Recent Advances in EAGO: Global and Robust Optimization in Julia

Matthew Wilhelm, Matthew D. Stuber

2018 AIChE Annual Meeting
Pittsburgh, PA, October 28th
Outline

Introduction to EAGO
  Motivation
  EAGO Optimization
  EAGO Toolkit

Algorithm Development
  Main Algorithm
  Relaxations
  Domain Reduction
Motivation

- We often model systems in terms of simulations.
- The resulting optimization problems are complex (nonsmooth, nonconvex).
- In development, we want to do things other than optimize.

Simulation
A large literature has developed around optimizing simulations. Can we provide enough flexibility to address these specialized forms?

No Barrier to Entry
Can we make a fast global solver that works like fmincon but provides strong guarantees?
The Julia Language

- Speed of C/Fortran
- Distributed Computing & Parallelism
- Extensive Compile Time Optimization

EAGO Optimization

- **Original Scope:** A general script-based solver that constructs McCormick relaxations by method overloading.
- **Issue:** "General" overloading approaches aren’t adequately general.
- **Issue:** Introducing auxiliary variables, rearranging expressions, and presolving can be quite beneficial.
Consider the kinetic parameter estimation problem given in [1]. Parameters are desired that minimize the sum-square error.

\[ \min_{p \in P} \sum_{i=1}^{n} \left( (x^i_A + \frac{2}{21} x^i_B + \frac{2}{21} x^i_D) - I^i_d \right)^2 \]

Decision parameters are the reaction rate constants

\[ p = (k_2 f, k_3 f, k_4) \in [100, 1200] \times [100, 1200] \times [0.001, 40] \]

The explicit Euler discretization of the kinetic mechanism is given by:

\[
x^{i+1}_A = x^i_A + \Delta t \left( k_1 x^i_Y x^i_Z - C_{O2} (k_2 f + k_3 f) x^i_A + \frac{k_2 f}{K_2} x^i_D + \frac{k_3 f}{K_3} x^i_B - k_5 (x^i_A)^2 \right)
\]

\[
x^{i+1}_B = x^i_B + \Delta t \left( k_3 f C_{O2} x^i_A - \left( \frac{k_3 f}{K_3} + k_4 \right) x^i_B \right)
\]

\[
x^{i+1}_D = x^i_D + \Delta t \left( k_2 f C_{O2} x^i_A - \frac{k_2 f}{K_2} x^i_D \right)
\]

\[
x^{i+1}_Y = x^i_Y + \Delta t \left( -k_1 s x^i_Y x^i_Z \right)
\]

\[
x^{i+1}_Z = x^i_Z + \Delta t \left( k_1 x^i_Y x^i_Z \right)
\]

The global optimal solution, within a 5 difference between lower and upper bounds, is found within 3 seconds.

Times comparable to C++ implementation in literature.
Consider the flooded bed bioreactor process optimization problem in [3].

$x_j$ input variables. The weight between variables $j$ and hidden layer node $i$ are $W_{ij}$ and the weight for the hidden node $i$’s output $D_i$.

\[
\begin{align*}
\max_{x \in X} & \quad B_2 + \sum_{i=1}^{3} \frac{2D_i}{1 + \exp(-2Y_i)} \\
Y_i &= B_i + \sum_{i=1}^{3} \sum_{j=1}^{8} W_{ij}x_j
\end{align*}
\]
EAGO provides native support for the JuMP AML [4].

JuMP provides automatic differentiation utilities and expression input via syntactic macros.

Similar complexity to Pyomo [5].

Example is solved in under less than one second.

---

Outline

Introduction to EAGO
  Motivation
  EAGO Optimization
  EAGO Toolkit

Algorithm Development
  Main Algorithm
  Relaxations
  Domain Reduction

Individual subroutines can be set using a simple API.

#Sets preprocessing rule of s to function f
@SetPreprocessingRule(s, f)

New framework for organizing relaxations

- All relaxations now generated from tape and parsed into expression graph via source-code transformation.
- Relaxations are registered with properties
  - Differentiable
  - MILP, etc.
- Schemes contain rules for composing relaxations using directed acyclic graph
  - FullAVM: Generate relaxation at each node.
  - SetValue: Propagate from user input.
  - User-defined schemes supported.
Relaxations - Currently Supported

- Outer-Approximations (LP/MIP) [8]
- $\alpha BB$-Type Relaxations [9]
- Interval Arithmetic [7]
- Convex/Concave Envelopes

---

7 Moore, R. Methods and Application of Interval Analysis, SIAM, 1979
Source-code transformation approach builds functions that additional Julia code can manipulate.

Integrates with Cassette.jl, Intel’s Parallel Accelerator or Cuda (CudaNative.jl [10]) utilities to generate relaxation functions into code that can be run on a GPU.

10 Besard, T. et al. IEEE Transactions on Parallel and Distributed Systems, 2018
Domain Reduction - OBBT

- Additional algorithms for: Poor Man’s LP/NLP [12], Newton/Krawczyk methods [13], and more.
- Integrates with new relaxation framework.

\[
\begin{align*}
\min_{x,y} & \quad x - y \\
y & = 0.1x^3 - 1.1x \\
-4 & \leq x \leq 4 \\
-2 & \leq y \leq 2
\end{align*}
\]

Domain Reduction - Constraint Walking

- Wengert tape storage for structures
- Adaptive methods or function built from DAG for constraint propagation (CP) [14]
- Supports for validated interval arithmetic CP [15]

---

14  Vu, X. et al. J. Global Optim, 45, 2008, 499
15  Moore, R. et al. Intro to Interval Analysis, SIAM 2009, 110
Future Work


- Automatic recognition of implicit functions in explicit forms.

- Incorporation of fixed-point routines for relaxing implicitly-defined functions for forming relaxations and domain reduction [13].

Acknowledgements

- Kamil Khan for the discussion on nonsmooth AD methods and differentiable McCormick relaxations
- Huiyi Cao for feedback on using EAGO
- University of Connecticut for providing funding
- The other members of the PSOR group at UCONN
Shameless Promotion...

Other EAGO Talks at AIChE

Session: Advances in Deterministic Global Optimization
Date: Tuesday, October 30, 2018
Session Time: 8:00 AM - 10:30 AM
Presentation Title: Quadratic Underestimators of Differentiable McCormick Relaxations for Deterministic Global Optimization
Presentation Time: 9:35 AM - 9:54 AM
Location: David L. Lawrence Convention Center, 409

Session: Advances in Computational Methods and Numerical Analysis
Date: Tuesday, October 30, 2018
Session Time: 12:30 PM - 3:00 PM
Presentation Title: Tightening McCormick Relaxations Via Reformulation of Intermediate Functions into Schema
Presentation Time: 2:05 PM - 2:24 PM
Location: David L. Lawrence Convention Center, 410
The EAGO suite is a registered Julia 1.0 package and can be installed as follows:

```julia
Pkg.add("EAGO")
```

Development versions, examples, and documentation may be accessed from the Processing System and Engineering Lab’s Github website:

https://github.com/PSORLab/EAGO.jl

Questions?