SANDIA REPORT

SAND2012-0529
Unlimited Release
Printed January 24, 2012

Optimization of Large-Scale Heterogeneous System-of-Systems Models

Genetha A. Gray, Willian E. Hart, Patricia D. Hough, Herbert K.H. Lee, Ojas D. Parekh, Cynthia A. Phillips, John D. Siirola, Laura P. Swiler, Jean-Paul Watson, David L. Woodruff

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2012-0529
Unlimited Release
Printed January 24, 2012

Optimization of Large-Scale Heterogeneous System-of-Systems Models

Ojas D. Parekh, Cynthia A. Phillips, John D. Siirola, and Jean-Paul Watson
Discrete Math & Complex Systems Department

William E. Hart
Data Analysis & Informatics Department

Laura P. Swiler
Optimization and Uncertainty Quantification Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1326

Genetha A. Gray and Patricia D. Hough
Quantitative Modeling and Analysis Department
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94451-9159

Herbert K.H. Lee
Applied Mathematics and Statistics
University of California, Santa Cruz
Basked School of Engineering
1156 High St, MS SOE2
Santa Cruz, CA 95064

David L. Woodruff
Graduate School of Management
University of California, Davis
Davis, CA 95616-8609

Abstract

Decision makers increasingly rely on large-scale computational models to simulate and analyze complex man-made systems. For example, computational models of national infrastructures are being used to inform government policy, assess economic and national security risks, evaluate infrastructure interdependencies, and plan for the growth and evolution of infrastructure capabilities. A major challenge for decision makers is the analysis of national-scale models that are composed of interacting systems: effective integration of system models is difficult, there are many parameters to analyze in these systems, and fundamental modeling uncertainties complicate analysis. This project is developing optimization methods to effectively represent and analyze large-scale heterogeneous system of systems (HSoS) models, which have emerged as a promising approach for describing such complex man-made systems. These optimization methods enable decision makers to predict future system behavior, manage system risk, assess tradeoffs between system criteria, and identify critical modeling uncertainties.

Acknowledgment

We thank the Laboratory Directed Research and Development Program at Sandia National Laboratories for funding this work. We with to thank management, specifically M. Daniel Rintoul, and the Sandia LDRD office for supporting this project and the project team.

Contents

1 Introduction

Decision makers increasingly rely on large-scale computational models of complex man-made systems to inform decisions. For example, computational models of national infrastructures are being used to inform government policy, assess economic and national security risks, evaluate infrastructure interdependencies, and plan for the growth and evolution of infrastructure capabilities. A feature common to these complex man-made systems is the interaction among related, yet conceptually distinct subsystems. While it is convenient to model each subsystem separately, the overall system behavior relies on the effective integration across all subsystems. One promising approach for describing such systems is as a "*Heterogeneous System-of-Systems*" (HSoS) model. These computational models typically have a large number of parameters with significant uncertainty and complicated interactions among the component subsystems.

While computational models of HSoS provide invaluable insight into the system behavior and allow the decision makers to run "what if" scenarios, supporting the decision-making process requires systematic *analysis* of the model. This analysis involves characterizing the performance of the system across the space of all possible conditions, states, or events. Typical analyses include identifying designs or operating conditions that best meet certain goals, understanding the range of solutions that represent the best trade-offs among competing goals, and quantifying and mitigating risk inherent in any decision. Each of these analysis activities can be expressed naturally as a form of optimization.

This report summarizes the results of a three-year Laboratory Directed Research and Development (LDRD) project at Sandia National Laboratories. The project focused on the development and application of numerical optimization methodologies as a unified approach for exploring the operational space and providing analysis of complex HSoS systems. This report will begin with a brief discussion of HSoS systems and the research space this project operated in. We will then go on to summarize the key research areas, activities, and outcomes: structured algebraic modeling, stochastic programming, hybrid optimization, and mixed discrete-continuous surrogates. We will close with a summary of key project metrics.

1.1 Heterogeneous System-of-Systems

Computational models used to simulate complex man-made systems are increasingly being described as "System of Systems" (SoS) models. Although many definitions of a SoS have been proposed, "most agree that a system of systems arises when a set of needs are met through a combination of several systems. Each system can operate independently, but each also must interact effectively with other systems to meet the specified needs" [5]. A feature of many SoS models is that they integrate a heterogeneous collection of constituent systems. Heterogeneous system of systems (HSoS) models can leverage system domain expertise in a modular fashion, so diverse aspects of man-made systems can be integrated into a single model (e.g., economics, climate, and human behavior). The constituent system models can

operate at different time scales and with different levels of resolution. Further, a HSoS model can integrate systems that are modeled with very different mathematical techniques, including systems dynamics, partial differential equations, and mathematical programs.

The classic example of a HSoS problem is aircraft design. An aircraft is a single complex engineered system; however, we commonly think of it as the intersection of numerous disciplines: aerodynamics, structural mechanics, combustion mechanics, communications and sensing, heating, ventilation, and air condition, electrical engineering, hydraulic engineering, etc. During aircraft design, it is convenient and common to treat each discipline separately: after all, the communications and radar systems do not depend on the details of the engine design - only that the engine can generate sufficient electricity to power the equipment. Other subsystems are more intimately coupled: the aerodynamic design may prefer the aircraft to be as light and slender as possible, yet the structural mechanics may need stronger, thicker wings to support loading. An added dimension is how the aircraft "fits" into an airline's existing fleet. In this case, critical parameters like range and capacity, which are usually taken as givens in the design process, now become variables. The overall fleet integration question becomes what new aircraft design (subsystem) will best enhance the operability or profitability of the airline route system.

An alternative view of the HSoS paradigm is managing the electric power grid. In the grid, there is a collection of independent generating companies, each with a fleet of generating units (power plants). Each generating unit has its own unique capabilities and operating parameters. Each generating company operates its fleet of generating units ostensibly to maximize its own net profitability. Opposite the generating companies are the consumers, ranging from individual households to large corporate or industrial sites. Linking them all together is the system operator, which commits individual generating units to produce or idle in order to meet the anticipated demands and transmission constraints. In this system, the decomposition into subsystems is along control boundaries instead of discipline boundaries.

After considering these and other applications considered "canonical" HSoS systems, we realized that the unifying characteristic among HSoS applications was not necessarily the *way* in which the systems were decomposed, but rather *that* the approach to modeling or analyzing the application *relied on decomposition in order to make the model or analysis tractable*. Further, there are many possible axes across which we could decompose the system. The two examples above highlight functional decomposition and control decomposition, which, along with spatial decomposition, are the most prevalent simulation decomposition approaches appearing in literature. However, it became readily apparent that there are numerous other decomposition strategies that could be employed when performing HSoS analysis. In particular, the systems could be decomposed in time, uncertainty space, and algorithmic space. This realization subtly shifted the focus of the project away from exploiting the characteristics of decompositions specific to individual applications and toward the more general identification and exploitation of the general "axis of decomposition" most appropriate to the desired *analysis*.

1.2 Optimization as a unifying approach

Optimization is a natural paradigm for analyzing HSoS models because it can readily be tailored to address many of a decision maker's analysis questions. This can range from classical optimization ("find the solution that yields the best value of a goal") to multi-criteria optimization ("quantify the best possible trade-off among multiple competing goals") to stochastic optimization ("identify the solution that best manages risk in some quantifiable manner"). In particular, several prevalent features in HSoS analysis applications motivate the optimization research in this project: (1) The core modeling components and the decisions facing the decision-maker are naturally discrete. (2) There is fundamental uncertainty in the data, which comes from a diverse range of sources, (3) HSoS models often describe how systems evolve over time, and (4) There are many criteria for assessing the performance of HSoS systems.

While the HSoS moniker is relatively new, models of complex systems that could easily be classified as HSoS models have long been used in many applications. However, optimization techniques are infrequently used to analyze these models. There are several characteristics of HSoS models that inhibit the effective application of optimization. First, these models are large and simulation alone can be computationally challenging; naively wrapping the simulation with an optimization algorithm quickly becomes computationally prohibitive. Second, the analysis of SoS models often requires a combination of both integer and continuous decision variables, and optimization has not been widely applied to general mixed integer-continuous applications. Third, there are no robust, scalable, general-purpose optimization packages that can directly handle many of the HSoS optimization analyses (in particular stochastic programming and multi-objective analysis).

This work sought to overcome these barriers and provide the foundational capabilities to support the direct analysis of HSoS models through the application of general-purpose optimization algorithms and approaches. A central focus of this project is a deliberate focus on incorporating both integer and continuous decisions into the optimization processes. To accomplish this, we focused on several key research ideas:

Structured algebraic modeling: Through this project, we explored new environments for expressing and manipulating structured algebraic models. The availability of an open, extensible, and manipulable algebraic representation of the HSoS model proved to be the key cornerstone upon which we could base our algorithmic research.

Multi-Stage Stochastic Optimization: Explicitly capturing and understanding how model uncertainties evolve with time is critical to generating any actionable analysis for HSoS systems. In this work, we sought to represent and manage uncertainty in the context of multi-stage stochastic optimization. We developed a general-purpose stochastic programming environment that provides a standard form for expressing multi-stage optimization problems. We then focused on developing scalable parallel stochastic programming solvers based on the Progressing Hedging decomposition algorithm [16].

Risk Management: Although expected value and worst-case risk analysis is commonly used to analyze man-made systems, it is well-known that these measures can yield undesirable solutions. The expected value discounts rare but high-consequence events, whereas the worst case results in solutions that tend to be excessively conservative. In this project, we adapted risk management measures from finance (notably the Conditional Value-at-Risk (CVaR)) and applied them within the context of multi-stage stochastic optimization of large-scale HSoS applications.

Infrastructures for Multi-Objective Optimization: Analysis of trade-offs between multiple criteria is well-known to be challenging for optimization, and few researchers have considered multi-objective optimization with uncertain objectives. One of the central challenges is a lack of robust, scalable multi-objective optimization algorithms. One alternative to relying on single algorithms is to use of numerous single- and multi-objective algorithms in a collaborative hybrid environment. Unfortunately, currently no optimization environments support the construction of such hybrid systems. In this project, we developed a new optimization infrastructure for expressing and constructing optimization processes.

Mixed discrete-continuous surrogates: A central challenge in developing computationally-tractable (and optimizable) models of HSoS problems is the transition from a nominally simulation-based model to an explicit algebraic model. Typically this is a manual process requiring the participation of both a domain expert and an optimization practitioner. Ideally, we would also like to be able to generate simplified (algebraic) surrogate models either automatically or with minimal supervision. The key challenge is developing surrogate methods for capturing the discrete components of a HSoS model.

2 Algebraic modeling

As we were forming the research thrusts of this LDRD, a central issue that we first had to address was *how we should represent the optimization problem*. The Discrete Mathematics group has a history of successes using AMPL [1, 6] to model and solve large-scale integer programs. Our previous application experience highlighted the value of Algebraic Modeling Languages (AML) for solving real-world optimization applications. These systems provide a structured environment for expressing optimization problems in a form that is readily amenable to optimization with state-of-the-art optimization algorithms. However, the strict syntax and closed nature of widely-used commercial AMLs did not provide a sufficient degree of structure, flexibility, and extensibility to support the algorithmic research that we intended to pursue in this project. In contrast, other open-source AML (and AML-like) environments (e.g., APLEpy [2, 10], CVXOPT [4], PuLP [13], PyMathProg [14], or OpenOpt [12]) lack the expressiveness and features that we felt would be necessary (notably, nonlinear modeling and remote computation).

Instead, we elected to base our work in this project on the prototype results of a 2007 Late-Start LDRD that explored modeling environments that could expose explicit algebra for programmatic interrogation and manipulation. Under the auspices of this project, the Python Optimization Modeling Objects (Pyomo) matured from a prototype modeling concept to a broadly-used full-featured AML. Pyomo is a collection of classes and services that support the direct formulation and manipulation of algebraic models within the Python programming environment.

Key features of Pyomo include:

- Embedded in a high-level, full-featured programming language
- Access to extensive third-party library functionality
- Abstract and Concrete modeling
- Support for linear and nonlinear problems
- Integrated support for distributed computation
- Cross-platform deployment capabilities
- Integrated support for obtaining data from numerous external sources
- Extensibility through component-based software architecture
- Advanced application scripting capabilities

For more detailed discussion of Pyomo, please see the manuscript [9]:

> W.E. Hart, J.P. Watson, and D.L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3), 2011.

and the forthcoming book [8]:

> W.E. Hart, C.D. Laird, J.P. Watson, and D.L. Woodruff. *Pyomo: Optimization Modeling in Python*. Springer Optimization and Its Applications. Springer, 2012. ISBN 978-1-4614-3225-8.

2.1 AML extensions

A central challenge in developing explicit algebraic models of Heterogeneous Systems-of-Systems is the discrepancy between the structured and often separable subsystems in an HSoS and the more uniform, regular modeling supported by traditional AMLs. While modeling constructs like sparse multidimensional sets can support constructing HSoS models directly, this process is arduous, time consuming, and error prone. Instead, we leveraged the extensible nature of Pyomo to develop new AML modeling constructs that can better and more intuitively capture the special structures common in HSoS models.

Block-oriented modeling

Classically, optimization problems fall into the following general form:

$$\begin{array}{lll} \min & f_i(x,y) & \forall\, i \in 1,\ldots,F \\ s.t. & g_j(x,y) \le 0 & \forall\, j \in 1,\ldots,G \\ & h_k(x,y) = 0 & \forall\, k \in 1,\ldots,H \\ & \{x \in \mathbb{R}^m \mid x^L \le x \le x^U\} & \\ & \{y \in \mathbb{Z}^n \mid y^L \le y \le y^U\} & \end{array} \tag{1}$$

While very general, this form removes or hides much of the structure that is present in the original problem. Instead of thinking about the model as a "flat" collection of equations and constraints, HSoS models are more intuitively modeled as a collection of *blocks* of equations and constraints coupled together by a high-level model:

$$\begin{array}{lll} \min & f_i(x,y) & \forall\, i \in 1,\ldots,F \\ s.t. & g_j(x,y) \le 0 & \forall\, j \in 1,\ldots,G \\ & h_k(x,y) = 0 & \forall\, k \in 1,\ldots,H \\ & \left[\begin{array}{ll} g_r(x,y,x_b,y_b) \le 0 & \forall\, r \in 1,\ldots,G_b \\ h_s(x,y,x_b,y_b) = 0 & \forall\, s \in 1,\ldots,H_b \\ [\,\cdots\,] & \forall\, t \in 1,\ldots,B_b \\ \{x_b \in \mathbb{R}^{m_b} \mid x_b^L \le x_b \le x_b^U\} & \\ \{y_b \in \mathbb{Z}^{n_b} \mid y_b^L \le y_b \le y_b^U\} & \end{array}\right] & \forall\, b \in 1,\ldots,B \\ & \{x \in \mathbb{R}^m \mid x^L \le x \le x^U\} & \\ & \{y \in \mathbb{Z}^n \mid y^L \le y \le y^U\} & \end{array} \tag{2}$$

In this framework, individual blocks correspond to distinct subsystems, allowing the subsystems to maintain their integrity within the model. Note that blocks, can contain sub-blocks, which can in turn contain sub-sub-blocks, et cetera. This hierarchical structure both explicitly captured and maintains the decomposable structure of an HSoS system and greatly simplifies the construction and reuse of (sub)system models. It also provides a significant degree of encapsulation (i.e. *variable scoping*), and directly associates variables with their corresponding constraints. From a modeling system design standpoint, it is also important to note that blocks are not a special sub-component of an optimization model. Rather, the optimization model itself is a special case of a block; that is, the optimization model is simply a block that also contains one or more objective functions.

Connecting blocks

A disadvantage of the general block-oriented model structure presented in Equation 2 is that the integrating constraints ($g_j(x, y)$ and $h_k(x, y)$) that bind the various subsystems together are specific to the actual subsystem models. This forces subsystem models to "promote" variables that would otherwise be local to that subsystem into the global model space. Consider the example of the electric power grid: the various components (generators, customers, system operators) are all interconnected through transmission lines that can be modeled logically as blocks. However, the coupling constraints necessary for "hooking up" the overall network are a function of the transmission model employed within the individual blocks: current and node angles for the DC approximation, and real flow, reactive flow, and node angles for the AC approximation. Ideally, to the extent possible, we could separate the connectivity of the overall HSoS model from the detailed modeling within the subsystem blocks.

To address this, we introduced a new modeling construct: the *connector*. A connector is logically a labeled "bag of named variables" that can be used within constraints as if it were a single variable. Individual subsystem blocks declare standard connectors for representing their interface to the other subsystems in the HSoS model, but populate the connectors with the variables that are specific to their internal model. The overall system integration (HSoS) model can then be recast into a series of (nominally equality) constraints coupling various block (subsystem) connectors together. When Pyomo generates the optimization model, it "expands" the constraints formed over connectors by duplicating the constraint for each variable within the connector, matching variables from multiple connectors based on their associated name.

Disjunctive programming

As we noted in section 1.2, discrete decisions naturally arise in HSoS models. These decisions frequently take the form of switching decisions that indicate the presence or absence of a component or capability in the model; for example, the unit is on or off, we build a new facility or not, a transmission line exists or it does not. This translates into a logical

(Boolean) variable that "turns on" or "turns off" a series of constraints. The key challenge is correctly implementing these switching decisions within the context of an AML. The standard approach is to relax the constraint(s) in question by adding a "Big-M" term (the binary switching variable multiplied by a suitably large constant, M) so that when the binary is false, the constraint can not become active. This is both tedious and potentially error-prone as each constraint must be systematically edited and an appropriate value of M calculated.

An alternative approach is to pose the switching decisions as disjunctions and then apply a transformation to convert the disjunctive program into a mathematical program that is solvable with a standard optimization algorithm [3]. To support this within Pyomo, we developed a Generalized Disjunctive Programming [15] extension based on Pyomo's block modeling concept. Here, *disjuncts* are specialized blocks that include a binary switching (or *indicator*) variable that dictates whether the block of constraints is active or not. We then provide standard, automated transformations for converting the disjunctive program into a mathematical program using either a Big-M [15] or Convex Hull [11] relaxation. This capability greatly simplifies the generation of complex HSoS models.

3 Stochastic Programming

A key aspect of HSoS analysis is to quantify how system uncertainties evolve over time and their resulting impact of that risk on the overall anticipated system behavior. However, although stochastic programming is a powerful tool for modeling decision-making under uncertainty, various impediments have historically prevented its widespread use. One factor involves the ability of non-specialists to easily express stochastic programming problems as extensions of their deterministic counterparts, which are typically formulated first. A second factor relates to the difficulty of solving stochastic programming models, particularly in the mixed-integer, non-linear, and/or multi-stage cases. Intricate, configurable, and parallel decomposition strategies are frequently required to achieve tractable run-times on large-scale problems.

We simultaneously address both of these factors through the PySP software package for formulating and solving large-scale stochastic programming problems in Python. To formulate a stochastic program in PySP, the user specifies both the deterministic base model (supporting linear, non-linear, and mixed-integer components) and the scenario tree model (defining the problem stages and the nature of uncertain parameters) in the Pyomo open-source algebraic modeling language. Given these two models, PySP provides two paths for solution of the corresponding stochastic program. The first alternative involves writing the extensive form and invoking a standard deterministic solver. For more complex stochastic programs, we provide an implementation of Rockafellar and Wets' Progressive Hedging algorithm [16]. Our particular focus is on the use of Progressive Hedging as an effective heuristic for obtaining approximate solutions to multi-stage stochastic programs. By leveraging the combination of a high-level programming language (Python) and the embedding of the base deterministic model in that language (Pyomo), we are able to provide completely generic and highly configurable solver implementations.

For more detailed discussion of PySP, see the manuscript [22]:

> J.-P. Watson, D. L. Woodruff, and W. E. Hart. PySP: Modeling and solving stochastic programs in Python. *Mathematical Programming Computation*, (*to appear*), 2012.

A central challenge to leveraging Progressive Hedging as a general-purpose heuristic for solving large-scale multi-stage stochastic optimization problems is that it was originally devised for problems possessing only continuous variables. Extending Progressive Hedging to multi-stage stochastic programs with integer variables leads to a variety of critical issues, especially in the context of very difficult or large-scale mixed-integer problems. Failure to address these issues properly results in either non-convergence of the heuristic or unacceptably long run-times. We investigated these issues and developed algorithmic innovations that have been integrated within the PySP framework.

For more detailed discussion, see the manuscript [21]:

J.-P. Watson and D. L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Comp.Mgmt.Sci*, 8(4):355–370, 2011.

Finally, a critical issue when applying stochastic programming for decision making under uncertainty is exactly *how* to quantify risk. A common approach observed in the literature is to optimize a function of the the *expected* (mean) value of the system performance. While optimizing the expected value is relatively straightforward to pose and solve, it ignores the variability in the system performance and can be adversely biased for systems exhibiting non-normally distributed outcomes. *Mean-variance* metrics address this by penalizing the expected value with the resulting variance. However, in our experience the key risk of interest to decision-makers is not deviation from the average system performance, but rather managing extreme events on one side of the average. That is, a one-sided tail-conditioned risk metric. To address this, we developed general-purpose extensions to PySP for expressing Conditional Value at Risk (CVaR) objectives. We also explored computational procedures for decomposing and effectively solving problems with chance constraints [20].

For more detailed discussion, see the manuscript [20]:

J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff. Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing*, 22(4):543–554, 2010.

4 Hybrid Multi-objective Optimization

Analysis of trade-offs among multiple criteria is well-known to be challenging for optimization, and few researchers have considered multi-objective optimization with uncertain objectives. One of the central challenges is a lack of robust, scalable multi-objective optimization algorithms. Previous research [18] suggests that good approximations of the multi-objective surface can be obtained through the use of numerous single- and multi-objective algorithms in a collaborative hybrid environment. Unfortunately, currently no optimization environments support the construction of such hybrid systems. In this project, we developed a new conceptual optimization framework for expressing and constructing hybrid optimization processes as part of the Common Optimization Library INterface (COLIN) library.

4.1 Conceptual Optimization Frameworks

The central challenge in developing a general-purpose hybrid environment is identifying a suitable optimization framework for coupling the various algorithms to the underlying optimization model. Ideally, optimization frameworks exist to provide standardized interfaces that simplify the construction of complex optimization applications. They provide standardized access to common functionality, especially interfaces for evaluating the optimization model and then storing and retrieving the subsequent results. By defining standardized access points, optimization frameworks facilitate the integration and reuse of modeling, algorithmic, and infrastructure components.

Unfortunately, optimization frameworks also rely on a rigid application programming interface (API). This API dictates both the concepts that the framework aims to support (e.g., organization, functions, methods) as well as the semantics for how we interact with the concepts (i.e. parameters and data types). This places a certain burden on consumers of the framework to "wrap" their algorithms, models, and components to fit the framework's API. In particular, this requires numerous conversions of data to and from the form required by the API. The real limitation is that the components become constrained to the framework's data types: a framework that uses a search domain of $(\mathbb{R}^m, \mathbb{Z}^n)$ (by far the most prevalent framework domain) explicitly excludes applications that use other data representations such as complex numbers, sequence pairs, and graph expressions. The reverse is also true: in an attempt to provide a general framework, the API often offers a superset of the functionality that a specific component can use (e.g., a linear programming solver cannot handle discrete variables). This requires each framework component to verify that it is being called with a compatible subset of the API. This also means that augmenting the framework API to include new features (e.g., an extended modeling domain) requires modifying every client component, at a minimum to augment the API verification to include validity checks for the extended API.

Instead of a full, rigid API, developing optimization applications only requires a "conceptual framework." The conceptual framework specifies overall organization, core components

and services, and methods; the actual data type used to interact with framework is, for the most part, irrelevant. Put a different way, many fundamental operations within an optimization algorithm rely on conceptual services like "perform a function evaluation," "evaluate constraint violation," or "store this solution." The actual domain data type passed to the optimization model or the representation of the solution stored in a cache or database is a contract between the optimization algorithm and the model or cache and should not involve the framework.

4.2 Concrete Variant Type Systems

Implementing a conceptual optimization framework in a strongly-typed language requires a complete infrastructure for storing and manipulating *variant* data; that is, concrete variables that may contain arbitrary data. For this project, we elected to build the variant type infrastructure on a derivative of the Boost[1] "Any" class. The Boost Any class supports a type-safe mechanism for storing and retrieving arbitrary data within a single concrete type. Our extensions to the standard Boost Any provide the option to store by value or reference, convert the Any type into a reference-counted object to improve performance and simplify memory management, and provide implicit coercion of arbitrary data into an Any. This allows us to implement an efficient conceptual optimization API by defining general interface methods that take and return Any objects.

The main challenge with working with variant data is how to retrieve data *from* the variant object. For the Any class (and indeed for any variant in a strict type-safe language), the consumer of the data must anticipate the data type stored in the variant *before* attempting to retrieve the data. This fundamentally limits the utility of the Any class, as all clients must check for (and convert) all possible incoming data types. To address this, we developed a general Any-based type management system. The Type Manager contains a registry of known conversions from one data type to another in the form of a *cast graph*. When a client wishes to retrieve data from an Any, the client passes the Type Manager the source Any and the destination data type. The Type Manager then identifies a feasible path through the cast graph and applies the necessary conversions and returns the data to the client as the requested type.

To support storage, retrieval, and transmission of Any objects we developed a serialization system based on the registry concepts in the Type Manager. The Serialization Manager contains two registries: the first is the database of serialization/deserialization functions and the second contains registered constructors for generating new instances of serializable types. Combined, these registries allow the Serialization Manager to retrieve arbitrary data from a serialized stream directly into Any objects without the need to anticipate the type of the next object in the stream.

[1]Boost C++ Library: `http://www.boost.org/`

4.3 Problem Transformations

The two central challenges in developing hybrid optimization algorithms are to ensure that the target optimization model is compatible with the individual optimization solvers and to facilitate the transmission of results from one solver to another. Our approach in COLIN is to declare concrete types (through C++ templates) for each fundamental classification of optimization problem (e.g., Linear Program, Nonlinear Program without derivatives, Nonlinear Program with first derivatives, Mixed Integer Program, etc.). Reformulations of one problem type to another can then be considered "type casts". For example, to convert a general Nonlinear Program (NLP) to an Unconstrained Nonlinear Program (UNLP), you would cast the NLP into an UNLP by applying a Penalty Function Reformulation, which wraps the original NLP problem within a UNLP problem that maps the constraint residuals from the original problem into a penalized objective in the new problem. By registering these default problem transformations with the Type Manager, we can support the automatic mapping of raw optimization models into reformulated models that are specific to and appropriate for each optimization algorithm in the hybrid optimization algorithm.

To facilitate the transmission of results from one solver (operating on one reformulation) to another (potentially operating on a different reformulation), we implement ideas from Polymorphic Optimization [17]. Each reformulation supports 3 data transformations (or *maps*). Mapping a reformulated search domain to the base search domain and mapping the base result to the reformulated result support the general optimization information flow. Reading results from another solver context requires a third approximate mapping of the base search domain into the reformulated search domain.

4.4 Solution Management

The final critical component for multi-objective hybrid optimization environments is a system for managing collections of candidate solutions to the problem. To provide this capability, we developed a unique multi-model solution caching system. This system implements an annotated database of solutions stored in the original model context. Any reformulated problem context can query the database, and through the reformulation mapping capability, automatically receives the data in the appropriate reformulated form. In addition, the cache supports constructing *views* of the data within the database. A view is a "window" into a subset of the data in the cache. Through the use of event callbacks, views automatically maintain consistency with the underlying cache. As views implement the full cache interface, they can be treated as standalone caches and nested arbitrarily. This capability has found widespread use within COLIN. Algorithms receive initial starting points and return results through subset views, allowing for seamless integration of both single-point algorithms (e.g., pattern search) and population-based algorithms (e.g., genetic algorithms). For multi-objective optimization, we maintain the current set of non-dominated solutions through a "Pareto view". By defining multiple nested Pareto view instances, we can automatically track both the complete high-dimensional non-dominated set as well as lower-dimensional

projections. Indeed, automatically tracking the best solution identified in single-objective optimization is simply the special case of a Pareto view over a single objective.

4.5 Solver Implementation

We implemented `coliny`, a general-purpose hybrid solver environment based on the COLIN optimization library. This solver combines interfaces to optimization solvers developed within the Acro project[2] with a general-purpose XML-based language for configuring and executing optimization work flows. This provides users with a flexible environment for specifying and executing arbitrary hybrid optimization algorithms. We demonstrated the utility of this system through the design and multi-objective analysis of sensor placement subsystems for water distribution systems [7].

[2]Acro: *A Common Repository for Optimizers*: `https://software.sandia.gov/acro`

5 Mixed Discrete-Continuous Surrogate Models

Much of the success or failure of numerical optimization stems from the ability of the optimization algorithm to identify and exploit special structure in the target problem. To that effect, many optimization environments require problems to be formulated as algebraic models to facilitate the detection of key characteristics (e.g., linearity, convexity, and gradient/Jacobian information). However, many HSoS models are implemented as a combination of multiple interacting computationally expensive simulations. In this case, algebraic representations are simply unavailable. Further, the computational expense of the constituent simulations may preclude directly embedding the simulation within the optimization algorithm. One common approach for optimizing "expensive" simulation-based models is to construct an approximate (algebraic) surrogate model of the simulation response surface based on a limited number of simulations, and then optimizing the surrogate model.

A significant challenge in applying surrogate optimization techniques to HSoS models is the aforementioned prevalence of discrete decisions. Typically, in surrogate models constructed over continuous variables, there is the assumption of continuity: as a continuous variable varies by a small amount, the response is assumed to vary smoothly. This is not always the case, and there are surrogate methods that can handle discontinuities in responses, but most surrogates (e.g. polynomial regression, splines, Gaussian process models, etc.) rely on assumptions of continuity.

In this project, we investigated the utility of several approaches for constructing mixed discrete-continuous surrogate models, focusing on the Adaptive COmponent Selection and Smoothing Operator (ACOSSO), Gaussian Processes with special correlation functions, treed Gaussian Processes, and categorical regression. For more detailed discussion, see the technical report [19]:

> L. P. Swiler, P. D. Hough, P. Qian, X. Xu, C. Storlie, and H. Lee. Surrogate models for mixed discrete-continuous variables. SAND 2012-0491, Sandia National Laboratories, 2012.

6 Project Summary and Outcomes

The following is a brief summary of the technical and programmatic achievements over the course of this three-year project.

6.1 Manuscripts, Reports, Presentations, and Software

This project directly supported the following manuscripts and reports:

W. E. Hart and J. D. Siirola. The PyUtilib Component Architecture. SAND 2010-2516 J, Sandia National Laboratories, 2010. *Submitted to The Python Papers.*

W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3:219–260, 2011.

J. D. Siirola. Current trends in parallel computation and the implications for modeling and optimization. In *Proc. 10th International Symposium on Process Systems Engineering.* 2009.

L. P. Swiler, P. D. Hough, P. Qian, X. Xu, C. Storlie, and H. Lee. Surrogate models for mixed discrete-continuous variables. SAND 2012-0491, Sandia National Laboratories, 2012.

J.-P. Watson, W. E. Hart, D. L. Woodruff, and R. Murray. Formulating and Analyzing Multi-Stage Sensor Placement Problems. In *Proc. Water Distribution System Analysis Conference 2010*, 2010.

J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff. Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing*, 22(4):543–554, 2010.

J.-P. Watson and D. L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Comp.Mgmt.Sci*, 8(4):355–370, 2011.

The project supported 37 presentations (including one keynote address) by team members at numerous national and international conferences and workshops, including:

- IMA Workshop on Mixed-Integer Nonlinear Programming

- 2009 INFORMS Computing Society
- 2009 Conference on Engineering Risk Control and Optimization
- 2009 INFORMS Western Regional Conference
- 2009 INFORMS Annual Meeting
- 10th International Symposium on Process Systems Engineering (PSE'09)
- 2010 ALIO-INFORMS Joint International Meeting
- 2010 Water Distribution Systems Analysis (WDSA) Conference
- 2010 INFORMS Practice Conference
- 2010 ICiS Optimization in Energy Systems Workshop
- 2010 AIChE Annual Meeting
- 2010 INFORMS Annual Meeting
- 12th International Conference on Stochastic Programming
- 19th Triennial Conference of the International Federation of Operational Research Societies (IFORS2011)
- 2011 SIAM Conference on Computational Science and Engineering
- 2011 SIAM Conference on Optimization
- 2011 Constraint Programming and Decision Making Workshop
- 2011 Annual Conference of the Production and Operations Management Society
- 2011 World Environmental & Water Resources Congress
- 2011 INFORMS Computing Society Conference

This project supported numerous software releases, including:

- PyUtilib 3.0-3.6
- Coopr 2.3-3.0
- Acro/Coliny 3.0-3.1
- UTILIB 4.1-4.2
- FAST 2.1-2.6
- PageMarkup 0.1-0.3

- TicketModerator 0.2-0.6.2.

There have been over 12,000 unique downloads and checkouts of software packages developed and supported by this project.

6.2 Programmatic Impact

The research supported by this project has gone on to form the basis of several follow-on projects:

- Research into stochastic programming and the PySP package provided the foundational basis for Optimization of Complex Systems (2010, ASCR) and Electric Grid Security (2011, LDRD). In turn, these projects have led to additional research into management of high-penetration solar generation (CRADA) and advanced grid management (ARPA-E).
- Research into structured algebraic modeling and hybrid optimization algorithms was a cornerstone of a renewed inter-agency agreement with the Environmental Protection Agency.
- Research in hybrid environments and integrated surrogate / optimization approaches led to research on surrogate-based co-optimization for uncertainty quantification (2012, Advanced Simulation & Computing).

Technology and tools developed through this project have been integrated into numerous software projects, including the COIN-OR project[3], Acro[4], DAKOTA[5], Coopr[6], DGM, TEVA-SPOT[7], and the Water Security Toolkit[8].

[3] http://www.coin-or.org
[4] http://software.sandia.gov/acro
[5] http://dakota.sandia.gov
[6] https://software.sandia.gov/coopr
[7] http://software.sandia.gov/trac/spot
[8] http://software.sandia.gov/trac/wst

References

[1] AMPL home page. `http://www.ampl.com/`, 2008.

[2] APLEpy: An open source algebraic programming language extension for Python. `http://aplepy.sourceforge.net/`, 2005.

[3] E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic and Discrete Methods*, 6(3):466–486, 1985.

[4] CVXOPT home page. `http://abel.ee.ucla.edu/cvxopt`, 2008.

[5] D. A. DeLaurentis and W. A. Crossley. A taxonomy-based perspective for systems of systems design methods. In *IEEE Systems, Man, & Cybernetics Conference*, volume 1, pages 86–91, 10-12 Oct. 2005. IEEE Paper No. 0-7803-9298-1/05.

[6] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming, 2nd Ed.* Brooks/Cole–Thomson Learning, Pacific Grove, CA, 2003.

[7] W. E. Hart, C. D Laird, R. Murray, C. A. Phillips, and J. D. Siirola. Evaluating multiobjective sensor placements. 2011 World Environmental & Water Resources Conference, Palm Springs, California, May 2011.

[8] W. E. Hart, C. D. Laird, J.-P. Watson, and D. L. Woodruff. *Pyomo: Optimization Modeling in Python*. Springer Optimization and Its Applications. Springer, 2012. ISBN 978-1-4614-3225-8.

[9] W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3:219–260, 2011.

[10] Suleyman Karabuk and F. Hank Grant. A common medium for programming operations-research models. *IEEE Software*, pages 39–47, 2007.

[11] S. Lee and I. E. Grossmann. New algorithms for nonlinear generalized disjunctive programming. *Computers & Chemical Engineering*, 24(9-10):2125–2141, 2000.

[12] OpenOpt home page. `http://scipy.org/scipy/scikits/wiki/OpenOpt`, 2008.

[13] PuLP: A Python linear programming modeler. `http://130.216.209.237/engsci392/pulp/FrontPage`, 2008.

[14] PyMathProg home page. `http://pymprog.sourceforge.net/`, 2009.

[15] R. Raman and I. E. Grossmann. Modelling and computational techniques for logic based integer programming. *Comp.Chem.Engng*, 18(7):563–578, 1994.

[16] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991.

[17] J. D. Siirola and Hauan S. Polymorphic optimization. *Computers & Chemical Engineering*, 31(10):1312–1325, 2007.

[18] J. D. Siirola, Hauan S., and A. W. Westerberg. Computing Pareto fronts using distributed agents. *Computers & Chemical Engineering*, 29(1):113–126, 2004.

[19] L. P. Swiler, P. D. Hough, P. Qian, X. Xu, C. Storlie, and H. Lee. Surrogate models for mixed discrete-continuous variables. SAND 2012-0491, Sandia National Laboratories, 2012.

[20] J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff. Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing*, 22(4):543–554, 2010.

[21] J.-P. Watson and D. L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Comp.Mgmt.Sci*, 8(4):355–370, 2011.

[22] J.-P. Watson, D. L. Woodruff, and W. E. Hart. PySP: Modeling and solving stochastic programs in Python. *Mathematical Programming Computation*, (*to appear*), 2012.

DISTRIBUTION:

1	MS 0899	RIM-Reports Management, 9532 (electronic copy)
1	MS 0359	D. Chavez, LDRD Office, 1911

Sandia National Laboratories