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 © 2018 Elsevier B.V. All rights reserved.

Mixed-Integer Nonlinear Decomposition Toolbox for Pyomo (MindtPy)

David E. Bernal*, Qi Chen, Felicity Gong, and Ignacio E. Grossmann

Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh 15213, PA, USA
bernalde@cmu.edu

Abstract

In this paper, we present a new software framework developed in Pyomo, MindtPy (Mixed-Integer Nonlinear Decomposition Toolbox for Pyomo) which implements several decomposition methods for solving Mixed-Integer Nonlinear Programs (MINLP). These methods use a decomposition scheme of MINLP in Mixed-integer Linear Programs (MILP) and Nonlinear Programs (NLP). This toolbox allows the user to specify certain algorithmic options for these methods, which were not available in a single solver. Using Process Systems Engineering applied problems; the computational implementation and performance are discussed.

Keywords: Mixed-integer nonlinear programming, decomposition methods, Pyomo.

1. Introduction

Optimization problems in Process Systems Engineering (PSE) can be expressed with algebraic equations and decision variables, and be solved via mathematical programming. When these problems include nonlinear equations in the objective and/or constraints and both continuous and discrete variables they become Mixed-Integer Nonlinear Programs (MINLP).

The nature of the functions involved in optimization in PSE motivated the development of solution methods for both generalized and specialized MINLP models. Convex MINLP models are of special interest since their continuous relaxation gives rise to convex problems with a unique optimum value.

The general form of a MINLP problem is shown in Eq. (1) below.

$$\begin{aligned} Z = \min_{w,y} f(w,v,y) \\ \text{s.t.}: g(w,v,y) \le 0 \\ A_1 w + A_2 v + By \le b \\ v \in V \subseteq \mathbb{R}^{n_v}, w \in W \subseteq \mathbb{R}^{n_w}, y \in Y \subseteq \mathbb{Z}^{n_y} \end{aligned} \tag{1}$$

In Eq. (1) we partition the continuous variables as $[w\ v]^T = x \in X = W \times V$ with w being those continuous variables involved in only linear terms in the constraints, and v the variables involved in nonlinear terms.

MINLP problems as the one shown in Eq. (1) can be solved Branch and Bound (B&B) methods and decomposition methods. In B&B, a systematic enumeration of the possible combinations of discrete variables is performed, followed by the solution of simpler continuous subproblems obtained by fixing the discrete variables.

Decomposition methods iterate between a relaxation of the original MINLP, using linear approximations of the original feasible region resulting in a Mixed-Integer Linear Programming (MILP), and the solution of a restricted subproblem where the original problem is projected in a certain discrete combination in the form of a Nonlinear Programming (NLP) subproblem. The relaxation provides a lower bound over the objective function, while the restriction, when feasible, provides an upper bound. The decomposition methods are designed to provide a monotonic improvement on the lower bound while providing new integer combinations to the subproblem to update the upper bound. If the bounds meet, the problem is solved to optimality.

In convex MINLP problems, decomposition methods such as Generalized Benders Decomposition (GBD) (Geoffrion, 1972), Outer-Approximation (OA) (Duran and Grossmann, 1986), and Partial Surrogate Cuts (PSC) (Quesada and Grossmann, 1992) are proved to converge to the optimal solution of convex MINLP problems. Another method is the Extended Cutting Plane (ECP) method (Westerlund and Pettersson, 1995), which avoids solving the nonlinear subproblem.

The motivation of this work is that the importance and difficulty of solving MINLP problems require the development of new methods and software. The decomposition methods rely on the solution of the building blocks of MINLP, MILP and NLP; which has improved considerably in recent years (Bonami et al., 2008).

Methods such as OA or ECP have been implemented as part of commercial MINLP solvers as DICOPT, α-ECP, and BONMIN, among others. In this paper, we present an open-source implementation of the OA, ECP, GBD, and PSC methods inside the Python Optimization Modeling Objects (Pyomo) (Hart et al., 2017) in an algorithmic framework called MindtPy. MindtPy exploits several advantages of Pyomo, e.g. optimization object-oriented programming, access to commercial NLP and MILP solvers, and packages in Python apart from Pyomo.

2. Decomposition methods

A common classification of MINLP regards its continuous relaxation, where if all the constraints and the objective are convex functions, then the MINLP is denoted as a convex MINLP (Lee and Leyffer, 2012) although MINLPs themselves are nonconvex.

The decomposition methods assume that there is a subset of the problem variables that when temporarily fixed, make the remaining optimization problem in the rest of the variables considerably more tractable. This is the case with MINLP problems, where the complicating variables are the integer variables, which when fixed, give rise to an NLP problem. The OA, GBD, and PSC methods solve a master MILP problem, which provides a lower bound and an integer combination, which when fixed results in the NLP subproblem.

The master problem of the GBD method at the iteration k is the one presented in Eq. (2).

$$
\begin{aligned}
& Z_{LB,GBD}^{k} = \min_{\eta,y} \eta \\
\text{s.t.:}\quad & \eta \geq f(v^k, w^k, y) + (\mu^k)^T g(v^k, w^k, y) \;\; \forall k \in K_{feas} \\
& (\mu^k)^T g(v^k, w^k, y) \leq 0 \;\; \forall k \in K_{infeas} \\
& A_1 w^k + A_2 v^k + By \leq b \\
& \eta \in \mathbb{R}^1, y \in Y \subseteq \mathbb{Z}^{n_y}
\end{aligned} \tag{2}
$$

Where μ^k are the Lagrange multipliers of the nonlinear constraints, and K_{feas} and K_{infeas} are the set of iterations where the NLP subproblems were feasible and infeasible, respectively. This problem only adds a single cut to the master problem per iteration.

In the OA method, the nonlinear constraints are replaced by linear constraints given by the 1st order Taylor approximations. Therefore, we obtain the master problem in Eq. (3) which is a valid relaxation of the MINLP.

$$
\begin{aligned}
& Z_{LB,OA}^{k} = \min_{\eta,x,y} \eta \\
\text{s.t.:}\quad & \eta \geq f(v^k, w^k, y^k) + \nabla f(v^k, w^k, y^k)^T [v - v^k \quad w - w^k \quad y - y^k]^T \;\forall k \in K \\
& g(v^k, w^k, y^k) + \nabla g(v^k, w^k, y^k)^T [v - v^k \quad w - w^k \quad y - y^k]^T \leq 0 \;\; \forall k \in K \\
& A_1 w + A_2 v + By \leq b \\
& \eta \in \mathbb{R}^1, v \in V \subseteq \mathbb{R}^{n_v}, w \in W \subseteq \mathbb{R}^{n_w}, y \in Y \subseteq \mathbb{Z}^{n_y}
\end{aligned} \tag{3}
$$

This method adds as many cuts as constraints plus the objective cut at each iteration. It can still converge to the optimal solution in finite iterations if only the cuts corresponding to the active constraints for each problem are added, reducing the master problem size.

The PSC method addresses the tradeoff between the master problem size and the strength of the derived cuts. The PSC method can be seen as a combination of the OA and the GBD methods, where the cuts are derived from the gradient-based linearizations of OA, and the KKT conditions as in GBD. The PSC master problem is shown in Eq. (4).

$$
\begin{aligned}
& Z_{LB,PSC}^{k} = \min_{\eta,v,y} \eta \\
\text{s.t.:}\quad & \eta \geq f(v^k, w, y) + \begin{bmatrix} \lambda^k \\ -\mu^k \end{bmatrix}^T \begin{bmatrix} g(v^k, w, y) & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} 1 \\ v - v^k \end{bmatrix} \;\forall k \in K_{feas} \\
& \begin{bmatrix} \lambda^k \\ -\mu^k \end{bmatrix}^T \begin{bmatrix} g(v^k, w, y) & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} 1 \\ v - v^k \end{bmatrix} \leq 0 \;\; \forall k \in K_{infeas} \\
& A_1 w^k + A_2 v + By \leq b \\
& \eta \in \mathbb{R}^1, v \in V \subseteq \mathbb{R}^{n_v}, w \in W \subseteq \mathbb{R}^{n_w}, y \in Y \subseteq \mathbb{Z}^{n_y}
\end{aligned} \tag{4}
$$

An interesting result is that the cuts in the GBD master problem are surrogates of those derived from the PSC methods, which are surrogates of the cuts in OA (Quesada and Grossmann, 1992). This fact results in the relation between the solutions of each master problem in Eq. (5).

$$
Z_{LB,GBD}^{k} \leq Z_{LB,PSC}^{k} \leq Z_{LB,OA}^{k} \tag{5}
$$

After solving each master problem, a candidate discrete combination y^{k+1} is found and fixed to solve an NLP subproblem. The NLP solution $(v^{k+1}, w^{k+1}, y^{k+1})$ is used to generate the next cuts in the master problem, and if feasible, it provides an upper bound.

As in OA, The ECP method also relies on the 1st order Taylor approximation cuts to generate a master problem, but instead of solving an NLP subproblem to obtain a new linearization point, it uses the previous master MILP solution. At each iteration, all the nonlinear constraints violated by a ε-tolerance are linearized and added to the master problem, converging to the optimal solution within that ε-tolerance.

Given the tradeoff between the size of the master problem and the strength of the lower bound predicted by it, none of the methods dominates the others and the performance depends on each problem. Another tradeoff is between solving or not solving the NLP subproblems, given that they can be expensive to solve but provide a better linearization point and an upper bound to the problem.

3. Implementation

The decomposition methods for convex MINLP can be seen as meta-algorithms, where the solution of the optimization problem relies on solutions of other subproblems provided by other solvers, namely MILP and NLP.

Given a convex MINLP written in Pyomo, our toolbox solves it using any of the previously discussed decomposition methods. The object-oriented structure of the optimization models in Pyomo allows us to generate a `block` component for our purposes. Inside the original model, the block `MindtPy_linear_cuts` stores all the data used such as the cuts in the master problem, or intermediate solutions.while solving the model.

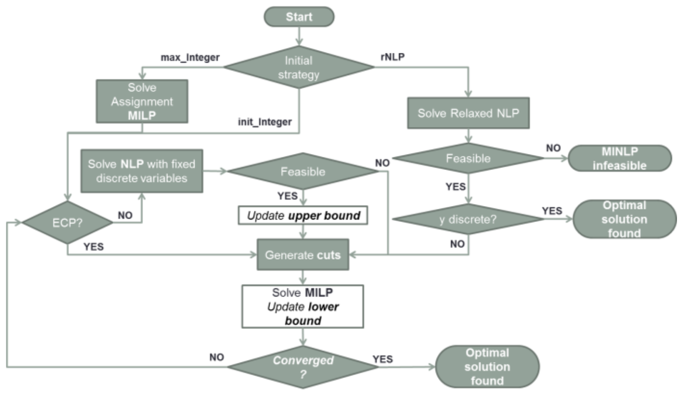

Figure 1. Simplified flow diagram of the MindtPy solution procedure

The first step after receiving the model is to perform an initialization for the decomposition methods. We require an initial integer combination to solve the nonlinear program, or a starting solution to that problem to populate the dual and gradient information required for the initial master problem. Given this, we implement several initialization procedures, 1) solve a continuous relaxation of the MINLP, providing a valid lower bound and a linearization point to generate the master problem; 2) use the user provided initial point as a fixed discrete combination and use this to solve the master problem; 3) maximize the sum of the discrete variables subject to the linear constraints in an assignment problem to obtain this initial discrete combination.

The gradients of the nonlinear constraints are calculated using the `differentiate` function in Pyomo, which performs an exact differentiation of the equations defining the corresponding derivatives. Given that these equations are encoded in expression trees, the differentiation is performed efficiently.

Other options, such as the inclusion of disjunctive cuts generated to ignore the previous solutions, known as integer cuts; the selection of different subsolvers for the MILP; and NLP problems, and the convergence tolerances are available for the user to modify. A simplified sketch of the algorithm flow-diagram is presented in Figure 1 and more can be found in the public repository of the project (Bernal and Chen, 2017)

4. Numerical example

To illustrate the use of MindtPy with different methods, we solve the problem in Eq. (6).

$$
\begin{aligned}
Z = \min_{x,y} & \; y_1 + 1.5y_2 + 0.5y_3 + x1^2 + x2^2 \\
\text{s.t.:} & \; (x_1-2)^2 - 2y_1 \le 0 \\
& \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ -1 & 0 \\ 0 & 1 \\ -1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \le \begin{bmatrix} 0 \\ 4 \\ -1 \\ 0 \\ -1 \\ -1 \end{bmatrix} \\
& 0 \le x \le 4, x \in \mathbb{R}^2, y \in \{0,1\}^3
\end{aligned} \tag{6}
$$

Table 1. Numerical example: Upper and lower bounds against iteration for each method.

Method	OA		PSC		GBD		ECP
Iteration	LB	UB	LB	UB	LB	UB	LB
1	1	11	1	11	-24	11	0.5
2	1.5	5	1.5	5	-23.5	11	0.5
3	3.5	3.5	3.5	3.5	-3.5	5	1
4					-2.5	5	2.02
5					3.5	3.5	3.5

Defining the problem as an optimization model inside Pyomo called `SimpleMINLP`, the command to solve this problem for example with OA and solving the relaxed NLP as an initial strategy we execute within Python importing Pyomo:

```
opt=SolverFactory('MindtPy')
opt.solve(SimpleMINLP, strategy = 'OA', initi_strategy = 'rNLP')
```

We solve the given problem using the four different methods and using an initial point $x^0 = 0, y^0 = 1$. The results for each iteration are presented in Table 1. The strength of the lower bound results in a convergence in fewer iterations for OA and PSC, while the ECP method takes more iterations since it does not solve the subproblem, although every iteration is computationally cheaper since it only requires the solution of an MILP.

5. Computational implementation and performance

In this section, we solved several convex MINLP with PSE applications from the MINLPLib2 (Vigerske, 2014). We used Gurobi7.5 and IPOPT3.12 as subsolvers. The tests were performed on a desktop running in Ubuntu16.04, with an Intel Core 2 Duo processor, a CPU of 3.4 GHz and 16 GB of RAM using Pyomo 5.2. We solve every instance with the four methods using the rNLP as initial strategy. The results are presented in Table 2. There we can see that for all the examples, OA managed to converge in fewer or equal iterations than PSC, which also converged in fewer iterations than GBD as mentioned in Eq. (5). We can see that ECP also converged in more iterations but the computational time per iteration was smaller, showing the trade-offs in performance.

Table 2. Solution details for different test cases using the 4 methods solved to $\varepsilon = 10^{-5}$. Termination without proof of optimality in less than 1000 iterations are marked with *

Method	OA		PSC		GBD		ECP	
Instance	Iters.	Time (s)	Iters.	Time (s)	Iters.	Time (s)	Iters.	Time (s)
`flay03m`	9	0.875	9	0.880	385	57.393	46	3.521
`batchdes`	2	0.247	4	0.402	45	16.703	4	0.265
`ex4`	3	1.152	369	147.713	430	182.987	8	1.526
`synthes3`	7	0.833	12	1.360	86	10.380	19	1.369
`enpro48pb`	3	1.064	3	1.002	*	*	6	1.239

6. Conclusions

An open source toolbox for MINLP solutions based on decomposition methods implemented in Pyomo was presented in this paper. This toolbox allows the users to modify several algorithmic options for these decomposition methods, allowing him/her to test easily different solution approaches to MINLP problems. MindtPy also provides algorithm designers with a platform to easily test and prototype ideas with access to all the capabilities of Pyomo, such as access to specialized subsolvers. We successfully implemented and tested using PSE related MINLP problems showing the flexibility of the Toolbox.

References

Bernal, D.E., Chen, Q., 2017. MindtPy [WWW Document]. URL https://github.com/bernalde/pyomo/tree/mindtpy

Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A., 2008. An algorithmic framework for convex mixed integer nonlinear programs. Discret. Optim. 5, 186–204. doi:10.1016/j.disopt.2006.10.011

Duran, M.A., Grossmann, I.E., 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. 36, 307–339. doi:10.1007/BF02592064

Geoffrion, A.M., 1972. Generalized Benders decomposition. J. Optim. Theory Appl. 10, 237–260. doi:10.1007/BF00934810

Hart, W.E., Laird, C.D., Watson, J.-P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D., 2017. Pyomo — Optimization Modeling in Python, Springer Optimization and Its Applications. Springer International Publishing, Cham. doi:10.1007/978-3-319-58821-6

Lee, J., Leyffer, S., 2012. Mixed integer nonlinear programming. Springer.

Quesada, I., Grossmann, I.E., 1992. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput. Chem. Eng. 16, 937–947. doi:10.1016/0098-1354(92)80028-8

Vigerske, S., 2014. Towards MINLPLib 2.0 Model instance collections.

Westerlund, T., Pettersson, F., 1995. An extended cutting plane method for solving convex MINLP problems. Comput. Chem. Eng. 19, 131–136. doi:10.1016/0098-1354(95)87027-X