Mario R. Eden, Marianthi Ierapetritou and Gavin P. Towler (Editors) Proceedings of the 13th International Symposium on Process Systems Engineering – PSE 2018
July 1-5, 2018, San Diego, California, USA

Pyomo.GDP: Disjunctive Models in Python

Qi Chen[a,*], Emma S. Johnson[b,c], John D. Siirola[c], Ignacio E. Grossmann[a]

[a]*Center for Advanced Process Decision Making, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA*
[b]*H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, 755 Ferst Drive, Atlanta, GA 30318, USA*
[c]*Sandia National Laboratories, Albuquerque, NM 87123, USA*
qichen@andrew.cmu.edu

Abstract

In this work, we describe new capabilities for the Pyomo.GDP modeling environment, moving beyond classical reformulation approaches to include non-standard reformulations and a new logic-based solver, GDPopt. Generalized Disjunctive Programs (GDPs) address optimization problems involving both discrete and continuous decision variables. For difficult problems, advanced transformations may be necessary such as intersecting multiple disjunctions via the disjunctive "basic step" or applying procedural reformulations. Large nonlinear GDP models may also be tackled using logic-based outer approximation. These expanded capabilities highlight the flexibility that Pyomo.GDP offers modelers in applying novel strategies to solve difficult optimization problems.

Keywords: python, Pyomo, modeling, optimization, disjunctive programming

1. Introduction

Optimization problems involving both discrete and continuous variables are commonplace in Process Systems Engineering (PSE) applications, including process design, planning, scheduling, and operations. Modelers typically formulate these problems as MILPs, or if nonlinear, MINLPs. However, there is no unique model formulation; for the same discrete-continuous problem, some formulations may be tractable while others may not (Grossmann and Trespalacios, 2013).

We claim that most MILP/MINLP formulations arise from attempts to model intuitively disjunctive (logical this-OR-that) problems. Generalized Disjunctive Programming (GDP), proposed by Raman and Grossmann (1994), provides a high-level modeling construct for expressing this application logic intuitively in the form of disjunctions and then applying systematic solution strategies. For example, unit existence in a process network is a disjunction: a candidate unit may exist—and its performance constraints enforced—or it may be absent—and its extensive quantities set to zero (Chen and Grossmann, 2017). Grossmann and Trespalacios (2013) gives the general form of GDP and MILP/MINLP models. Once a GDP model is posed, we can apply reformulations to convert the model structure to a MILP/MINLP and use available mixed-integer solvers, or we can solve the GDP directly using logic-based algorithms. The flexibility of being able to describe a problem using a single GDP model, and then to attempt different solution strategies is invaluable in addressing difficult PSE optimization challenges.

GDP modeling and optimization capability exists in both the open source Pyomo (Hart et al., 2017) and commercial GAMS (GAMS, 2017) algebraic modeling languages (AMLs),

making it accessible to a wide audience of modelers. Both AMLs support the formulation of disjunctions and the standard conversions to MILP/MINLP: the big-M (BM) reformulation described in Raman and Grossmann (1994) and the hull reformulation (HR) described in Furman et al. (2016). GAMS also allows the direct solution of GDP problems using logic-based outer approximation (LOA) via the LOGMIP solver (Vecchietti and Grossmann, 1997). However, the flexibility to develop and utilize more advanced GDP solution techniques may be necessary to tackle challenging problems.

A key challenge in reformulating GDP models is the generation of MILPs/MINLPs with tight continuous relaxations without growing the problem to an intractable size (Ruiz and Grossmann, 2017). BM leads to smaller formulations, but HR yields tighter relaxations. There has therefore been interest in advanced reformulations to better address this trade-off. Trespalacios and Grossmann (2016) introduced a method for strengthening the BM formulation using cutting planes derived from the HR, leading to a hybrid reformulation tighter than the BM, but smaller than the HR. Another recent development has been the introduction of basic steps for GDP by Ruiz and Grossmann (2012) building on the disjunctive programming concept introduced by Balas (1985). By performing basic steps, the GDP formulation is progressively transformed into disjunctive normal form, tightening the HR of the GDP, and consequently its continuous relaxation. For difficult problems, these advanced reformulations can make a difference in tractability; however, they have not yet been implemented as general-use GDP transformations in a modeling environment, limiting their application.

In some cases, particularly for nonlinear GDP models with higher complexity in the continuous space, it can be useful to solve the GDP using direct decomposition algorithms such as logic-based outer approximation (LOA) (Türkay and Grossmann, 1996). However, initialization can be an issue for LOA, both for the primary algorithm as well as its NLP sub-problems. Therefore, there is a need for a flexible algorithm that modelers can tune to fit their needs and adapt to incorporate domain-specific knowledge.

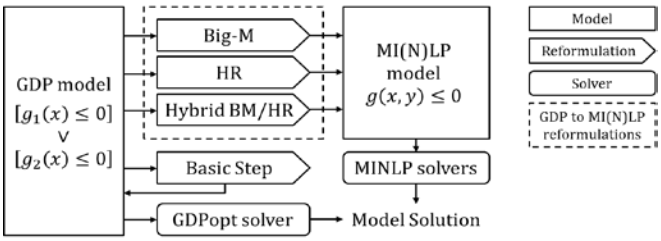

Figure 1. GDP model solution approaches with Pyomo.GDP

In this paper, we present new implementations in Pyomo.GDP enabling use of advanced reformulations such as hybrid BM/HR cutting planes and basic steps, and the flexible LOA implementation in the new direct GDP solver GDPopt. Figure 1 shows how these new capabilities fit in for the solution of a GDP model. In section 2, we discuss the advanced reformulations. In section 3, we present details on GDPopt. We demonstrate hybrid cuts and GDPopt with a case study in section 4, and we conclude in section 5.

2. Advanced GDP reformulations to MILP/MINLP

When reformulating a GDP model to MILP/MINLP, the main challenge is achieving the right compromise between problem size and tightness of the continuous relaxation. We introduce two new automatic tools that modelers can use to find a good trade-off: hybrid big-M cutting planes and GDP basic steps.

2.1. Hybrid BM/HR cutting plane algorithm

The hybrid BM/HR reformulation is based on the idea that cutting planes can be derived from the HR to strengthen the BM relaxation without explicit addition of many new variables and constraints (Trespalacios and Grossmann, 2016). We refer the reader to Section 6 and Figure 13 of Vecchietti et al. (2003) for theoretical background on the BM/HR cutting plane algorithm implemented in Pyomo.GDP.

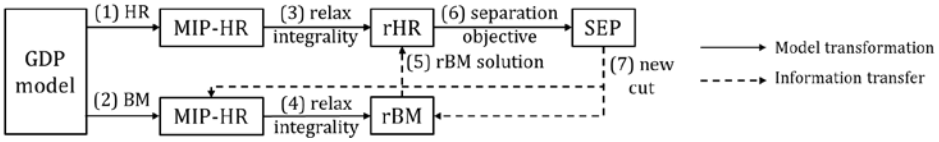

Figure 2. Hybrid BM/HR cutting plane algorithm

The Pyomo.GDP cutting plane implementation relies on pre-existing automatic transformations for BM and HR. The algorithm is outlined in Figure 2. Given a Pyomo concrete model `m`, the following command will generate a MILP/MINLP representation:

```
TransformationFactory('gdp.cuttingplane').apply_to(m)
```

Following transformation, the MILP/MINLP may be solved using any standard solver.

2.2. GDP Basic Steps

Basic steps in GDP bring a formulation incrementally from conjunctive normal form to disjunctive normal form by intersecting disjuncts with constraints or other disjuncts. Ruiz and Grossmann (2012) show that by doing so, the HR of the problem is tightened at the expense of a growth in the number of disjuncts. The transformation that applies a basic step between disjunctions “d” and “e” is implemented with the following command:

```
TransformationFactory('gdp.basic_step').apply_to(m, targets=[m.d, m.e])
```

The result is a transformation from a GDP model to another GDP model with a basic step applied. The tightened GDP could then be reformulated to MILP/MINLP or used in the context of the hybrid BM/HR cutting plane algorithm.

3. GDPopt: a flexible logic-based nonlinear GDP solver for Pyomo

We introduce GDPopt, a logic-based solver with a LOA implementation built on top of the Pyomo.GDP framework. GDPopt offers an alternative to reformulation of nonlinear GDPs into MINLPs. For PSE applications with complex nonlinear functions in the continuous space, such as those resulting from mixing, reaction, and equilibrium relations in process networks, LOA allows solution of reduced space NLP sub-problems rather than the full-space representations encountered when solving MINLPs, thereby improving robustness.

Figure 3 illustrates the LOA algorithm. From the original nonlinear GDP model, (1) an initialization procedure is used to generate a linearization, yielding a linear GDP model. (2) An automatic reformulation is then applied to the linear GDP model to yield the MILP master problem. The solution to the MILP master problem provides a lower bound on the overall (minimization) problem as well as (3) a proposed realization of the discrete variables. By (4) fixing the disjunctions and Boolean variables of the nonlinear GDP model based on the optimal MILP master problem solution, we obtain a reduced space NLP model omitting the constraints in inactive disjuncts. The optimal NLP solution gives

an upper bound on the overall (minimization) problem as well as the optimal continuous variable values for (5) generating an outer approximation (OA) cut. GDPopt uses OA/ER cuts described in Viswanathan and Grossmann (1990). The algorithm then loops back through steps 2-6 until a convergence criterion is met. GDPopt also supports the hybrid mixed-integer/GDP models described in Vecchietti and Grossmann (1997).

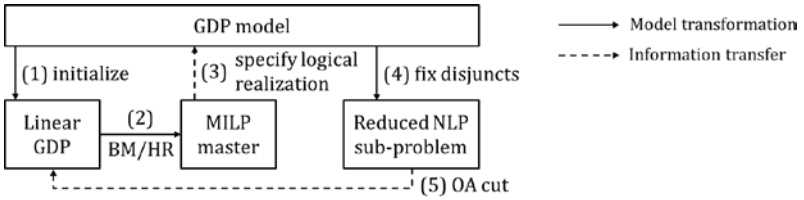

Figure 3. Logic-based outer approximation flow diagram

Two major challenges in applying LOA are the initial linearization of the nonlinear GDP and the solution of the NLP sub-problems. Türkay and Grossmann (1996) introduce a set-covering algorithm to ensure that linearizations are generated for each disjunct. We propose a modification whereby the set cover only needs to include disjuncts that contain nonlinear constraints. Taking advantage of Pyomo model characterization capabilities, GDPopt supports this new set-covering scheme, reducing the number of initialization sub-problems needed for LOA. Furthermore, because GDPopt is written in Python, it offers advanced modelers great flexibility to shape the algorithm according to their needs. GDPopt supports the use of custom initialization schemes for LOA as well as the capability to define callbacks before or after every LOA step outlined above. These callbacks have full access to the underlying Pyomo models used by LOA, and they can even be used to implement custom initialization schemes for the NLP sub-problems, providing warmstart information to the NLP solver that can be important for convergence (Biegler, 2010).

Usage of GDPopt is similar to the invocation of any other Pyomo solver:

```
SolverFactory('gdpopt').solve(m)
```

4. Case study

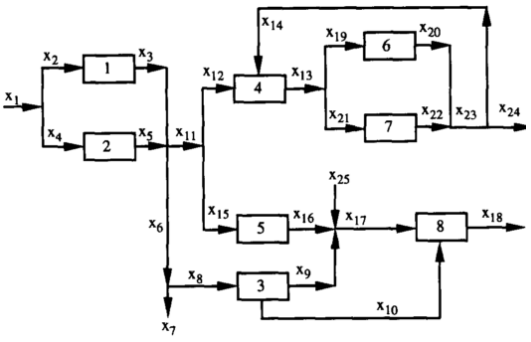

Figure 4. Eight process problem superstructure

We demonstrate the application of advanced reformulations and GDPopt using a disjunctive formulation of a process design problem from Duran and Grossmann (1986). Figure 4 shows the superstructure for the selection among eight process units.

The goal is to minimize annual cost of operation for a process with eight candidate units by determining the optimal flowsheet structure and flowrates. The mathematical formulation for this problem can be found in Appendix A of Türkay and Grossmann (1996). For brevity, we illustrate below a selection of the model-building and solution

steps for the case study in Pyomo.GDP. The complete code is publicly available as a test problem in the Pyomo Github repository.

All models begin with declaration of an empty model object:

```
m = ConcreteModel(name='Eight Process Problem')
```

We can then define model variables, constraints, and disjunctions, such as the one between selection of unit 1 versus unit 2, as shown in Equation 2:

$$\begin{bmatrix} Y_1 \\ \exp(x_3) - 1 = x_2 \\ x_4 = x_5 = 0 \end{bmatrix} \bigvee \begin{bmatrix} Y_2 \\ \exp\left(\frac{x_5}{1.2}\right) - 1 = x_4 \\ x_2 = x_3 = 0 \end{bmatrix} \tag{2}$$

```
m.use_unit1 = Disjunct(); m.use_unit2 = Disjunct()
m.use_unit1.inout1 = Constraint(expr=exp(m.flow[3])-1 == m.flow[2])
m.use_unit1.no_unit2_flow1 = Constraint(expr=m.flow[4] == 0)
m.use_unit1.no_unit2_flow2 = Constraint(expr=m.flow[5] == 0)
m.use_unit2.inout2 = Constraint(expr=exp(m.flow[5]/1.2)-1 == m.flow[4])
m.use_unit2.no_unit1_flow1 = Constraint(expr=m.flow[2] == 0)
m.use_unit2.no_unit1_flow2 = Constraint(expr=m.flow[3] == 0)
m.use1or2 = Disjunction(expr=[m.use_uni1, m.use_uni2])
```

Table 1. MINLP statistics following BM, HR, and Hybrid transformations

	NLP Relaxation	# vars	# binary	# constraints
HR	67.4	84	12	196
BM	-435.4	37	12	83
Hybrid BM/HR	-140.29	37	12	84

Once defined, the GDP model can be transformed or solved. Table 1 shows the problem size and root node NLP solution for the standard reformulations, demonstrating the trade-off between relaxation quality and problem size. All three examples solve to the optimal solution of 68.0 within 1 CPUs using the DICOPT solver. This example demonstrates the impact of the hybrid BM/HR cutting planes on the relaxation, as the addition of a single constraint, improves the BM relaxation by 50%.

Table 2. Iteration results for GDPopt

Iteration	Z_{NLP} (UB)	Z_L (LB)
Set cover #1	68.0	
Set cover #2	100.3	
1		68.0

Another option is to solve the GDP model directly using GDPopt. Table 2 displays the GDPopt iteration results. Compared to Table 5 in Türkay and Grossmann (1996), only two set covering sub-problems are necessary, reducing the number of NLPs that must be solved for initialization. Subsequently, the algorithm converges after the first master iteration.

5. Conclusions

In this work, we introduce new GDP solution capabilities for the Pyomo.GDP ecosystem, including the hybrid BM/HR cutting planes transformation, GDP basic steps, and the GDPopt logic-based solver. These additions highlight how the Pyomo algebraic modeling language provides a first-class framework for GDP modeling and optimization of discrete-continuous problems in PSE. By modeling in Pyomo.GDP, users can directly express problem logic and leverage a variety of automated solution techniques Further, as an open tool, advanced modelers are able to modify the transformations and GDPopt solution algorithm to incorporate their domain-specific knowledge.

Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

E. Balas, 1985. Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems. SIAM J. Algebr. Discret. Methods 6, 466–486.

L.T. Biegler, 2010. Nonlinear Programming. Society for Industrial and Applied Mathematics.

Q. Chen, I.E. Grossmann, 2017. Recent Developments and Challenges in Optimization-Based Process Synthesis. Annu. Rev. Chem. Biomol. Eng. 8, 249–283.

M.A. Duran, I.E. Grossmann, 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. 36, 307.

K.C. Furman, N. Sawaya, I.E. Grossmann, 2016. A computationally useful algebraic representation of nonlinear disjunctive convex sets using the perspective function. Optimization Online.

GAMS Development Corporation, 2017. General Algebraic Modeling System (GAMS) Release 24.9.2.

I.E. Grossmann, F. Trespalacios, 2013. Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. AIChE J. 59, 3276–3295.

W.E. Hart, C.D. Laird, J.-P. Watson, D.L. Woodruff, G.A. Hackebeil, B.L. Nicholson, J.D. Siirola, 2017. Pyomo — Optimization Modeling in Python. 2nd ed. Springer.

R. Raman, I.E. Grossmann, 1994. Modelling and computational techniques for logic based integer programming. Comput. Chem. Eng. 18, 563–578.

J.P. Ruiz, I.E. Grossmann, 2017. Global optimization of non-convex generalized disjunctive programs: a review on reformulations and relaxation techniques. J. Glob. Optim. 67, 43–58.

J.P. Ruiz, I.E. Grossmann, 2012. A hierarchy of relaxations for nonlinear convex generalized disjunctive programming. Eur. J. Oper. Res. 218, 38–47.

F. Trespalacios, I.E. Grossmann, 2016. Cutting Plane Algorithm for Convex Generalized Disjunctive Programs. INFORMS J. Comput. 28, 209–222.

M. Türkay, I.E. Grossmann, 1996. Logic-based MINLP algorithms for the optimal synthesis of process networks. Comput. Chem. Eng. 20, 959–978.

A. Vecchietti, I.E. Grossmann, 1997. LOGMIP: a disjunctive 0-1 nonlinear optimizer for process systems models, in: PSE '97-ESCAPE 7. Pergamon Press Ltd, pp. S427–S432.

A. Vecchietti, S. Lee, I.E. Grossmann, 2003. Modeling of Discrete/Continuous Optimization Problems: Characterizaton and Formulations of Disjunctions and their Relaxations. Comput. Chem. Eng. 27, 433–448.

J. Viswanathan, I.E. Grossmann, 1990. A combined penalty function and outer-approximation method for MINLP optimization. Comput. Chem. Eng. 14, 769–782.