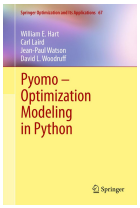

Pyomo – Optimization Modeling in Python, by W. E. Hart, C. Laird, J. P. Watson, and D. L. Woodruff, Springer, 2012. See http://www.springer.com. ▷

Reviewed by: Nedialko B. Dimitrov, Operations Research Department, Naval Postgraduate School, ned@nps.edu.

If a simple, intuitive tool for a task exists, the task is done more often, by more people. This basic principle is as true for gardening and gadgets, as it is for computation in operations research. The book, *Pyomo – Optimization Modeling in Python*, documents a simple, yet versatile tool for modeling and solving optimization problems.

Pyomo, which stands for Python Optimization Modeling Objects, is an *algebraic modeling language* (AML) developed by a diverse set of researchers and hosted at Sandia National Laboratories. An AML allows a user to specify the algebraic form of an optimization problem, independent of its solution algorithm. Further, separating algebraic problem specification and data allows a user to solve varied instances of the same problem with relatively little overhead.

To solve an optimization problem, a researcher has two basic options: to use a high level modeling language that is separate from, but works in conjunction with, a general purpose solver, or to interface with a solver, possibly custom-made, directly through solver-specific libraries. The advantages of a modeling language are ease of problem specification, and the ability to switch solvers with no overhead. The advantage of a direct interface with a solver is the configurability of the optimization algorithm. Pyomo stands apart because it combines these two advantages.

The core difference between Pyomo and other AMLs is its development. Typically, languages like AMPL and GAMS are created with special-purpose syntax for defining optimization problems. This leads to relative ease for defining standard optimization problems, but invariably, for complex problems, a user requires complex logic for data pre-processing, data post-processing, and even solving. Existing AMLs add some standard programming language constructs to allow for these tasks, but using them is cumbersome to a user. Pyomo, on the other hand, starts with a popular, general purpose, modern programming language—Python—and adds AML constructs. In doing so, the complex pre-processing, post-processing, and solving logic can be easily implemented by a user while still maintaining the mathematical specification and solver independence of an AML. The idea of building an AML on top of a standard programming language is appealing and other projects also follow that model: CVX, CVXPY, PuLP-OR.

The book, by Bill Hart, Carl Laird, Jean-Paul Watson, and David Woodruff, is essential to the usability of Pyomo, serving as *the* Pyomo documentation. The main alternative to learning the power and flexibility of Pyomo is to read example code, or dive into the open-source software's internals. Because Pyomo is only a few years old and under active development, it does not yet

have the large documentation base of comparable AMLs. *Pyomo – Optimization Modeling in Python* is about 230 pages and the first book on Pyomo, contrasting with the thousands of pages and multiple books—some on their 23rd version—for a mature AML like GAMS.

The book has contents for both an inexperienced user, and a computational operations research expert. It begins with a simple introduction to Pyomo design-philosophy and Coopr, the ecosystem of software packages within which Pyomo resides. The introduction describes key choices made by the Pyomo developers: for example, the motivation for creating Pyomo in the first place, the decision to make Pyomo open source, and the naming and semantic distinctions employed in Pyomo. The book continues to describe in detail each building block Pyomo makes available to a user for creating an optimization model. The book abounds with examples of each of the concepts discussed.

Chapters one through seven of the book are meant to be a comprehensive reference, for both novice and experienced users. To facilitate the text's use as a reference, the authors create multiple methods of accessing the book's contents. For example, an impatient reader can simply read the examples and edit them to create his own optimization problem, referencing to the book's text only when needed. An experienced reader can read the text front to back, learning about features of Pyomo they may not have previously used. The authors increase the text's usability as a reference by incorporating things like abstracts for each chapter, and tables listing each Pyomo building block's attributes.

The book also contains content clearly targeted at experts, especially in the later chapters. For example, the book describes how Pyomo can extract optimization problem data directly from an SQL database. Furthermore, the book features advanced optimization topic chapters: on Nonlinear Programming, complete with a chemical reactor design example; on Stochastic Optimization, complete with a 2-stage, 3-scenario example and discussion of using special purpose algorithms like Progressive Hedging; and on general purpose solver scripting, complete with a critical Bender's decomposition example. A novice user would likely be able to implement and run the examples in these chapters, but a detailed description of the mathematics behind the examples is beyond the scope of the book and would only be understood by an expert.

Overall, the book successfully targets multiple audiences. For novices, it provides a step-by-step introduction to the basic semantics of an AML, and even a short Python tutorial as an appendix. For intermediate users, it provides a detailed description of Pyomo features. For advanced users, it provides examples of expert features for custom algorithms like Progressive Hedging and Bender's decomposition. Because of its broad goals, the book is best read as a reference, with the reader skimming over sections not immediately useful to them.

The book's true value, however, comes in increasing Pyomo's usability. Pyomo is promising because it makes optimization modeling easier than it has been before—by merging a standard, powerful programming language with an AML. Prior to this book, documentation on Pyomo was scarce; only a motivated user could spend the effort necessary to understand Pyomo's structure to solve their optimization problem. While the book is essential for understanding Pyomo, the authors clearly recognize this documentation is an initial and ongoing effort. Throughout the book, they are careful to point out that Pyomo is an evolving project, with a design that is mature in some areas but changing in others. The authors also diligently point out Pyomo's drawbacks, such as inefficiency for a careless user and verbosity in model specification. But all of those drawbacks are things that can ultimately be addressed in the future, by a large and motivated team of open-source developers.

Pyomo – Optimization Modeling in Python and the Pyomo software enable the practical computation, pre-processing, and post-processing of optimization models. Easy to use, accessible tools, like a well-documented Pyomo, can increase the applicability of Operations Research—so that non-experts can create and solve optimization models. An open-source, documented, functional Pyomo enables a person without a multi-thousand dollar license for an AML, without

knowledge of the details of optimization algorithms, without explicit training in Operations Research, to solve an optimization problem of interest. Doing so brings computational Operations Research to a mass market, and in turn brings the benefits of masses of programmers and users to computational Operations Research.