Computers and Chemical Engineering 57 (2013) 10–23

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Block-oriented modeling of superstructure optimization problems

Zev Friedman [1], Jack Ingalls [2], John D. Siirola *, Jean-Paul Watson

Sandia National Laboratories, Discrete Math and Complex Systems Department, P.O. Box 5800, MS 1326, Albuquerque, NM 87185-1326, USA

ARTICLE INFO

Article history:
Received 3 October 2012
Received in revised form 1 April 2013
Accepted 3 April 2013
Available online 17 April 2013

Keywords:
Superstructure optimization
Generalized disjunctive programming
Stochastic programming
Algebraic modeling language

ABSTRACT

We present a novel software framework for modeling large-scale engineered systems as mathematical optimization problems. A key motivating feature in such systems is their hierarchical, highly structured topology. Existing mathematical optimization modeling environments do not facilitate the natural expression and manipulation of hierarchically structured systems. Rather, the modeler is forced to "flatten" the system description, hiding structure that may be exploited by solvers, and obfuscating the system that the modeling environment is attempting to represent. To correct this deficiency, we propose a Python-based "block-oriented" modeling approach for representing the discrete components within the system. Our approach is an extension of the Pyomo library for specifying mathematical optimization problems. Through the use of a modeling components library, the block-oriented approach facilitates a clean separation of system superstructure from the details of individual components. This approach also naturally lends itself to expressing design and operational decisions as disjunctive expressions over the component blocks. By expressing a mathematical optimization problem in a block-oriented manner, inherent structure (e.g., multiple scenarios) is preserved for potential exploitation by solvers. In particular, we show that block-structured mathematical optimization problems can be straightforwardly manipulated by decomposition-based multi-scenario algorithmic strategies, specifically in the context of the PySP stochastic programming library. We illustrate our block-oriented modeling approach using a case study drawn from the electricity grid operations domain: unit commitment with transmission switching and $N-1$ reliability constraints. Finally, we demonstrate that the overhead associated with block-oriented modeling only minimally increases model instantiation times, and need not adversely impact solver behavior.

1. Introduction

Process design and expansion planning are cornerstones of the Process Systems Engineering literature. While there are many challenges involved in design and planning, a central issue is managing discrete choices throughout the process: which chemical pathway to exploit, how many trays in a distillation column, how to arrange columns in a separation train, where to build a plant, where to site a distribution system, etc. In each of these examples, one approach to the problem is to systematically construct a mathematical "superstructure" describing all possible discrete alternatives, and then rely on a numerical optimization algorithm to determine a single best realization (Grossmann, 1996). While conceptually straightforward, actually implementing such superstructures as mathematical optimization problems (i.e., programs) is an arduous task that requires careful bookkeeping. In particular, existing modeling languages for specifying mathematical programs generally lack the capability to specify superstructure optimization problems in a hierarchical manner that mirrors the physical system structure. Instead, the modeler must generate a "flat" mathematical description of the superstructure, which both significantly complicates the modeling effort (specifically with respect to validation and readability) and masks structure that may ultimately be used by a solver, if properly exposed. Further complicating the design problem is that the decision-maker must make these discrete decisions in the face of uncertainty (e.g., in material prices, demands, and construction times). Although systematic approaches exist for expanding – via Monte Carlo sampling – a deterministic model into a stochastic model, the process is rather tedious and typically implemented by hand as a "one-off" activity.

In this paper, we present a unified, systematic software infrastructure for modeling superstructure-based optimization problems based on the Pyomo[3] open source optimization modeling environment (Hart, Laird, Watson, & Woodruff, 2012; Hart,

* Corresponding author. Tel.: +1 505 284 5419; fax: +1 505 844 4728.
E-mail addresses: jdsiiro@sandia.gov (J.D. Siirola), jwatson@sandia.gov (J.-P. Watson).
[1] Current address: Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI 53706-1685, USA.
[2] Current address: Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA.

[3] Pyomo: *Python Optimization Modeling Objects:* https://software.sandia.gov/pyomo.

0098-1354/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compchemeng.2013.04.008

Watson, & Woodruff, 2011). Pyomo provides native Python support for expressing structured algebraic optimization models. Pyomo additionally includes extensions to support the representation of generalized disjunctive programs (GDPs) (Raman & Grossmann, 1994), and includes automated implementations of the standard transformations of GDPs into mixed-integer programs via both Big-M and Convex Hull relaxations (Lee & Grossmann, 2000). Borrowing from capabilities found in simulation modeling environments, Pyomo provides "block-oriented" algebraic modeling facilities that allow the modeler to explicitly represent a hierarchical superstructure and connections between sub-components independent of the actual variables and equations employed within the individual sub-components. This separation greatly simplifies model management, allowing the modeler to easily substitute different component representations or propose new superstructures. Similarly, block-oriented modeling facilities component re-use and simplifies the process of model validation. Further, the open nature of the Pyomo modeling environment facilitates the construction of custom optimization algorithms that can directly leverage the block structure of a superstructure optimization model. For example, we address the issue of solving multi-scenario mathematical programs via the PySP[4] stochastic programming package (Watson, Woodruff, & Hart, 2012). PySP is built on the Pyomo library and can automatically generate the extensive form of a multi-scenario program given a deterministic Pyomo model and a characterization of parameter uncertainty, expressed as a discretized scenario tree. Additionally, PySP provides a general implementation of the Rockafellar and Wets (1991) progressive hedging scenario-based decomposition algorithm, including extensions to support the solution of problems with discrete decision variables (Watson & Woodruff, 2011). Both Pyomo and PySP are distributed and installed within the larger Coopr library for optimization[5], co-developed by Sandia National Laboratories, the University of California Davis, and Texas A&M University.

We demonstrate our modeling system through a daily operations problem for an electricity distribution system: unit commitment with transmission line switching, subject to $N-1$ reliability constraints. In this example, we explicitly model alternative decisions via disjunctive programming constructs. Both generating technologies and transmission segments are modeled as discrete blocks, completely separating the description of the distribution superstructure from that of the individual network components. This separation provides for a more understandable and abstracted description of the superstructure optimization problem, and ultimately allows us to rapidly explore alternative transmission approximations (e.g., DC or AC power flow, with or without line loss) and different generation technologies (e.g., new carbon sequestration and/or renewable generation technologies). Our approach also facilitates the rapid expression of the superstructure under different structural simplifications (e.g., single bus, aggregated buses, or full network model).

The remainder of this paper is organized as follows. We begin by introducing our block-oriented modeling approach in Section 2. The balance of the paper is organized around our case study problem, that of unit commitment with transmission switching, subject to $N-1$ reliability constraints. This problem is introduced in Section 3, specifically by considering the traditional "flattened" mathematical programming formulation. We provide an alternative, block-oriented description of the basic formulation in Section 4; there, both the block syntax and basic GDP constructs are illustrated. We present computational results of our framework in Section 5, focusing on both the need for significant preprocessing capabilities when dealing with hierarchically structured optimization models and the overhead associated with such capabilities. Finally, we conclude in Section 6 with a summary of our results, and steps toward future research.

2. Motivation and modeling approach

A significant challenge in constructing algebraic optimization models for engineering decision support is the dichotomy between the graph-like and highly structured representations used to describe the engineering problem and the generally unstructured or "flat" algebraic form required by optimization modeling environments and solvers, e.g., including AMPL (Fourer, Gay, & Kernighan, 1990, 2002) and GAMS (Brooke, Kendrick, & Meeraus, 1988; Bussieck, Meeraus, & Kallrath, 2003). In the case of the former, capabilities for expressing hierarchical structure in both the models and the generated matrix forms for input to solvers are notably absent. To address this challenge, we propose to formulate superstructure optimization models using the "equation block" language constructs newly available in the Pyomo mathematical programming library. A Pyomo equation block is a collection of modeling components (e.g., sets, parameters, variable, constraints, and sub-blocks). Pyomo equation blocks have many features in common with other block-oriented optimization modeling environments, notably adopting key concepts from JModelica.org (Åkesson, Årzén, Gäfvert, Bergdahl, & Tummescheit, 2010), SML (Colombo, Grothey, Hogg, Woodsend, & Gondzio, 2009), and ASCEND (Piela, 1989). However, Pyomo is unique in its ability to support expressing, composing, and manipulating generic equation blocks within an algebraic modeling language for mathematical programming.

An equation block (or simply a *block*) represents a single component in a superstructure system and consists of the sets, parameters, variables, and constraints that describe the behavior of that component. This concept is analogous to the *model* construct in Modelica (Modelica, 2012) and ASCEND, and the *block* construct in SML. Individual Pyomo blocks then form the "nodes" in the graph-like engineering representation of the superstructure. A key focus of Pyomo blocks that is not present in structured approaches like SML is support for abstract composition of block-oriented models. To facilitate connecting a block (component) to other blocks in a superstructure model, each block declares a series of *connectors* that represent its interface to the rest of the model. A Pyomo connector is very similar to the connector concept in Modelica: it is a named collection of one or more objects that can be referenced and manipulated as a distinct entity. Arcs in the superstructure representation are implemented simply by "connecting" connectors together. However, unlike Modelica, which connects connector pairs using an explicit `connect(a,b)` function, connections in Pyomo can be any relational expression involving one or more connectors. This simple mechanism provides a powerful separation of concerns: the definition of the system superstructure relies only on the list of blocks and connectors, and is independent of both the component constraints and the variables that are associated with the connectors. This separation is analogous to the concept of *encapsulation* in object-oriented programming: an object contains internal ("private") data and methods, and only communicates with other objects through well-defined interfaces. In the case of block-oriented modeling, a block's sets, parameters, variables, and constraints are encapsulated within the block, and the connectors define the interface of that block with the rest of the model. While Pyomo blocks do not enforce a formal notion of "private variables", we have found that preserving encapsulation

[4] PySP: *Python Stochastic Programming:* https://software.sandia.gov/trac/coopr/wiki/PySP.

[5] Coopr: A Common Optimization Python Repository: http://software.sandia.gov/trac/coopr.