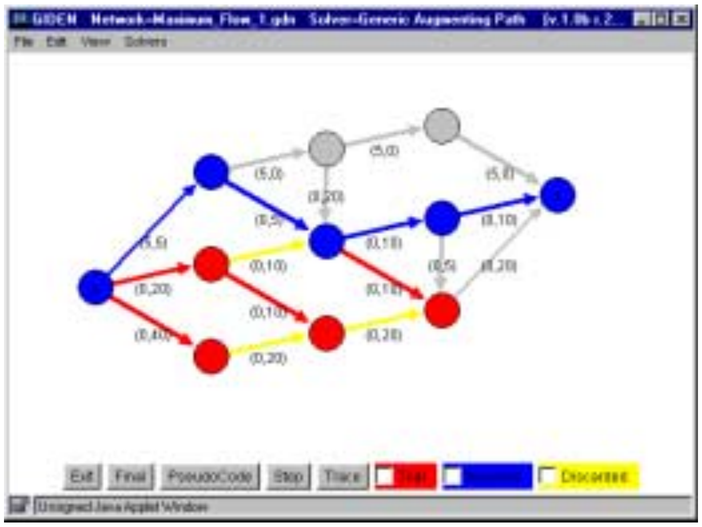


Figure 7: This GIDEN applet window depicts an intermediate state in the solution of a maximum flow problem by an augmenting path method.

- — The NEOS QAP Case Study supports experiments with quadratic assignment problems applied to facility location.
- — The GA Playground offers a toolkit for experimenting with local-search heuristics known as genetic algorithms, in applications to a variety of standard optimization problems.
- — RIOT, a Remote Interactive Optimization Testbed, collects over a dozen applet demonstrations of optimization methods for scheduling, design, routing, network flow, portfolio selection, numerous graph problems, and other applications.

Visualization and Optimization [C.V. Jones 1998] also describes and links to a number of applets, along with examples of several alternative technologies.

GIDEN, developed by Jonathan Owen, Collette Coullard, and David Dilworth [1997], is a more ambitious Java-based optimization project. It provides an interactive software environment for the visualization of network optimization problems, solutions, and algorithms. Features include a graphical interface for building and modifying networks, an expandable toolkit of animated solvers, and a library of network-related data structures. The current GIDEN toolkit includes a variety of solvers for each of several classic network optimization problems, including minimum spanning tree, maximum flow, shortest path, and minimum-cost flow. New problems and solvers can be added in a modular way.

Version 1 of GIDEN can be run as an applet that creates its own network modeling window (Figure 7). Network diagrams and associated data are depicted in the window by selecting built-in examples or by building a new example through a combination of `Edit` menu entries and mouse clicks. Solvers, chosen from the `Solvers` menu, may be run straight through or stepped along in a variety of ways, with colors depicting the current state of the algorithm. An optional auxiliary window steps through corresponding pseudocode for the algorithm.

Version 2 of GIDEN, which provides a much richer set of network creation and display facilities, is currently available only as a stand-alone Java program, requiring a computer on which a Java Virtual Machine has been installed as an application independent of any browser. The reasons for this difference point up some of the challenges of the applet approach. Java programs past the demo stage can take considerably more time to load and to run using a browser than under a dedicated Java Virtual Machine. Moreover, although in principle an applet runs in the same way through a browser on any computer, larger and more complex Java programs are more likely to require fine tuning to ensure compatibility, particularly if they do a lot of graphics.

An equally serious and perhaps more surprising difficulty is that, to ensure security, browsers do not normally allow applets to write local files. This rules out any serious use of applets to create new network problems, unless the developer is willing to set up a central site to register users and store their files. An alternative approach is to use a *signed* applet, which requests specified permissions, such as to read and write local files, by presenting a *certificate* attesting to its authenticity. Permissions are granted only if the local browser has a matching certificate in its database or if the user grants an exception for the current session. Such an arrangement may strike an acceptable balance between security and ease of use.

Whereas the GIDEN applet encapsulates both the graphical interface and the solvers, some of the RIOT applets handle only the graphical interface, while sending algorithmic work to remote solvers. For attacking large or hard network optimization problems by use of computationally intensive solvers, this client-server approach may be advantageous. Network optimization systems commonly use the same data structures for both the solver and the display routines, however, making a local/remote division problematical. Features that animate the actions of the solver, for instructional purposes or to report progress, also tend to require a close tie between the solver and the graphical display routines. Finally, solvers for many small or easy network problems are fast enough that the delay involved in communicating with a remote solver would be significant.

If a client-server arrangement is desirable but the applet framework is too restrictive, then there remains the alternative of writing independent client programs in Java, C++, or any other programming language that affords an adequate graphical interface. Microsoft's ActiveX Controls can integrate client program downloading with the browser interface, or ordinary downloading via `ftp` may suffice. Distribution then becomes more of an effort, however, as the client program must be separately compiled for each platform on which it is to run.

Application Service Providers

Do optimization clients and servers have commercial potential? We have seen a variety of configurations that have been implemented successfully, in the sense that they have been publicly available and have operated as intended over a period of time. None charge for their services, however. A number of thorny issues remain to be resolved before optimization on the Internet can grow beyond the prototype stage into a viable form of e-commerce.

One promising paradigm for commercialization of optimization servers is the concept of an *application service provider* that rents the use of software via the Internet rather than selling software to be installed on customers' computers. An ASP typically offers an integrated package that includes use of specified software on a remote server, upgrades and maintenance, user support, and possibly access to advanced computing resources. An organization that buys an ASP's services hopes to spend less than it would by adding equivalent internal staff and equipment. Potential savings are particularly attractive for software that requires specialized expertise and computing resources, as is the case with much of the software used in operations research and management science applications.

Growing interest in ASPs has helped to motivate a variety of software for running existing applications on remote servers. Products such as Microsoft's Windows Terminal Server, Citrix's MetaFrame and New Moon's LiftOff permit centralized servers to run complex software systems, including full graphical user interfaces, that function on each local client as if the software had been installed locally. The details of the client-server link vary from one package to the next, both in the degree to which they combine proprietary communications protocols with standard Web technology, and in the balance between work done on the client and on the server.

An ASP's business can thus be built around any software product, using widely available tools. Developers of specialized optimization applications have in fact taken an interest in creating ASPs in this way; publicly available demonstration versions have been made available by, for example, LogicTools and MultiSimplex.

The prospects for general-purpose optimization ASPs are harder to assess, however, owing to the special nature of the optimization software marketplace. In simulation, statistics, vehicle routing, logistics, supply-chain management, and other application areas of operations research and management science, popular general-purpose software tends to consist of integrated packages of modeling and solving tools; each package has one developer that is the obvious source of maintenance and support for ASP services. The general-purpose optimization software market is much more decentralized, however, offering a broad menu of solvers for varied purposes and an independent choice of a modeling languages and environments. Modelers and application developers often combine optimization products from different sources, requiring a flexible sort of expertise that is not easily centralized.

Thus while any of the general-purpose optimization servers we have described could be considered a prototypical ASP, the process of "scaling up" these prototypes to full-fledged optimization e-commerce sites poses great challenges to the ASP concept. Although current developments in optimization client design provide a good start in addressing these challenges, the full benefits of ASPs to optimization will only be achieved through significant enhancements to every part of the client-server system:

- Servers will require sophisticated allocation strategies that can cope with exceptionally variable solver resource requirements, together with coordination mechanisms that can balance the interests of model builders and solver providers.
- Solvers will need more standardized interface libraries that simplify the work of connecting to the server and that support frequent communication with the server and clients.
- Modelers and application developers will want better guidance in choosing among solvers and in selecting solver features and options.
- Users will benefit from availability of a programming interface that makes the server callable from within specialized applications.

Work is underway to integrate the current NEOS Server into a more highly automated system, called @NEOS, that will provide a test-bed for improvements in many of these areas.

Servers. The greatest obstacle to effective resource allocation for optimization requests is the potential for exponential growth. Even the best integer programming solvers, for instance, sometimes require a running time to optimality that would exceed the lifetime of any component, human or machine, of the system. Even the most experienced modelers elicit this behavior in experimenting with new formulation and solution strategies.

An optimization server thus needs some way to protect itself (and its customers) from requests that can soak up all available resources. A time limit or a charge proportional to resource use would seem easy to implement, but could readily become the kind of nuisance factor that only discourages the use of central resources. Servers will instead need to adopt more flexible strategies that can take advantage of prior experience with different problems and solvers.

A first step would be to build a database of solver performance that could be automatically updated as optimization requests are carried out. A new *adaptive scheduler* could then employ information from the database, together with specific customer preferences, in making initial allocations of computing resources to requests. For long-running jobs, such a scheduler could also monitor performance and take simple actions, such as increasing or decreasing a job's priority, moving a job to a faster or slower machine, suspending a job while querying its owner for instructions, and terminating a job.

The next step would be to introduce a broader conception of the optimization server, not simply as a tool to carry out whatever work is requested, but as a *broker* between customers and solvers. Solver providers would correspondingly assume the role of independent *agents* competing for customers' business. The central issue in the implementation of an ASP would then become the design of a *coordination mechanism* for acting on service requests.

Since each party to an optimization request has its own interests, it is no simple matter to devise a mechanism that keeps all of them satisfied. The broker could assign requests to agents based on some overall model of solver performance and resource availability, for example, or could conduct a kind of auction in which agents bid for customers' business. Requests could be scheduled after they were assigned to agents, or scheduling could be made an integral part of the assignment process. Pricing could involve agent "rents" as well as charges determined by various measures of resource use.

All of these e-commerce aspects of an optimization server have been considered to some extent in studies of earlier systems, including

— DecisionNet [Bhargava, Krishnan, and Mueller 1997; Bhargava, Chowdhary, and Krishnan forthcoming], a prototype broker for "decision technologies" such as optimization; and

— IBIZA [Arora et al. forthcoming], a computational workbench for creating and simulating electronic markets in information products.

It remains to put these ideas to the test in contexts specific to optimization. Service logs from the NEOS Server and its @NEOS extensions should help to provide data for this purpose.

Solvers. For every solver added to an existing optimization server, an appropriate front end or interface routine must be written. Although this work is straightforward in principle, it can be time-consuming, particularly as there is seldom any one person familiar with the requirements of both the server and the solver to be added.

One particularly challenging aspect of front end design is to manage interaction with the solver while it is running. Servers need to communicate with running solvers to support adaptive scheduling. Clients want to monitor or control lengthy solver runs, particularly within complex schemes (like metacomputing applications) that require multiple solves and processors.

To encourage solver developers to undertake this interface work and to ensure reliable results, an established optimization server will need to provide an interface library that encapsulates most of the details. Such a library will have a purpose similar to, for example, the AMPL interface library [Gay 1997], but with an object-oriented design and support for common interface standards such as CORBA. One of the planned @NEOS extensions to the NEOS Server is a library of this sort.

Modelers and application developers. The NEOS Server already offers over 25 different solvers, yet they are only a sample of those available. Each solver offers as many as several dozen distinct algorithmic options. For the person who must develop and debug an optimization model, the proliferation of solvers can quickly become too much of a good thing.

As an example, consider that the NEOS Solvers list already offers seven alternatives for general nonlinear optimization: CONOPT, DONLP2, FILTER, LANCELOT, LOQO, MINOS, and SNOPT. The server's website provides for each of these an introductory page and a hyperlink to a list of option names and one-line descriptions. A link is also provided to any guide or reference material that the solver's developers have made available, but using this material requires a considerable investment of time as well as a knowledge of nonlinear programming theory and algorithms. Even experienced modelers often resort to trying solvers one after another with their default options, sticking with the first one that seems to return useful results.

The NEOS Server might deal with this situation by linking more closely to information available in the NEOS Guide and by incorporating better on-line option descriptions. The @NEOS project envisions a more automated solution, however, provided by an advanced *solver query engine*. Such a system would most likely be built as a kind of database query manager, taking as input a list of problem characteristics and producing as output a list of suggested solvers.

As an example, the query engine's input list of characteristics in the nonlinear case could include,

— for the objective, whether it is to be minimized or maximized, whether it is linear, quadratic, or more general in form, how many separable terms it contains, and whether it is convex, concave, or neither;

— for the constraints, the numbers of equalities and inequalities having analogous properties, as well as a measure of the sparsity of the derivative (Jacobian) matrix.

Information in the database would indicate the degree to which different solvers would require or prefer the presence or absence of different characteristics. The mechanisms for storing and applying this information are not as yet well understood, though there has been some work on the problem in the context of other mathematical software [Lucks and Gladwell 1992]. The output would include a ranked list of solvers, and preferably also a list of suggested option combinations for each solver.

A solver query engine could communicate directly with modelers, but its usefulness would then depend on the willingness and ability of people to give correct lists of characteristics for the problems they want to solve. If problem characteristics could instead be automatically extracted from modelers' submissions, the query engine could operate much more reliably. This suggests the desirability of an *optimization problem analyzer* that would accept any of several problem formats as input and would produce as output the lists of problem characteristics required by the query engine. For nonlinear optimization there is already a utility along these lines, MProbe [Chinneck forthcoming], which takes the output of the AMPL modeling language translator as its input, and produces a variety of tables and graphs concerning problem size, convexity, and other useful properties. The same approach might well be used to produce a list of characteristics relevant to solver choice.

Given a problem analyzer whose output feeds into a solver query engine, an optimization ASP could generate solver suggestions tailored to each problem submitted, taking into account which solvers were currently available. The server could then proceed by sending the problem to the solver ranked #1 by the query engine, automating the solver choice entirely. Making the choice such a black box might encourage overconfidence in the system's abilities, however; it would also be desirable for the query engine to optionally return a ranked list of solvers, together perhaps with some indication as to why solvers ranked as they did. This could help a customer to track down why, say, a model thought to be linear was recommended only for nonlinear solvers.

As a further enhancement, the query engine might extract information from the database of the server's adaptive scheduler via some automatic and ongoing process. This would allow it to rank solvers based on actual as well as expected performance.

Users. An optimization model is seldom an end in itself. Most models are intended to be built into more general application software, so as to provide a convenient interface for some broader function. Applications of this kind have become widespread throughout the manufacturing and service industries, with particularly fast growth in such areas as production planning, transportation, and logistics. Developers are also now starting to build optimization into web-based applications, to provide fast configuration decisions for complex products such as telecommunications equipment and secured loans. Users of these products are seldom required to know much about optimization, and indeed are often unaware that a minimization or maximization is being carried out to produce some of the information displayed.

Application software of this kind could make good use of an optimization server. In this context the application would act as a client, but would access optimization services not by use of web pages and buttons, but through equivalent procedure calls. The conversion of the NEOS Server to a callable "automatic" NEOS has been the first major goal of the @NEOS project. It has already been tested to some extent, though in a general-purpose setting, as a component of the AMPL Remote Access modeling environment.

As the optimization server is enhanced in the various ways previously described, the equivalent procedure calls will need to be extended accordingly. Thus a carefully designed application program interface for Internet optimization services would do much to encourage development of client applications. An API of this kind would provide a standard and general way for applications to invoke remote optimization services, and could thus serve as a catalyst to greatly expand the benefits of optimization on the Internet into a variety of powerful and flexible environments.

Further Directions

In most of the initial development of optimization as an Internet resource, the basic unit of request has been the solution of one problem using one solver. Many of the same ideas could be applied to allow two or more remote solvers to be used in combination to attack especially hard problems. Dotti et al. [2000] describe one possibility, an architecture that would permit researchers at several institutions to experiment with metaheuristics for combinatorial optimization problems by combining diverse heuristic methods that are shared as network resources.

We can also expect that Internet resources of the kinds we have described for optimization will be concurrently developed for other modeling techniques. Future network services should thus be able to support iterative schemes that involve combinations of solvers for optimization, statistical estimation, and numerical computation generally. The same services could also provide access to tools for problem analysis and for database access. The MMM method management system [Günther et al. 1997] is a prototype for the integration of diverse computational software into such an arrangement; it provides a common framework to facilitate the use of outputs from one tool as inputs to other, possibly quite different tools.

The idea of optimization servers as Internet resources is also likely to be expanded to facilitate remote access to optimization models and data. Developers of models for particular applications might post them to a kind of model server, for example; a user of this server would send the inputs required by a chosen model and would receive the model's specified outputs. In this way optimization at a higher conceptual level could also come to be viewed as an Internet resource.

Web Links

ActiveX Controls `www.microsoft.com/com/tech/activex.asp`

adding solvers (NEOS Server) `www-neos.mcs.anl.gov/neos/solvers/ADMIN:ADDSOLVER/`

ADIFOR `www.mcs.anl.gov/adifor/`

ADOL-C `www.math.tu-dresden.de/wir/project/adolc/`

AMPL `www.ampl.com/ampl/`

AMPL Remote Access `www.ampl.com/ampl/NEOS/`

applet `java.sun.com/applets/`

application service provider `www.aspindustry.org/`

BARON `archimedes.scs.uiuc.edu/cgi/run.pl`

Benchmarks for Optimization Software `plato.la.asu.edu/topics/benchm.html`

complementarity problems `www.cs.wisc.edu/cpnet/`

cooperative effort (NEOS Server) `www-neos.mcs.anl.gov/neos/collab.html`

CORBA `www.omg.org/`

cutting and packing `prodlog.wiwi.uni-halle.de/sicup/`

DecisionNet `www.heinz.cmu.edu/project/dnet/`

Decision Tree for Optimization Software `plato.la.asu.edu/guide.html`

e-Optimization.Community `www.e-optimization.com/`

Frequently Asked Questions (NEOS Guide) `www.mcs.anl.gov/otc/Guide/faq/`

GA Playground `arieldolan.com/ofiles/ga/gaa/gaa.html`

GAMS `www.gams.com/`

GIDEN `giden.iems.nwu.edu/`

global optimization `solon.cma.univie.ac.at/~neum/glopt.html`

HIRON `vasbo.comtel.ru/hiron.htm`

iMW `www.mcs.anl.gov/metaneos/softtools/imw.html`

iNEOS `www.mcs.anl.gov/metaneos/softtools/ineos.html`

Interactive Simplex Tool `www.mcs.anl.gov/otc/Guide/CaseStudies/simplex/`

LiftOff `www.newmoon.com/`

LogicTools `www.logic-tools.com/`

lp/data `netlib.bell-labs.com/netlib/lp/data/`

LPL `www2-iiuf.unifr.ch/tcs/lpl/`

LPL server `www2-iiuf.unifr.ch/tcs/lpl/lplwww/lplfile.htm`

Mathematical Programming Glossary
`www.cudenver.edu/~hgreenbe/glossary/glossary.html`

MetaFrame `www.citrix.com/`

MetaNEOS `www.mcs.anl.gov/metaneos/`

MILP `pinnacle.edrc.cmu.edu:8080/milp.shtml`

MINLPLIB `www.mcs.dundee.ac.uk/~sleyffer/MINLP_TP/`

MIPLIB `softlib.rice.edu/softlib/catalog/miplib.html`

MMM `http://meta-mmm.wiwi.hu-berlin.de`

MP-MODEL `www-neos.mcs.anl.gov/neos/solvers/IP:MPMOD-XPRESS/`

MProbe `www.sce.carleton.ca/faculty/chinneck/mprobe.html`

MPS (format)
`www.mcs.anl.gov/otc/Guide/OptWeb/continuous/constrained/linearprog/mps.html`

MultiSimplex `www.multisimplex.com/`

MW `www.cs.wisc.edu/condor/mw/`

NEOS Guide `www.mcs.anl.gov/otc/Guide/`

NEOS Guide Case Studies `www.mcs.anl.gov/otc/Guide/CaseStudies/`

NEOS Server for Optimization `www-neos.mcs.anl.gov/neos/`

NEOS Solvers `www-neos.mcs.anl.gov/neos/server-solvers.html`

Netlib `netlib.bell-labs.com/` or `www.netlib.org/`

NetSolve `www.cs.utk.edu/netsolve/`

Nexus `www.globus.org/nexus/`

NIMBUS `nimbus.math.jyu.fi/`

Ninf `ninf.etl.go.jp/`

Optimization Online `www.optimization-online.org`

Optimization Software Guide (NEOS Guide) `www.mcs.anl.gov/otc/Guide/SoftwareGuide/`

Optimization Technology Center `www.mcs.anl.gov/otc/`

Optimization Tree (NEOS Guide) `www.mcs.anl.gov/otc/Guide/OptWeb/`

OptiW `fb0445.mathematik.tu-darmstadt.de:8081/optiwww/themen.html`

OR-Library `mscmga.ms.ic.ac.uk/info.html`

QAP Case Study `www.mcs.anl.gov/otc/Guide/CaseStudies/qap/`

RIOT `riot.ieor.berkeley.edu/riot/`

SDPLIB `www.nmt.edu/~sdplib/`

semidefinite programming `www.zib.de/helmberg/semidef.html`

servlet `java.sun.com/product/servlet/`

SIF (format) `www.numerical.rl.ac.uk/lancelot/sif/sifhtml.html`

signed applet `java.sun.com/security/signExample/`

SMPS (format) `ttg.sba.dal.ca/sba/profs/hgassmann/smps.html`

sparse SDPA (format) `www.nmt.edu/~sdplib/FORMAT`

submission tool (NEOS Server) `www-neos.mcs.anl.gov/neos/server-submit.html`

Try AMPL! `www.ampl.com/ampl/TRYAMPL/`

TSPfast `www.wxs.nl/ onno.waalewijn/tspfast.html`

TSPLIB `www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/`

TSPx `www.wxs.nl/ onno.waalewijn/tspx.html`

UniCalc `www.rriai.org.ru/UniCalc/calculate.html`

Visualization and Optimization `www.chesapeake2.com/cvj/itorms/`

Windows Terminal Server `www.microsoft.com/ntserver/terminalserver/default.asp`

ZIB MATHPROG `ftp.zib.de/pub/Packages/mathprog/index.html`

References

Arnold, D.C.; Blackford, S.; Dongarra, J.; Eijkhout, V.; and Xu, T. forthcoming, "Seamless access to adaptive solver algorithms," *Proceedings of the 16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation,* Lausanne, Switzerland.

Arora, A.; Cooper, G.; Krishnan, R.; and Padman, R. forthcoming, "IBIZA: E-market infrastructure for custom-built information products," *Information Systems Frontiers.*

Babichev, A.B.; Kadyrova, O.B.; Kashevarova, T.P.; Leshchenko, A.S.; and Semenov, A.L. 1993, "UniCalc, a novel approach to solving systems of algebraic equations," *Interval Computations,* Vol. 2, pp. 29–47. Retrieved 7 July 2000 from `ftp://ftp.rriai.org.ru/pub/articles/int_c_93.tex`.

Bhargava, H.K.; Chowdhary, V.; and Krishnan, R. forthcoming, “Pricing and product design: Intermediary strategies in an electronic market,” *International Journal of Electronic Commerce.*

Bhargava, H.K. and Krishnan, R. 1998, “The World Wide Web: Opportunities for operations research and management science,” *INFORMS Journal on Computing,* Vol. 10, pp. 359–383.

Bhargava, H.K.; Krishnan, R.; and Mueller, R. 1997, “Decision support on demand: On emerging electronic markets for decision technologies,” *Decision Support Systems,* Vol. 19, pp. 193–214.

Brooke, A.; Kendrick, D.; and Meeraus, A. 1992, *GAMS: A User’s Guide, Release 2.25,* Scientific Press/Duxbury Press, Pacific Grove, California.

Carcione, L.; Mould, J.; Pereyra, V.; Powell, D.; and Wojcik, G. 1999, Nonlinear Inversion of Piezoelectric Transducer Impedance Data, Weidlinger Associates, Los Altos, California.

Chinneck, J.W. forthcoming, “Analyzing mathematical programs using MProbe,” *Annals of Operations Research.*

Czyzyk, J.; Owen, J.H.; and Wright, S.J. 1997, “Optimization on the Internet,” *OR/MS Today,* Vol. 24, No. 5 (October), pp. 48–51. Retrieved 7 July 2000 from `http://lionhrtpub.com/orms/orms-10-97/neos.html`.

Czyzyk, J.; Mesnier, M.P.; and Moré, J.J. 1998, “The NEOS Server,” *IEEE Journal on Computational Science and Engineering,* Vol. 5, pp. 68–75.

Czyzyk, J.; Wisniewski, T.; and Wright, S.J. 1999, “Optimization case studies in the NEOS Guide,” *SIAM Review,* Vol. 41, pp. 148–163.

Dongarra, J.J. and Grosse, E. 1987, “Distribution of mathematical software via electronic mail. *Communications of the ACM,* Vol. 30, pp. 403–407.

Dotti, F.L.; Cristal, M. de O.; da Costa, C.M.; Nunes, M.L.; and Müller, F.M. 2000, “Design and implementation of cooperative optimization centers,” 16th International Conference on CAD/CAM, Robotics and Factories of the Future, University of the West Indies, Trinidad and Tobago.

Epema, D.H.J.; Livny, M.; van Dantzig, R.; Evers, X.; and Pruyne, J. 1996, “A worldwide flock of Condors: Load sharing among workstation clusters,” *Future Generation Computer Systems,* Vol. 12, pp. 53–65.

Ferris, M.C.; Mesnier, M.P.; and Moré, J.J. 2000, “NEOS and Condor: Solving optimization problems over the Internet,” *ACM Transactions on Mathematical Software,* Vol. 26, pp. ??–??.

Ferris, M.C. and Munson, T.S. 2000, “Modeling languages and Condor: Metacomputing for optimization,” *Mathematical Programming,* DOI 10.1007/s101070000158.

Foster, I. and Kesselman, C. 1999, *The Grid: Blueprint for a New Computing Infrastructure,* Morgan Kaufmann Publishers, San Francisco, California.

Foster, I.; Kesselman, C.; and Tuecke, S. 1994, “The Nexus task-parallel runtime system,” *Proceedings of the 1st International Workshop on Parallel Processing,* pp. 457–462. Retrieved 7 July 2000 from `ftp://ftp.globus.org/pub/globus/papers/india_paper_ps.pdf`.

Fourer, R.; Gay, D.M.; and Kernighan, B.W. 1990, “A modeling language for mathematical programming,” *Management Science,* Vol. 36, pp. 519–554.

Fourer, R.; Gay, D.M.; and Kernighan, B.W. 1993, *AMPL: A Modeling Language for Mathematical Programming,* Duxbury Press, Pacific Grove, California.

Gay, D.M. 1985, “Electronic mail distribution of linear programming test problems,” *Committee on Algorithms Newsletter,* Vol. 13, pp. 10–12. Also Numerical Analysis Manuscript 86-0, AT&T Bell Laboratories, Murray Hill, New Jersey.

Gay, D.M. 1997, “Hooking your solver to AMPL,” Bell Laboratories, Murray Hill, New Jersey. Retrieved 7 July 2000 from http://www.ampl.com/ampl/REFS/abstracts.html#hooking2.

Good, M. and Goux, J.-P. 1999, “iMW: A web-based problem solving environment for metacomputing applications,” Department of Electrical and Computer Engineering, Northwestern University. Retrieved 7 July 2000 from http://www.mcs.anl.gov/metaneos/papers/imw.ps.

Good, M.; Goux, J.-P.; Nocedal, J.; and Pereyra, V. 2000, “iNEOS: An interactive environment for nonlinear optimization,” Department of Electrical and Computer Engineering, Northwestern University. Retrieved 7 July 2000 from http://www.mcs.anl.gov/metaneos/papers/ineos.ps.

Goux, J.-P.; Kulkani, S.; Linderoth, J.; and Yoder, M. forthcoming, “An enabling framework for master-worker applications on the computational grid,” *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing,* Pittsburgh, Pennsylvania. Retrieved 7 July 2000 from http://www.mcs.anl.gov/metaneos/papers/mw2.ps.

Gropp, W. and Moré, J.J. 1997, “Optimization environments and the NEOS Server,” in *Approximation Theory and Optimization,* Eds. M.D. Buhmann and A. Iserles, Cambridge University Press, pp. 167–182.

Günther, O.; Müller, R.; Schmidt, P.; Bhargava, H.K.; and Krishnan, R. 1997, “MMM: A WWW-based Method ManageMent system for using software modules remotely,” *IEEE Internet Computing,* Vol. 1, pp. 59–68.

Hürlimann, T. 1999, *Mathematical Modeling and Optimization: An Essay for the Design of Computer-Based Modeling Tools,* Kluwer Academic Publishers, Norwell, Massachusetts.

Jones, C.V. 1998, “Visualization and Optimization,” *Interactive Transactions of OR/MS,* Vol. 2, No. 1.

Litzkow, M.; Livny, M.; and Mutka, M.W. 1988, “Condor — A hunter of idle workstations,” *Proceedings of the 8th International Conference of Distributed Computing Systems,* pp. 104–111.

Lucks, M. and Gladwell, I. 1992, “Automated selection of mathematical software,” *ACM Transactions on Mathematical Software,* Vol. 18, pp. 1–34.

Miettinen K. and Mäkelä, M.M. 2000, “Interactive multiobjective optimization system WWW-NIMBUS on the Internet,” *Computers and Operations Research,* Vol. 27, pp. 709–723.

Moré, J.J. and Wright, S.J. 1993, *Optimization Software Guide,* SIAM Publications, Philadelphia, Pennsylvania.

Nakada, H.; Sato, M.; and Sekiguchi, S. 1999, Design and implementations of Ninf: Towards a global computing infrastructure,” *Future Generation Computing Systems,* Vol. 15, pp. 649–658.

Owen, J.H.; Coullard, C.R.; and Dilworth, D.S. 1997, “GIDEN: A graphical environment for network optimization,” *Sixth Industrial Engineering Research Conference Proceedings,* Eds. G.L. Curry, B. Bidanda and S. Jagdale, Institute of Industrial Engineers, Norcross, Georgia, pp. 507–512.

Smarr, L. and Catlet, C.E. 1992, “Metacomputing,” *Communications of the ACM,* Vol. 35, pp. 45–52.

Tawarmalani, M. and Sahinidis, N.V. 1999, “Global optimization of mixed-integer nonlinear programs: A theoretical and computational study,” Department of Mechanical and Industrial Engineering and Department of Chemical Engineering, University of Illinois at Urbana-Champaign. Retrieved 7 July 2000 from http://archimedes.scs.uiuc.edu/papers/comp.pdf.