PySP: A Software Environment for Leveraging and Exploiting Deterministic Metaheuristics in the Solution of Stochastic Programs

Jean-Paul Watson[1], David L. Woodruff[2]

[1] Discrete Math and Complex Systems Department
Sandia National Laboratories
Albuquerque, NM USA
jwatson@sandia.gov

[2] Graduate School of Management
University of California Davis
Davis, CA USA
dlwoodruff@ucdavis.edu

Abstract

The vast majority of metaheuristics research focuses on the development of efficient approximation algorithms for difficult NP-hard *deterministic* optimization problems. However, uncertainty is pervasive in real-world problems, practically limiting the real-world impact of metaheuristics research. Thus, we strongly argue for the need to transition from deterministic to stochastic metaheuristics research. In the course of this transition, it is critical that we leverage the extensive research performed on deterministic counterparts of stochastic optimization problems, as the latter pose an even greater computational challenge than the former. To facilitate this transition, we introduce a Python-based programming environment – PySP – to facilitate the rapid development and prototyping of stochastic optimization models and metaheuristic-based algorithms for their solution. Given efficient solution algorithms for individual scenarios, horizontal or scenario-based decomposition algorithms are particularly appealing. Consequently, PySP provides a generic and highly customizable implementation of Rockafellar and Wets' progressive hedging algorithm, with specific extensions to support mixed-discrete optimization. The talk will focus on the general PySP framework for approaching the deterministic-to-stochastic transition, provide specific code examples for several applications, and outline some of the algorithmic research questions that can be addressed.

1 Introduction

Historically, metaheuristics research has overwhelmingly focused on the development of efficient approximation algorithms for solving difficult *deterministic* optimization problems. However, uncertainty is pervasive in real-world problems and simply representing stochastic parameters via point approximations (e.g., distribution means or worst-case bounds) commonly yields overly conservative and costly solutions. Alternatively, one can represent stochastic variants of most well-studied combination optimization problems (e.g., facility location, logistics, scheduling, and routing) as stochastic programs [1, 7], in which parameter uncertainty is captured via a scenario tree and the objective is (typically) to minimize the expected cost over all scenarios. While possessing the advantage of integrated modeling of uncertainty, the practical computational difficulty of stochastic programs, particularly in the mixed-integer case, is well known. Thus, it is highly desirable in principle to leverage the extensive research on algorithms devoted to solving deterministic optimization problems – problems for which the deterministic variants are already difficult enough to necessitate the introduction of metaheuristics.

Stochastic programs are generally solved via decomposition, due to the complexity of directly solving the extensive form of the problem. Examples of approaches include the L-shaped method [8], dual decomposition [2], and Progressive Hedging (PH) [6]. Dual decomposition and PH are known as horizontal or scenario-based decomposition methods, as they decompose the problem by individual, deterministic scenarios. In doing so, it is conceptually possible to leverage existing, highly efficient solvers for individual scenarios - including general mixed-integer solvers and problem-specific metaheuristics. Lokketangen and Woodruff [5] pioneered such an approach in the context of PH, in which scenario-based

decomposition is effectively used as a metaheuristic (more precisely, a matheuristic in present terminology) for solving difficult, stochastic mixed-integer problems. This approach has seen broad use over the last 15 years, e.g., see [4, 3]. Yet despite this potential, there is a significant effort associated with developing the code infrastructure associated with a full-featured PH implementation. Our objective in this talk is to introduce a Python-based software package that provides – we argue – a "95%" solution, in addition to the necessary interfaces for users to supply the remaining 5%. PySP enables users to quickly implement and test a PH-based metaheuristic, allowing for more time to explore algorithmic ideas and avoiding the months-long effort required to construct such an experimental infrastructure from scratch.

2 PySP: Introduction

Recently, we introduced the PySP open-source software library for modeling and solving stochastic (optionally multi-stage and mixed-integer) programs (`https://software.sandia.gov/trac/coopr/wiki/PySP`). PySP is part of the Coopr software library (`https://software.sandia.gov/trac/coopr`), which was recently accepted as part of IBM's COIN-OR open-source initiative. A key component of Coopr is Pyomo, which allows for the object-oriented expression of abstract (e.g., as is done in AMPL) and concrete deterministic mathematical programs. PySP extends Pyomo by providing mechanisms for expressing stochastic programs and the associated scenario tree. Specifically, the deterministic, single-scenario model is specified in Pyomo, while the scenario tree and associated uncertain parameter data is specified in PySP. Additionally, PySP provides generic algorithms for solving the resulting stochastic programs - directly via the extensive form or through decomposition via PH. The PH algorithm is generic and highly customizable, e.g., providing capabilities typically required to obtain effective PH implementation in the mixed-integer case [9]. In particular, we provide built-in mechanisms for linearization of the proximal quadratic objective function terms, setting the ρ parameter on a per-variable and/or per-scenario basis, detecting convergence, fixing variables, detecting cycling behavior, and accelerating convergence. Through Coopr and the third-party Pyro (`http://www.xs4all.nl/~irmen/pyro3`) library, we also provide parallel computing facilities for distributed computation. The use of Python allows for rapid prototyping, improving productivity and allowing more time for exploration of algorithm design alternatives.

3 PySP: Interfacing with Deterministic Metaheuristics

To interface PySP with a deterministic metaheuristic, a user must develop – in Python – a problem-specific solver *plugin*. All solver plugins provide two specific functions: (1) the ability to execute a solve given a particular Pyomo deterministic problem instance and (2) the ability to load a solver-specific solution into a Pyomo problem instance. Examples of solver plugins provided in Coopr include GUROBI, CPLEX, and CBC – all mixed-integer solver interfaces. In the talk, we will illustrate the development of solver-specific plugins through two examples: a p-median GRASP metaheuristic and a multi-commodity flow tabu search metaheuristic. The coding tasks are very specific and isolated in both cases. First, the user must translate an input Pyomo problem instance into a problem-specific input file format. Second, the user must extract the solution from a problem-specific output format and load the solution into the Pyomo instance. Due to a combination of the brevity of Python and the streamlined design of Pyomo, both tasks can be accomplished in less than 100 lines of Python code. With knowledge of Python, Pyomo, and the problem-specific metaheuristic solver, the task can be accomplished in a matter of hours.

4 Augmenting the Metaheuristic's Objective Function

A complicating factor in the use of PH to solve stochastic programs via deterministic scenario solvers involves the need for the introduction in the objective of linear and quadratic "penalty" terms for all non-anticipative variables. These terms are required to gradually enforce convergence of PH to an implementable solution, and must be incorporated into the deterministic metaheuristic solver. Because they are isolated to the objective function, these terms commonly can be handled with little or no solver modifications, specifically in the widespread case of iterative or local search algorithms; the move operators

are rarely impacted. However, situations can arise. The talk will discuss how such integration is handled in the case of our p-median and multicommodity network flow examples.

5 Research Issues with Progressive Hedging

Given an integrated metaheuristic solver, PySP allows users to rapidly reach the point where it is possible to execute PH to solve a stochastic optimization problem of interest using an existing deterministic metaheuristic. And while PH will execute "out of the box", it will in all likelihood not be particularly efficient, nor is guaranteed to converge. Metaheuristics researchers are acutely aware of the impact of parameters and algorithm configuration on performance. In the case of PH, critical research questions include the specification of the ρ parameter, detection and acceleration of convergence, stability of scenario sub-problem solutions, run-times allocated to solve the scenario sub-problems, and the accuracy with which scenario sub-problems need to be solved. Our goal in introducing PySP to the metaheuristics community (its use thus far has been isolated to the mathematical programming community) is to minimize the time expended by researchers to reach the point where such questions can start to be answered, and to maximize the time spent there – rather than (re-) developing an extensive and intricate computational environment.

6 Acknowledgments

This work was funded in part by the US Department of Energy's Office of Advanced Scientific Computing Research and Sandia's Lab-Directed Research and Development (LDRD) program. 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.

References

[1] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, 1997.

[2] C.C. Caroe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1–2):37–45, 1999.

[3] T.G. Cranic, X. Fu, M. Gendreau, W. Rei, and S.W. Wallace. Progressive hedging-based metaheuristics for stochastic network design. Technical Report CIRRELT-2009-03, University of Montreal CIRRELT, January 2009.

[4] Y. Fan and C. Liu. Solving stochastic transportation network protection problems using the progressive hedging-based method. *Networks and Spatial Economics*, 10(2):193–208, 2010.

[5] A. Løkketangen and D. L. Woodruff. Progressive hedging and tabu search applied to mixed integer (0,1) multistage stochastic programming. *Journal of Heuristics*, 2:111–128, 1996.

[6] 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.

[7] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial and Applied Mathematics, 2009.

[8] R.M. Van Slyke and R.J. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[9] J.P. Watson and D.L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 2010.