Project Summary

By allowing people to try out dozens of high-quality optimization packages and interfaces free of the difficulties of downloading and installation, the NEOS Server [12] has provided a software *cyberinfrastructure* [46, 47] that greatly enhances the applicability of large-scale optimization methods. As designers and developers of the NEOS Server at Northwestern University and Argonne National Laboratory, the investigators are uniquely positioned to undertake new research projects that envision ambitious further extensions of the concept of optimization as an Internet resource.

Intellectual merit. The proposed research will address two new project areas, each motivated by the challenges of integrating a general computing concept with the particular difficulties of optimization:

- ▷ Bringing on-demand high-performance computing to researchers and practitioners in large-scale optimization, by integrating advanced computing platforms with the NEOS framework.
- ▷ Standardizing representations and protocols for distributed optimization, by bringing emerging "web services" concepts to bear on services for large-scale optimization.

The proposed research will begin by focusing on a particular project in each of these areas:

- ◇ A NEOS solver that exploits specialized multi-processor computing architectures for purposes of nonlinear optimization over discrete as well as continuous decision variables.
- ◇ An XML-based standard form and application programming interface for a broad range of optimization problem types.

Other challenging projects in each area will be initiated as the research program proceeds. This research will take advantage of developments in discrete nonlinear programming, web services, and high-performance computing that have made substantially more ambitious projects possible in the past few years.

Broader impacts. This project will provide diverse practitioners and researchers with unprecedented access to computing resources for optimization. The proposed standards research will also greatly enhance the interoperability as well as the accessibility of optimization software.

The proposed work will also further the NEOS project's mission of making optimization a part of the worldwide software cyberinfrastructure that supports science, engineering, and commerce. The broad impact of this mission is a direct result of the great variety of application areas in which the NEOS solvers have been applied, including supply-chain management, duty scheduling, combinatorial auctions, agricultural economics, chemistry, roll cutting, petroleum engineering, physics, electrical engineering, neuroscience, circuit design, network design, protein structure prediction, power engineering, process modeling, engineering mechanics and robotics. Indeed, as with other infrastructures, the NEOS Server has proved to have many valuable uses that could not have been predicted in advance.

Project Description

Extending a Cyberinfrastructure to Bring High-Performance Computing and Advanced Web Services to the Optimization Community

The transformation of *products* into *services* has been a major trend in computing and networking over the past decade. In the field of large-scale optimization, this trend has been exemplified by the great success of the ***NEOS Server*** [12] as a *managed service provider* [10] for applications in operations research and diverse areas of science and engineering. By allowing people to try out dozens of high-quality optimization packages and interfaces without having to download and install complex software, the NEOS Server has provided a software *cyberinfrastructure* [46, 47] that greatly enhances the applicability and usefulness of optimization methods.

As designers and developers of the NEOS Server at Northwestern University and Argonne National Laboratory, we are uniquely positioned to undertake new research projects that envision ambitious further extensions of the concept of optimization as a service. We propose in particular to investigate two new project areas, each motivated by the challenges of integrating a general concept from computer science with the particular difficulties of optimization:

- ▷ Bringing on-demand high-performance computing to researchers in large-scale optimization, by integrating advanced computing platforms with the NEOS framework.
- ▷ Standardizing representations and protocols for distributed optimization, by bringing new "web services" concepts to bear on services for large-scale optimization.

For each of these project areas, our proposed research first focuses on a timely project whose potential is likely to be fully realized during the grant period:

- ◇ A multi-processor NEOS solver for nonlinear optimization over discrete as well as continuous decision variables.
- ◇ An XML-based standard form and application programming interface for a broad range of optimization problem types.

We also cite other challenging projects in each area that we plan to initiate during the grant period, as our research program proceeds.

Our projects will have the benefits of high-performance computational equipment and expertise at Argonne National Laboratory, as well as an active doctoral program at Northwestern University. We will also be able to take advantage of recent developments in discrete nonlinear programming, web services, and high-performance computing so as to undertake projects that are substantially more ambitious than what could have been realistically proposed even three years ago.

Section 1 of this proposal places our work within the context of cyberinfrastructure and the optimization community, both in general terms and in the specific case of the NEOS Server. Sections 2 and 3 then present detailed descriptions of the proposed research in the two project areas that we cited above. Section 4 describes the broad impact that we expect this work to have, and section 5 summarizes results from previous NSF support that serve as the foundation for the proposed research.

1. Background and goals

We begin by describing the place of Operations Research within NSF's broad cyberinfrastructure (CI) initiative; this presentation is adapted from the report [46] of an NSF workshop that we helped organize. We then tie our work to the CI initiative by presenting the NEOS Server as a testbed for research into software cyberinfrastructures for optimization.

Cyberinfrastructure and operations research. We are all familiar with infrastructures: road systems, rail networks, power grids. An infrastructure does not produce goods or services itself; rather, it makes a wide range of productive activities possible. The interstate highway infrastructure does not itself carry out supply-chain management, for example, but it permits the development of supply-chain management systems that would not be possible otherwise. Indeed, it paves the way for phenomena that were not foreseen when it was built, such as crossdocks and suburban sprawl. The effectiveness of infrastructures depends critically on standards (track gauges and standard time for railroads, bridge heights for highways, voltages for power grids) and on accessibility to a broad base of users.

Among the major infrastructures of modern life, cyberinfrastructures constructed from computers, data networks, software, and communications standards are among the newest and most elaborate instances. The Internet and the Web are the best known examples. Like other infrastructures, they facilitate myriad applications — the web's use for unexpected purposes is already legendary — and they depend critically on software standards such as HTTP and HTML.

Operations Research also has the characteristics of an infrastructure, in the sense that it is a collection of theory, algorithms, and software that underpins and facilitates productive activity. OR has had major impacts in design, manufacturing, and services, without itself being any one of these things. Like infrastructures generally, OR serves many purposes that were not envisioned by its creators.

The potential benefits are great, but weaknesses in the key criteria of standards and accessibility currently prevent OR from truly filling the role of an infrastructure. Although hundreds of excellent OR software tools have been produced by academia and industry, they communicate with each other and with their applications in *ad hoc* ways and through awkward interfaces. Individual OR researchers and practitioners have ready access to only a small fraction of available tools, and are able to implement and validate far fewer ideas than they can generate. The OR community is consequently not well positioned to exploit new computing resources, in particular the high performance, multi-processor computing resources commonly associated with CI.

Many of the dangers of foregoing or under-funding CI initiatives cited in the NSF panel report of Atkins *et al.* [47] apply directly to the OR community. OR problems are pervasive and yet there is a low degree of sharing. CI tools can promote the development of core OR, as well as OR cyberinfrastructures, by enabling research collaborations across institutions, locations, time, and fields of endeavor. CI expertise is also needed to ensure that data and software acquired at great expense and effort are available for future researchers. Incompatible software tools and structures that isolate the OR community (and isolate researchers from each other within the community) need to be replaced, and OR researchers must take the lead in fostering interoperability. There must be investment in the maintenance of successful OR

products so that time and talent is spent in breaking new ground rather than in reproducing past efforts.

Cyberinfrastructure and optimization. Over the past decade, we and our collaborators have designed and built the NEOS Server, the preeminent software cyberinfrastructure for the support of large-scale optimization. This work has revolutionized the activities of a broad *optimization community* having its roots in operations research, the management sciences, computer science, and numerous engineering and scientific disciplines.

The NEOS Server has revolutionized optimization research, teaching, and practice, by providing immediate access to far more solvers than optimization users could hope to install locally. Even many commercial solver developers have made their products available through the Server, to encourage potential customers to try them out. We have a lengthy collection of testimonials to the Server's value (see section 5), and it was recognized in 2003 with the Mathematical Programming Society's triennial Beale-Orchard-Hays Prize for Excellence in Computational Mathematical Programming.

For the optimization community, the NEOS Server provides the characteristics generally associated with a cyberinfrastructure:

- ◇ facilitating applications though not directly carrying them out;
- ◇ enabling more applications than were originally imagined;
- ◇ providing open access to Internet-based resources;
- ◇ encouraging standards for information interchange.

We see the NEOS Server as becoming not simply a stand-alone tool for the optimization community, moreover, but a resource that is interoperable with other analytic activities in business, science, and engineering. Towards this end, we have recently rewritten the Server to use now-standard conventions for data transfer (XML) and remote procedure calls (XML-RPC). At the same time we are pursuing a more ambitious project to define standards for representing optimization problem instances [24] and to provide a general framework for optimization services, as a foundation for what can be viewed as the "next generation NEOS" [44].

The NEOS Server was designed from the outset to be scaled up; it has one central server, but farms out solver requests to workstations that can be anywhere on the Internet. Loads as high as 10,000 requests in a week have been handled without evident strain on the system. Nevertheless, the Server's design relies on one central computer to handle communications and scheduling. Until now, moreover, its distributed facilities have been no more than single-processor workstations. Thus the NEOS Server has yet to play a part in bringing the optimization community into the world of high-performance computing and web services, and its ability to help with the very hardest and largest optimization problems has remained limited. The goal of our proposed research is to address these limitations.

2. Proposed research on Access to advanced computing resources for optimization

"Advanced" computing can mean any of several ways to use multiple processors and networks to accomplish what cannot be done effectively by individual computers, including:

- ◇ high-performance computing, using large numbers of specialized processors and specialized interconnections;
- ◇ distributed computing, using conventional computers working together through Internet connections;
- ◇ high-throughput computing, using the computational resources of networked computers that would otherwise go idle.

A great variety of optimization problems have features that permit advanced computing to be used to advantage. For example, the metaNEOS project of 1997–2001 applied advanced computing approaches in solving all of the following:

- ▷ the 10^{10}-variable deterministic equivalent of a 10^7-scenario stochastic program on a computational grid of about 800 workstations, in about 32 hours of wall-clock time [39];
- ▷ a previously intractable quadratic assignment problem using an average of 650 worker machines over a one-week period, providing the equivalent of almost 7 years of computation on a single workstation [4];
- ▷ a mixed-integer nonlinear programming problem with parallel efficiency of up to 80% on 600 million search-tree nodes [28].

How many applications have benefited from the pioneering work described above? Essentially, none. Indeed, only a tiny fraction of people trained in optimization have experience with any kind of advanced computing. For most members of the optimization community, whose focus is modeling and solving rather than computing, it is a daunting (and disheartening) challenge to assemble and configure the hardware and software resources necessary to apply or even experiment with such advanced computational approaches.

To bring advanced computing of this kind to the optimization community in a realistic way, we must provide a cyberinfrastructure that packages the advanced computing resources behind an interface appropriate to optimization modelers. The NEOS Server is the obvious candidate to provide the desired cyberinfrastructure. Nevertheless, the Server will only be able to succeed in this goal after it has met a number of significant research and implementation challenges, both to enable users to efficiently interact with solvers, and to enable advanced computing systems to effectively run such solvers on demand.

As an initial project to prove the effectiveness of making advanced computing resources available through NEOS, we will address the problem of nonlinear optimization over a combination of integer and continuous decision variables. We have the advantage of experience in this area gained from the metaNEOS project [28] listed above. The proposed project will develop substantially new algorithmic ideas, however, and will employ new advanced computing platforms. Lessons from this project will later be put to use in making advanced computing resources available

through NEOS for other difficult problems such as stochastic programming, routing, and quadratic assignment.

Online parallel integer nonlinear optimization. Many operations, engineering, and scientific design applications involve discrete parameters that affect the optimality of the final design. Problems of this type are modeled as mixed integer nonlinear programming problems (MINLPs), which are conveniently expressed in terms of smooth functions f and c as

$$\begin{aligned} \text{(MINLP)} \qquad & \underset{x,y}{\text{Minimize}} \ f(x,y), \\ & \text{Subject to} \ \ c(x,y) \geq 0, \quad y \ \text{ integer}. \end{aligned}$$

MINLPs combine the difficulties of optimizing over integer-valued variables with the challenges of handling nonlinear functions. Applications include blackout prevention for electric power systems [16], the design of batch plants [31, 34], the synthesis of processes [17], the design of distillation sequences [52], the optimal positioning of products in a multi-attribute space [17], the minimization of waste in paper cutting [53], and the optimization of core reload patterns for nuclear reactors [49]. The surveys [29, 30] and the monograph [22] list numerous other applications.

We restrict our attention to MINLPs with convex functions f and c, which are common in many engineering and scientific applications. Convex functions have the advantage of producing guaranteed lower bounds when the integrality of variables is relaxed. In the subsequent discussion of further technical work we indicate the novel approach that we will take in extending ideas from the convex case to the more general nonconvex case.

Two factors make MINLPs an ideal candidate for a demonstration project to illustrate the benefits of a cyberinfrastructure that can bring advanced computing to the optimization community. First, the solver "MINLP" [37] is one of the most popular solvers currently available through the NEOS Server. Since 1999 we have solved over 100,000 MINLP problems, ensuring that our work will have a large impact. Second, the tree-search nature of the MINLP solver's algorithm has a particularly strong potential for exploiting parallel algorithms.

Prior work using advanced computing. The "MINLP" solver uses a *branch-and-bound* approach that is most conveniently viewed in terms of a tree search. Initially, all integer restrictions are dropped and the resulting continuous nonlinear *relaxation* is solved. The algorithm then selects one integer variable that takes a fractional value in the relaxation, say $y_i = \hat{y}_i$, and branches on it by generating two new problems with additional bounds $y_i \leq \lfloor \hat{y}_i \rfloor$ and $y_i \geq \lceil \hat{y}_i \rceil$ respectively. The relaxations of these new, more constrained problems are then solved. The repetition of this process generates the branch-and-bound tree, with a relaxed subproblem at each node. A node of the tree can be fathomed — removed from further investigation — if all variables of its associated relaxation come out integer, or if it is infeasible or has a lower bound that is worse than the current best integer solution.

The tree structure of branch-and-bound gives rise to a natural parallelism with little communication. There have been several implementations of parallel branch-and-bound algorithms. Pekny and Miller [48] implement parallel branch-and-bound to solve asymmetric traveling salesman problems on a BNN butterfly multiprocessor. Eckstein [18] implements a parallel branch-and-bound solver for mixed-integer *linear*

programming (MILP) on a CM-5. Chen and Ferris [8], see also [9], implement a parallel MILP solver on a computational grid. Anstreicher *et al.* [4] solve some large quadratic assignment problems on a computational grid. Laursen [36] shows that parallel branch-and-bound without communication between workers can be efficient, provided a good initial solution is known. Androulakis and Floudas [3] implement a parallel branch-and-bound algorithm for nonlinear global optimization. Sandia has developed PICO [19], a massively parallel optimization framework for integer linear optimization.

In the metaNEOS project [28] cited earlier in this section, we develop a parallel branch-and-bound solver for heterogeneous distributed workstations on a computational grid managed by Condor [41]. The parallelization strategy employs a master-worker paradigm in which the master manages a pool of MINLP tasks corresponding to subtrees of the branch-and-bound tree, illustrated in Figure 1 (left). Each worker solves a MINLP task by nonlinear branch-and-bound. The right image in Figure 1 shows (clockwise from top left) the number of workers, the lower and upper bounds, the number of problems on the stack, and the number of NLPs per task.

Proposed research. We propose to develop a new parallel branch-and-bound solver for MINLPs, based on our metaNEOS work [28], but capable of running efficiently on Argonne National Laboratory's high-performance computing (HPC) platforms: a Linux cluster with 350 Pentium 4 nodes, and a BlueGene machine with 1024 dual processor PowerPC 440 nodes.

HPC platforms pose new challenges for a parallel MINLP solver, because of the rigid way in which resources are allocated. The amount of resources (in terms of processors and time) must be specified ahead of the run. This is usually hard to do for challenging optimization problems such as MINLPs, as the amount of work can vary dramatically even among problems of the same size. At best, the remaining amount of work required to solve a MINLP can be estimated after the solution process has been running for a while.

We therefore propose to bracket the parallel solver by serial pre-solve and post-processing stages. The pre-solve stage will prepare the MINLP for the parallel solve and will estimate the resources required. The post-processing stage will check the solution and if necessary will pre-process for an additional parallel solve. This general framework is illustrated in Figure 2.

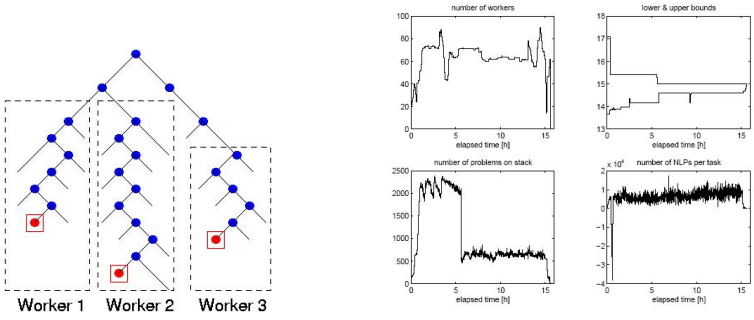


Figure 1: Parallel branch-and-bound strategy (left)
and result of Condor run on `trimlon7` problem (right).

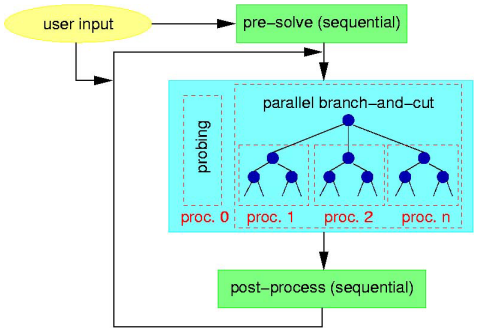


Figure 2: New parallel branch-and-bound strategy.

The pre-solve phase will start solving the MINLP sequentially using enhancements known to work well for branch-and-bound [38]. An initial tree will be generated using a branch-and-cut approach and strong branching, providing strong cutting planes, good estimates for pseudo-costs, and priorities for the integer variables. We envisage that by observing the development of the bounds after solving a few hundred node subproblems in this way, we will be able to estimate the resources required to solve the MINLP to a specified optimality gap. Hence this approach will allow problem-specific scheduling of the MINLP tasks. An additional advantage is that easy MINLPs can be solved in this stage without wasting parallel resources.

The tree that is generated during the sequential pre-solve stage will provide the initial task pool for our MINLP solver for HPC environments. The parallel-search stage will run for a pre-specified amount of time on a given number of processors. We will experiment with the possibility of running a few probing nodes that exploit heuristic MINLP techniques in an attempt to find good integer solutions quickly. Once an initial integer solution has been identified, these probing nodes will become regular branch-and-cut nodes. This design is similar to that of our grid-based MINLP solver [28], except that new resources cannot easily be added. Thus, we require a post-processing stage.

The post-processing stage has two phases. In the first (parallel) phase, we clean up the tree by removing as many open nodes as possible in order to return a tree of manageable size to the sequential phase. The sequential post-processing phase either passes the solution to the user, or parses the tree to improve the branching heuristic employed in the parallel search. It generates a new resource request, based on updated estimates on the amount of resources required to solve the MINLP.

Initially, we will deploy a simplified post-processing stage that only collects the results from the parallel stage, and returns the best solution found together with lower and upper bounds to the user. This will allow us to quickly gain experience with real applications submitted to NEOS, which will guide more sophisticated strategies later. Another advantage of this layered approach is that it provides feedback on the progress of the solution without requiring write statements from the individual processors.

Further technical work. We will explore extensions to the basic MINLP solver in two areas. First, we will develop an approximate strong branching strategy based on a mixed integer quadratic approximation [21]. This permits a strong branching decision at a fraction of the cost of standard NLP strong branching. In

addition, we will develop new branch-and-cut ideas for MINLP. We have identified the following classes of cuts that are likely to be useful in MINLP: knapsack covers, knapsack covers with a single continuous variable, mixed integer rounding, and disjunctive MINLP cuts. We will generalize these cuts to MINLPs by applying them to linearizations of the nonlinear functions similar to [2]. The linearizations about (x_k, y_k) with optimal multipliers z_k,

$$\eta \geq f_k + \nabla f_k^T \begin{pmatrix} x - x_k \\ y - y_k \end{pmatrix} \quad \text{and} \quad 0 \geq c_k + \nabla c_k^T \begin{pmatrix} x - x_k \\ y - y_k \end{pmatrix},$$

are valid inequalities (where $f_k = f(x_k, y_k)$, etc.).

We have observed that the structure of MINLPs is often such that the nonlinear constraints involve only the continuous variables x, making it impossible to apply the first three cut classes directly. However, we can remedy this situation by summing the linearizations weighted with $(1, z_k)$, which gives the valid cut

$$\eta \geq f_k + z_k^T c_k + \left(\nabla f_k + \nabla c_k^T z_k\right)^T \begin{pmatrix} x - x_k \\ y - y_k \end{pmatrix}.$$

Now observe that $z_k^T c_k = 0$ and $\nabla_x f_k + \nabla_x c_k^T z_k = 0$ by the optimality of (x_k, y_k), giving the valid inequality

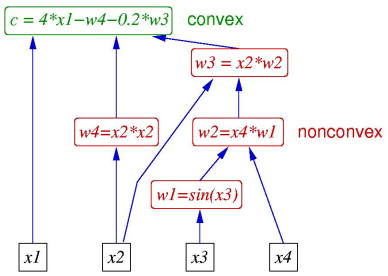
$$\eta \geq f_k + \left(\nabla_y f_k + \nabla_y c_k^T z_k\right)^T (y - y_k),$$

which can be shown to be equivalent to the corresponding Benders cut. This cut has the correct format to derive a knapsack cover with a single continuous variable.

Second, we propose a new approximation approach to multidimensional functions that exploits recent advances in automatic differentiation and generates tighter relaxations from convex approximations. Most functions can be expressed as a series of unary (e.g., $\sin(x_1)$) and binary (e.g., x_2/x_3) operations. The following example is motivated by the modeling of electrical transmission networks. Consider the constraint

$$4x_1 - x_2^2 - 0.2 \cdot x_2 x_4 \sin(x_3) \leq 1 \tag{2.1}$$

with suitable bounds on x_i. One possible realization of this constraint as a computational graph is given in Figure 3, and the corresponding nonlinear system of unary and binary equations is given in (2.2), where we have folded the linear expression $c = 4x_1 - w_4 - 0.2w_3$ into a single node.



$$4x_1 - w_4 - 0.2w_3 \leq 1 \tag{2.2a}$$
$$w_4 = x_2^2 \tag{2.2b}$$
$$w_3 = x_2 w_2 \tag{2.2c}$$
$$w_2 = x_4 w_1 \tag{2.2d}$$
$$w_1 = \sin(x_3) \tag{2.2e}$$

Figure 3: Computational graph of (2.1) and (2.2).

The first constraint is now convex, and the last four constraints are approximated by SOS-2 and SOS-3 variables [51, 45]. We note that the convexification of individual expressions significantly reduces the number of SOS variables that are needed. The SOS-5 approximation used in [45] would require N^4 SOS variables to discretize the constraint of four variables, whereas our approach requires only $2 \times N^2 + 2 \times N$ variables, where N is the number of discretization points in each dimension.

We will develop an SOS-reformulation tool based on the new standard for representing optimization problems that is described in the next section. Thus, our reformulation tool will be independent of the modeling language or the input format, making it more widely applicable. In addition, we will mention the new convexity and problem analysis tools described in the next section to derive tighter bounds and exploit convex nodes of the expression tree, leading to a tighter approximation.

3. Proposed research on Optimization Services

When the first version of NEOS was created about a decade ago, the World Wide Web was just coming into general use, and Web-based services were built using whatever tools were handy. Successive versions of NEOS have been able to take advantage of more standard tools that have become available. Thus the current NEOS 5 uses XML tags for input files and XML-RPC for remote procedure calls.

In a project with Kipp Martin of the University of Chicago, we have been working out the details of a new generation of NEOS-like systems, based on specialization of current "web services" standards to produce a new concept of ***Optimization Services.*** We envision an OS framework of languages and protocols that supports a next generation of distributed optimization systems. Much of this framework is described in the dissertation [44] of Jun Ma, who continues to be an essential member of this project as a postdoctoral associate.

We begin this section by describing more fully our vision for the Optimization Services framework. Since a detailed description of all the OS components introduced in [44] would be far too long for this proposal, we focus on one aspect of OS that is likely to be the first to have a widespread effect in the optimization community: the design of new standard forms for representing optimization problem instances.

The Optimization Services framework. This project represents the first systematic approach to addressing and solving general issues in optimization system and software development, and to standardizing all major instance representations and communications in distributed optimization systems. It defines a framework in the sense of specifying how a set of cooperative classes and interfaces should be designed and implemented in order to solve optimization problems. To this end,

- ◇ it consists of multiple classes or components, each of which may provide an abstraction of some particular optimization concept;
- ◇ it defines how these abstractions work together to solve an optimization problem;
- ◇ its optimization-related components are reusable, providing a generic behavior that many different types of applications can make use of;
- ◇ it organizes patterns of frequently repeated activities at a higher level.

Its design employs an architecture that is XML-based, service-oriented, optimization centered, distributed and decentralized. The associated OS Protocol is an application-level networking protocol that includes over 20 specifications of specialized communication languages or vocabularies. Optimization within a local environment is covered as a special case.

Through standardization of modeling representation, communication, discovery, and registration, the OS framework provides an open infrastructure for all optimization system components including modeling language environments, servers, registries, communication agents, interfaces, analyzers, solvers, and simulation engines. The goal is that all algorithmic codes will be implemented as services under this framework and that customers will use these computational services much like utilities, with special knowledge of optimization algorithms, problem types, and solver options required no more than necessary. A supply chain modeler, for example, will concentrate on writing a good supply chain model, while the steps of detecting the problem structure, finding appropriate solvers, applying solvers to problem instances, providing sufficient computing resources, and retrieving the solution will be handled automatically. The combination of distributed system embedded intelligence, smooth coordination of tasks, and seamless integration is what makes Optimization Services unique and significant.

The OS framework retains the goals of NEOS while addressing many outstanding design and implementation challenges faced by the current NEOS Server in a large-scale, distributed optimization environment. The benefits of the framework's common format for instance representation and standard application programming interface (API) for solvers are considered in the next subsection. Its distributed design also contrasts with the tightly coupled centralized structure of NEOS, wherein all solvers are connected to the server and all optimization requests must go through it.

Optimization Services adopts a decentralized *service-oriented architecture* that scales much more readily than a centralized design. There is still in some sense a "central" server in the middle, but it functions as a lightweight *registry* that maintains information about available optimization software. No solvers are actually executed by this registry; instead users directly contact the solvers in a peer-to-peer mode. The advantages of such a decentralized architecture are highly significant. Indeed the Internet has become popular precisely because it has a decentralized architecture; there is no such thing as a "central repository server" that hosts all the Web pages. Development and maintenance happen spontaneously. It is our vision that this kind of decentralized architecture can better promote research and development in Operations Research.

Design of a standard for optimization problem instances. In a distributed environment of the kind we contemplate for the OS framework, the modeling language software, solver software, and data used to generate an optimization problem instance may reside on different machines using different operating systems. Thus it is essential to have an open standard for exchanging problem instances.[1] Current optimization software is limited by its reliance on a plethora of

[1]By *instance* we mean a particular problem for which answers can be sought in the form of specific values for decision variables, in contrast to a *model* that is a description of a class of optimization problems. Typically a model is a *symbolic*, *general*, *concise,* and *understandable* representation of an optimization problem, whereas an instance is an *explicit*, *specific*, *verbose,* and *convenient* description of a problem's objective and constraints [23]. Thus a model plus data

input formats, as can be seen by even a cursory look at the list of solvers available on the NEOS Server. The nearly 50 solvers in the NEOS lineup require instance inputs of about a dozen different kinds, including MPS [33] and various "LP" formats for linear and integer programming, SMPS extensions to the MPS format for stochastic programming, formats such as SDPA specific to semidefinite programming, DIMACS min-cost flow and other formats for network linear programming, and proprietary formats used by two modeling language processors. Other solvers recognize input programmed as functions in various languages including Fortran, C, C++, and Matlab.

A number of broader new standards have been proposed (if not widely adopted) in recent years. Various extensions of the MPS format to nonlinear programming have been put forward, notably the xMPS format by Halldórsson, Thorsteinsson and Kristjánsson [32]. However, most recent proposals have been based on XML (Extensible Markup Language [50]), which has become the established standard for communicating data between web services. Fourer, Lopes and Martin [24] propose the XML-based language LPFML for representing instances of mixed-integer linear programs; Chang [7] and Kristjánsson [35] have also proposed XML representations for linear programming instances. Ezechukwu and Maros [20] describe an Algebraic Markup Language that uses XML to describe the model rather than the instance, and Bradley [6] introduces an XML markup grammar for networks. See also [5] for a good overview of the uses of XML technologies in operations research.

Proposals of new APIs to standardize interactions with solvers have also been a subject of recent activity. The extensive COIN-OR (COmputational INfrastructure for Operations Research) project (Lougee-Heimer [43] or `www.coin-or.org`) includes the OSI (Open Solver Interface) library, an API for linear programming solvers, and NLPAPI, a subroutine library with routines for building nonlinear programming problems. Another nonlinear interface, MOI (Modeler-Optimizer Interface), is proposed along with xMPS in [32], and a similar interface is used in the LINDO API [40].[2]

As part of our OS framework design we have developed OSiL, an XML-based language designed as a new standard for representing optimization problem instances of many significant kinds. As a brief example, Figure 4 shows the part of one OSiL file that specifies the decision variables; a much more complete collection of examples appears in [25]. OSiL serves as an instance-level format flexible enough to handle linear and mixed-integer programs, quadratic programs, and very general nonlinear programs, while its underlying principles are sufficiently powerful to allow for future extensions. Thus it has the potential to serve as a new standard that subsumes the many currently used input formats.

Designing a standard form may appear to involve not much more than adding "tags" (delimited by < and > characters) to structured text files. In fact there is con-

is required to generate an instance. A linear programming model is typically described by linear algebraic expressions, for example, while the corresponding instance is represented as a list of nonzero coefficients of variables in the objective and constraints, along with bounds on the variables and the constraint expressions.

[2]These are based on representing the nonlinear part of each constraint and the objective function in postfix (reverse Polish) notation [1] and then assigning integers to operators, characters to operands, and integer indices to variables so that the data structure corresponds to the implementation of a stack machine.

```
<variables numberOfVariables="2">
   <var lb="0" name="x0" type="C"/>
   <var lb="0" name="x1" type="C"/>
</variables>
```

Figure 4: Excerpt from an OSiL file,
declaring two continuous variables x_0, $x_1 \geq 0$.

```
<xs:complexType name="Variables">
   <xs:sequence>
      <xs:element name="var" type="Variable" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="numberOfVariables"
                          type="xs:positiveInteger" use="required"/>
</xs:complexType>
```

Figure 5: The schema corresponding to Figure 4.

siderable research that must lie behind an XML-based form for anything as complicated as optimization, as we have learned in this initial part of our project. Every XML vocabulary must be defined by a "schema" in strictly specified ways; Figure 5 shows the schema for the `<variables>` tag used in Figure 4. For nonlinear expressions, we wanted to be able to take advantage of the natural tree structure of XML files, as shown in Figure 6, but many considerations were involved in determining whether a schema could be devised to permit such an arrangement to be written naturally and processed efficiently. We were able to take advantage of a feature of schemas that permits a single abstract XML element to be specialized to elements representing over a hundred different operators and functions.

We also had to consider how OSiL files would be used. Although there are standards for shipping XML files around the Internet (SOAP [11]), the content of an OSiL file must at some point be converted to an in-memory data structure, which we call `OSInstance`. We found that `OSInstance` could be defined almost exactly in parallel with the OSiL schema, even in the tricky nonlinear case, avoiding all of the

```
<nl idx="1">
    <cos>
        <plus>
            <variable idx="0"/>
            <variable idx="1"/>
        </plus>
    </cos>
</nl>
```

Figure 6: The OSiL element for the nonlinear expression $\cos(x_0 + x_1)$.

complication that would attend the definition of a separate data-structure standard. A single API then handles reading and writing of both OSiL files and `OSInstance` structures. Indeed, the most critical test for OSiL will come when its API is put into use to begin connecting actual modeling systems and solvers.

The proposed research will take this initial work much further. To be truly comprehensive, OSiL and `OSInstance` will have to be extended to provide standards for additional difficult cases, including complementarity and disjunctive constraints, and cone, constraint, stochastic, and semidefinite programming. XML-based languages for communicating solver options (OSoL) and for returning results (OSrL) will need to be established before OSiL can be truly useful for interfacing modeling systems with solvers. Complete realization of the OS framework will require study and design of numerous other XML dialects, for example to permit communication between the modeler, the central repository, and various problem analyzers.

Establishing a standard. Whatever its good qualities may be, a format or representation does not become a standard until it is widely used. Thus in our dissemination of this work we will be particularly concerned to take steps that will promote its wide use.

In support of OSiL and its associated languages, APIs, and protocols, we will contribute a project to the previously mentioned COIN-OR website. A tie-in with COIN-OR's efforts to disseminate high-quality open-source OR software will help to provide visibility for our efforts. Moreover, although COIN-OR is not a standards-setting body at present, we may urge it to consider that role within the next few years. Our interest in working with COIN-OR is attested by the recent election of two participants in the Optimization Systems project (Profs. Fourer and Martin) to the organization's governing boards.

We will also produce open-source software to demonstrate the potential of our proposed standards. We will make the APIs for our project available in several popular languages; as this is written, we already have a complete API for OSiL in Java, and an API for the linear part of OSiL in C++. We will also write interfaces from our standards to popular solvers, and will provide several kinds of conversion utilities to promote the initial use of our standards. A translator from AMPL's `nl` format [27] to OSiL, for example, will enable the AMPL modeling language to send problems to any solver that has an OSiL interface; similar ties to OSoL and OSrL will allow for a complete AMPL interface.

To further demonstrate the potential of OSiL, we will develop two tools that will work on the `OSInstance` expression tree. One, the SOS-reformulation utility for mixed-integer nonlinear optimization, has been discussed in the previous section. The other will be a convexity detection and analysis tool based on DrAMPL [26], a sophisticated "metasolver" that analyzes AMPL problem instances to symbolically prove or disprove convexity and derive tight bounds on nonlinear expressions. We will develop a language-independent version of DrAMPL that can be applied directly to the `OSInstance` nonlinear data structure, making this tool potentially usable with a broad class of modeling languages. Like our APIs and conversion utilities, these tools will be contributed to COIN-OR to promote their widespread use within the optimization community.

4. Broader Impact

From its outset, the NEOS project's mission has been to make optimization a part of the worldwide software cyberinfrastructure that supports science, engineering, and commerce. Its broad impact has thus primarily been a function of the great variety of applications to which its solvers have been put. Indeed, as with other infrastructures, the NEOS Server has proved to have substantial uses that could not have been predicted in advance. A sample of applications that have been reported to us, collected at `neos.mcs.anl.gov/neos/stories.html`, includes such varied areas as combinatorial auctions, supply-chain management, duty scheduling, agricultural economics, chemistry, roll cutting, petroleum engineering, physics, electrical engineering, neuroscience, circuit design, network design, protein structure prediction, power engineering, process modeling, engineering mechanics and robotics.

A second kind of broad impact derives from the NEOS Server's influence as a successful cyberinfrastructure. Optimization, with its numerous independent modeling and solving packages, was a natural area in which to try out the algorithm-server idea. But we believe that it will now lead to consideration of similar ideas in other areas that employ complex computational software of a general-purpose nature.

Our collection of applications also includes a large number from educational settings of all kinds and at all levels. In addition to the many cases where NEOS is used in teaching optimization itself, it is used more broadly in courses where optimization is a key component, such as microeconomics and civil engineering design. We have also heard from many graduate students who find it valuable to their research.

The factors that underlie the NEOS Server's broad impact to date are equally relevant to the two research directions set forth in this proposal. Thus we are confident that the proposed research will have the same broad impact.

5. Results of Prior NSF Support

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

Robert Fourer and Jorge J. Moré, "Next-Generation Servers for Optimization as an Internet Resource": *Grant DMI-0322580, $374,969 plus $17,820 supplement for September 2003 through August 2006.*

The greatest part of our research under these grants has centered on the development of the NEOS Server as a managed service provider for large-scale optimization problems, as presented in Section 1. Nearly 175,000 optimization requests were handled in 2005, for many educational, research, and commercial projects as described in Section 4's discussion of the Server's broader impact. Our MINLP solver has handled over 75,000 submissions since 1999, making it the most popular NEOS solver.

These grants have fostered the development of human resources in part by supporting the writing of two doctoral dissertations, by Leo Lopes [42] and Jun Ma [44]. They have also supported two administrator/developers of NEOS, Elizabeth Dolan and Jason Sarich, and one postdoctoral associate, Dominique Orban.

Representation standards for optimization problem instances. An initial XML-based standard for linear and integer programming was developed in col-

laboration with Leo Lopes and Kipp Martin [24]. The size of the XML files for linear programs was expected to be the greatest obstacle, but in fact the parsing speed required the most attention.

A subsequent project with graduate student (now postdoctoral fellow) Jun Ma [44] showed that nonlinear expression trees translate nicely to tree-oriented XML representations, as discussed in Section 3. Challenging design issues also emerged in handling user-defined functions, database references in problem instances, and extensions for stochastic programming.

Analyzing and categorizing optimization problems prior to solving. A collaboration with Dominique Orban of École Polytechnique de Montréal produced methodologies for determining optimization problem types — including new approaches to attempting a proof or disproof of convexity — and for using a database of solvers to determine which are most suitable to a given problem. John Chinneck of Carleton University, Ottawa worked with us to adapt some of his convexity-analysis software so as to provide another point of comparison.

We found it necessary to include in our tests a wider variety of convexity-disproving methods than expected. Also, in accumulating the expression bounds that are essential to convexity proving, we found unexpected connections to theories of domain reduction in constraint programming.

Internet services for optimization. Version 5 of the NEOS Server, released in August 2005, represented a complete revamping of the Server's infrastructure using modern software tools. Version 5 is implemented in Python, uses XML-RPC for remote function calls, and improves resource utilization through a simple scheme that differentiates between long-running and short-running submissions. The Python rewrite significantly reduced the code base size while improving readability and maintainability.

The use of XML-RPC clients — freely available for C, C++, Java, Perl, Python, Ruby, and other languages — introduced added flexibility as expected, but also made possible a well-documented application programming interface for NEOS. This interface greatly simplified the writing of software to submit optimization requests, retrieve results, and communicate during execution. XML-based formats were introduced for all communications.

The use of a database to record all submissions was another innovation of NEOS 5. Complex database queries to obtain statistics and other information on the submissions were made possible as a result.

Work on an ambitious optimization services framework for distributed environments proceeded on several levels, as described in detail in Section 3.

Benchmarking and verification. The supported work by Dolan and Moré on performance profiles [13] was influential and widely adopted. It led to a study by Dolan, Moré, and Munson [14] on optimality measures and associated convergence tests, and their impact on benchmarking. We found that important differences in performance profiles arise when benchmark studies employ a consistent convergence test. Results show that almost all of the current optimization solvers use convergence conditions that are not scale invariant.

Another report [15] by the same authors documents extensions and improvements to the COPS test-problem set and provides a comparisons of five solvers. The COPS problems and testing framework have been released to the public.

References Cited

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, Reading, MA, 1986.

[2] I. Akrotirianakis, I. Maros, and B. Rustem. An outer approximation based branch-and-cut algorithm for convex 0-1 MINLP problems. Technical report 2000-06, Department of Computing, Imperial College, London, June 2000.

[3] I. P. Androulakis and C. A. Floudas. Distributed branch and bound algorithms for global optimization. In P. Pardalos, editor, *Parallel Processing of Discrete Problems*, volume 106 of *The IMA Volumes in Mathematics and Its Applications*. Springer, 1999.

[4] K. Anstreicher, N. Brixius, J.-P. Goux, and J. Linderoth. Solving large quadratic assignment problems on computational grids. *Mathematical Programming*, 91:563–588, 2002.

[5] G. Bradley. Introduction to extensible markup language (XML) with operations research examples. *Newletter of the INFORMS Computing Society*, 24:1–20, 2003.

[6] G. Bradley. Network and graph markup language (NaGML) — data file formats. Technical Report NPS-OR-04-007, Department of Operations Research, Naval Postgraduate School, Monterey, CA, USA, 2004. Available from the author, `bradley@nps.navy.mil`.

[7] T-H. Chang. Modelling and presenting mathematical programs with XML:LP. Masters thesis, Department of Management, University of Canterbury, Christchruch, NZ, 2003.

[8] Q. Chen and M.C. Ferris. FATCOP: A fault tolerant condor-PVM mixed integer program solver. *SIAM Journal on Optimization*, 11(4):1019–1036, 2001.

[9] Q. Chen, M.C. Ferris, and J. Linderoth. FATCOP 2.0: advanced features in an opportunistic mixed integer programming solver. *Annals of Operations Research*, 103:17–32, 2001.

[10] Stacy Collett. MSPs: The new hosts. *Computerworld*, 39(46):62–64, 2005.

[11] World Wide Web Consortium. SOAP version 1.2 part 0: Primer. W3C recommendation, 24 June 2003.

[12] E.D. Dolan, R. Fourer, J.J. Moré, and T.S Munson. Optimization on the NEOS server. *SIAM News*, 35(6):4, 8–9, 2002.

[13] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[14] E.D. Dolan, J.J. Moré, and T.S. Munson. Optimality measures for performance profiles. *SIAM Journal on Optimization*, 16:891–909, 2006.

[15] E.D. Dolan, J.J. Moré, and T.S. Munson. Benchmarking optimization software with COPS 3.0. Technical report ANL/MCS-TM-273, Mathematics and Computer Science Division, Argonne National Laboratory, February 2004.

[16] V. Donde, V. López, B. Lesieutre, A. Pinar, C. Yang, and J. Meza. Identification of severe multiple contingencies in electric power networks. In *Proceedings of the 37th North American Power Symposium*. Iowa State University, October 2005.

[17] M. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed–integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.

[18] J. Eckstein. Parallel branch–and–bound algorithms for general mixed integer programming on the CM–5. *SIAM Journal on Optimization*, 4(4):794–814, 1994.

[19] J. Eckstein, C.A. Phillips, and W.E. Hart. PICO: An object-oriented framework for parallel branch and bound. In *Proceedings of the Workshop on Inherently Parallel Algorithms in Optimization and Feasibility and their Applications*, Studies in Computational Mathematics, pages 219–265. Elsevier Scientific, 2001.

[20] O.C. Ezechukwu and I. Maros. OOF: open optimization framework. Technical Report ISSN 1469-4174, Department of Computing, Imperial College of London, London, UK, 2003.

[21] R. Fletcher and S. Leyffer. Numerical experience with lower bounds for MIQP branch–and–bound. *SIAM Journal on Optimization*, 8(2):604–616, 1998.

[22] C. A. Floudas. *Nonlinear and Mixed–Integer Optimization*. Topics in Chemical Engineering. Oxford University Press, New York, 1995.

[23] R. Fourer. Modeling languages versus matrix generators for linear programming. *ACM Transactions on Mathematical Software*, 9:143–183, 1983.

[24] R. Fourer, L. Lopes, and K. Martin. LPFML: A W3C XML schema for linear and integer programming. *INFORMS Journal on Computing*, 17:139–158, 2005.

[25] R. Fourer, J. Ma, and K. Martin. OSiL: An instance language for optimization. Technical report, Northwestern University, January 2006.

[26] R. Fourer and D. Orban. DrAMPL: A meta solver for optimization. Technical report, IEMS Department, Northwestern University, Evanston, IL, January 2006.

[27] D.M. Gay. Hooking your solver to AMPL. Technical report, Bell Laboratories, Murray Hill, NJ, 1993. Revised 1994, 1997; available at `www.ampl.com/REFS/abstracts.html#hooking2`.

[28] J.-P. Goux and S. Leyffer. Solving large MINLPs on computational grids. *Optimization and Engineering*, 3:327–346, 2002.

[29] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3:227–252, 2002.

[30] I. E. Grossmann and Z. Kravanja. Mixed–integer nonlinear programming: A survey of algorithms and applications. In A.R. Conn L.T. Biegler, T.F. Coleman and F.N. Santosa, editors, *Large–Scale Optimization with Applications, Part II: Optimal Design and Control*, New York, Berlin, 1997. Springer.

[31] I. E. Grossmann and R. W. H. Sargent. Optimal design of multipurpose batch plants. *Ind. Engng. Chem. Process Des. Dev.*, 18:343–348, 1979.

[32] B.V. Halldórsson, E.S. Thorsteinsson, and B. Kristjánsson. A modeling interface to non-linear programming solvers an instance: xMPS, the extended MPS format. Technical report, Carnegie Mellon University and Maximal Software, 2000.

[33] IBM. Passing your model using mathematical programming system (MPS) format, 2003. `http://www-306.ibm.com/software/data/bi/osl/pubs/Library/featur11.htm`.

[34] G.R. Kocis and I.E. Grossmann. Global optimization of nonconvex mixed–integer nonlinear programming (MINLP) problems in process synthesis. *Industrial Engineering Chemistry Research*, 27:1407–1421, 1988.

[35] B. Kristjánsson. Optimization modeling in distributed applications: how new technologies such as XML and SOAP allow OR to provide web-based services, 2001. `http://www.maximal-usa.com/slides/Svna01Max/index.htm`.

[36] P.S. Laursen. Can parallel branch and bound without communication be effective? *SIAM J. Optimization*, 4(2):288–296, 1994.

[37] S. Leyffer. *User manual for MINLP*. University of Dundee, 1998.

[38] J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal of Computing*, 11:173–187, 1999.

[39] J.T. Linderoth and S.J. Wright. Implementing a decomposition algorithm for stochastic programming on a computational grid. *Computational Optimization and Applications*, 24:207–250, 2003.

[40] Lindo Systems, Inc. LINDO API user's manual. Technical report, Lindo Systems, Inc., 2002. `http://www.lindo.com/lindoapi_pdf.zip`.

[41] M. Livny, J. Basney, R. Raman, and T. Tannenbaum. Mechanisms for high-throughput computing. In *Speedup 11*, 1997. Available from `www.cs.wisc.edu/condor/doc/htc_mech.ps`.

[42] Leonardo B. Lopes. *Modeling Stochastic Optimization: From Idea to Instance*. PhD thesis, Northwestern University, 2004.

[43] Robin Lougee-Heimer. The Common Optimization INterface for operations research. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

[44] Jun Ma. *Optimization Services*. PhD thesis, Northwestern University, 2005. Available at `www.optimizationservices.org` or `gsbkip.chicagogsb.edu/os/thesis.pdf`.

[45] A. Martin, M. Möller, and S. Moritz. Mixed integer models for the stationary case of gas network optimization. *Mathematical Programming*, 105(2–3):563–2, 2006.

[46] National Science Foundation. An operations cyberinfrastructure: Using cyberinfrastructure and operations research to improve productivity in the enterprise. Workshop report, National Science Foundation, August 30-31, 2004. Available at `www.optimization-online.org/OCI/OCI.pdf`.

[47] National Science Foundation. Revolutionizing science and engineering through cyberinfrastructure. Report of the blue-ribbon advisory panel on cyberinfrastructure, National Science Foundation, January 2003. Available at `http://www.nsf.gov/od/oci/reports/atkins.pdf`.

[48] J. F. Pekny and D. L. Miller. A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems. *Mathematical Programming*, 55(1):17–33, 1992.

[49] A.J. Quist, R. van Geemert, J.E. Hoogenboom, T. Illés, E. de Klerk, C. Roos, and T. Terlaky. Optimization of a nuclear reactor core reload pattern using nonlinear optimization and search heuristics. Draft paper, Delft University of Technology, Faculty of Applied Mathematics, Department of Operation Research, Mekelweg 4, 2628 CD Delft, The Netherlands, September 1997.

[50] Aaron Skonnard and Martin Gudgin. *Essential XML Quick Reference*. Pearson Education, Inc, Boston, MA, 2002.

[51] J. A. Tomlin. A suggested extension of special ordered sets to non-separable non-convex programming problems. In P. Hansen, editor, *Studies in Graph Theory and Discrete Programming*, pages 359–370. North Holland Publishing Company, 1981.

[52] J. Viswanathan and I. E. Grossmann. Optimal feed location and number of trays for distillation columns with multiple feeds. *I&EC Research*, 32:2942–2949, 1993.

[53] T. Westerlund, J. Isakson, and I. Harjunkoski. Solving a production optimization problem in the paper industry. Report 95–146–A, Department of Chemical Engineering, Abo Akademi, Abo, Finland, 1995.