Recent Advances in EAGO: Global and Robust Optimization in Julia

Matthew Wilhelm, Matthew D. Stuber

2018 AIChE Annual Meeting
Pittsburgh, PA, October 28th

Process Systems and Operations Research Laboratory

UCONN SCHOOL OF ENGINEERING

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

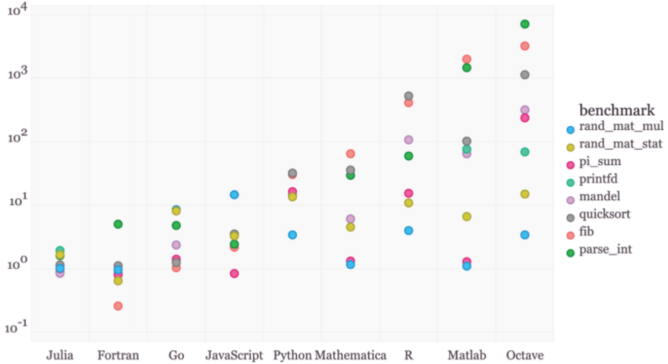
A large literature has developed around optimizing simulations. Can we provide enough flexibility to address these specialized forms?

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



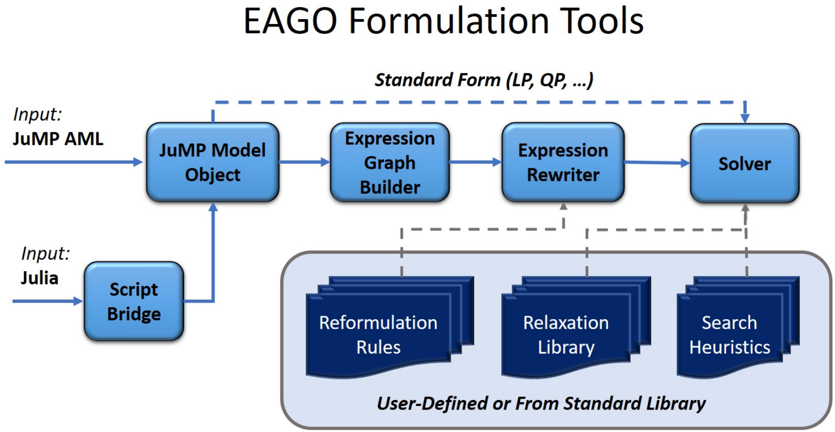
Performance comparison of various languages performing simple microbenchmarks. Benchmark execution time relative to C. (Smaller is better; C performance = 1.0.)

1 Bezanson, J, et al. SIAM Review, 59, (2017), p65-98

EAGO Optimization

```julia
struct Storage
    val::Float64
end

+(x::Storage,y::Storage) = x.val + y.val

function objective(x::Float64)
    # Type assertions in definition and storage
    y = zeros(Float64,2)
    # Operators nested in structures
    sto = Storage(x[1])
    sto += sto
    # Collection operators
    a = []
    push!(a,y[1]^2)
    # Control flow elements
    if (x > 0)
        y[1] = x^2 + x
    else
        y[1] = sin(x)
    end
    # Access from array structure
    y[1] + y[2] + sto.val + pop!(a)
end
```

- **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.


EAGO Formulation Tools
Standard Form (LP, QP, ...)
Input:
JuMP AML
JuMP Model Object
Expression Graph Builder
Expression Rewriter
Solver
Input:
Julia
Script Bridge
Reformulation Rules
Relaxation Library
Search Heuristics
User-Defined or From Standard Library

Interface - Script

Consider the kinetic parameter estimation problem given in [1]. Parameters are desired that minimize the sum-square error.

$$\min_{\mathbf{p}\in P} \sum_{i=1}^{n} \left((x_A^i + \frac{2}{21} x_B^i + \frac{2}{21} x_D^i) - I_i^d \right)^2$$

Decision parameters are the reaction rate constants

$$\mathbf{p} = (k_{2f}, k_{3f}, k_4) \in [100, 1200] \times [100, 1200] \times [0.001, 40]$$

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

$$
\begin{aligned}
x_A^{i+1} &= x_A^i + \Delta t \left(k_1 x_Y^i x_Z^i - C_{O2}(k_{2f} + k_{3f}) x_A^i + \frac{k_{2f}}{K_2} x_D^i + \frac{k_{3f}}{K_3} x_B^i - k_5 (x_A^i)^2 \right) \\
x_B^{i+1} &= x_B^i + \Delta t \left(k_{3f} C_{O2} x_A^i - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_B^i \right) \\
x_D^{i+1} &= x_D^i + \Delta t \left(k_{2f} C_{O2} x_A^i - \frac{k_{2f}}{K_2} x_D^i \right) \\
x_Y^{i+1} &= x_Y^i + \Delta t \left(-k_{1s} x_Y^i x_Z^i \right) \\
x_Z^{i+1} &= x_Z^i + \Delta t \left(k_1 x_Y^i x_Z^i \right)
\end{aligned}
$$

2 Mitsos A. et al. SIAM Journal on Optimization, SIAM, 20, (2009), 573-601

Interface - Script

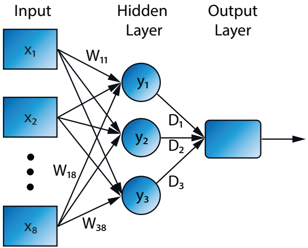
```julia
function fd(p,data)

    x = zeros(1000,typeof(p)); SSE = 0    # data storage array
    x[4] = 0.4; x[5] = 140                # sets initial condition

    # sets known parameter values
    T = 273; delT = 0.01; cO2 = 2e-3; k1 = 53; k1s = k1*1e-6
    K2 = 46exp(-6500/T-18); K3 = 2*K2;
    for i=1:200

        # Advances one time-step
        temp1 = delT*(k1s*x[5i-1]*x[5i]-cO2*(p[1]+p[2])*x[5i-4])
        temp2 = delT*(p[1]*x[5i-4]/K2+p[2]*x[5i-3]/K3-k5*x[5i-4]^2)
        x[5i+1] = x[5i-4] + temp1 + temp2
        x[5i+2] = x[5i-3] + delT*(p[2]*cO2*x[5i-4]-(p[2]/K3+p[4])x[5i-3])
        x[5i+3] = x[5i-2] + delT*(p[1]*cO2*x[5i-4]-p[1]/K2)
        x[5i+4] = x[5i-1] + delT*(-k1s*x[5i-1]*x[5i])
        x[5i+5] = x[5i] + delT*(k1*x[5i-1]*x[5i])

        # Updates the SSE value
        SSE += ((x[i+1] + (2/21)*x[i+2] + (2/21)*x[i+5]) - data[i])^2

    end

    return SSE
end

# load data from file to variable d
using JLD2, FileIO
@load "KineticData.jld2" d

# makes function with data fixed to file contents
f(x) -> fd(x,d)

# solve optimality problem
objective, solution, feasibility = ScriptSolve(f,lb,ub)
```

- 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.

$$\max_{\mathbf{x}\in X} B_2 + \sum_{i=1}^{3} \frac{2D_i}{1+\exp(-2Y_i)}$$

$$Y_i = B_i + \sum_{i=1}^{3}\sum_{j=1}^{8} W_{ij}x_j$$

- 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.



3 Cheema, J. et al. Biotechnology progress 18(6), 2002 p1356-1365.

4 Schweidtmann, AM., Mitsos Journal of Optimization Theory and Applications, 2018, p1-24.

Interface - JuMP

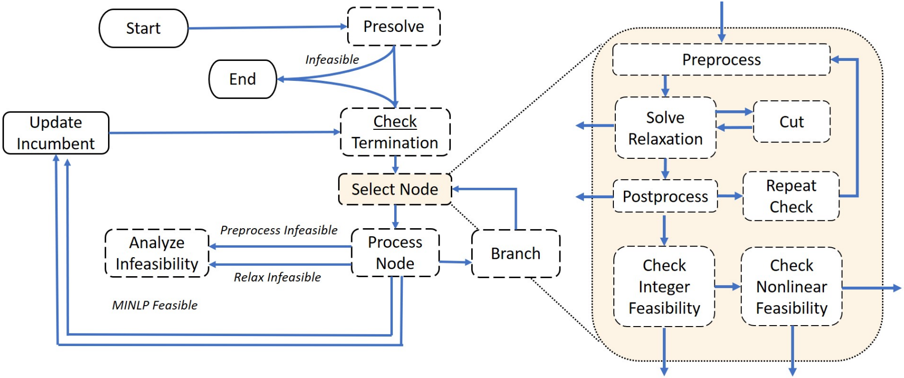
```julia
using JuMP
using EAGO

# Box constraints for input variables
xLBD = [0.623, 0.093, 0.259, 6.56, 1114,  0.013, 0.127, 0.004]
xUBD = [5.89,  0.5,   1,     90,   25000, 0.149, 0.889, 0.049]

# Weights associated with the hidden layer
W = [ 0.54, -1.97,  0.09, -2.14,  1.01, -0.58,  0.45, 0.26;
     -0.81, -0.74,  0.63, -1.60, -0.56, -1.05,  1.23, 0.93;
     -0.11, -0.38, -1.19,  0.43,  1.21,  2.78, -0.06, 0.40]

# Weights associated with the output layer
D = [-0.91 0.11 0.52]

# Bias associated with the hidden layer
B1 = [-2.698 0.012 2.926]

# Bias associated with the output layer
B2 = -0.46

# Model construction
model = Model(with_optimizer(EAGO.Optimizer()))
@variable(model, xLBD[i] <= x[i=1:8] <= xUBD[i])
@NLexpression(m, prop[i], B1[i] + sum(W[i,j]*x[i] for i=1:3,j=1:8))
@NLobjective(model, Max, B2 + sum(D[i]*(2/(1+exp(-2*prop[i])) for i=1:3))

# Solves the model
optimize!(model)

# Access functions for the solution
ObjectiveValue = getobjectivevalue(model)
Feasibility = getfeasibility(model)
Solution = getvalue(x)
```

- EAGO provides native support for the JuMP AML [4].
- JuMP provides automatic differentiation utilities and expression input via syntactic macros.
- Similiar complexity to Pyomo [5].
- Example is solved in under less than one second.

4 Dunning, I. et al. SIAM Review, 59, 2017, 295-320
5 Hart, W. et al. Mathematical Programming Computation, 3, 2011, 219

Outline

Solver Routine



- EAGO now supports a Branch-and-Cut [6] framework.
- Individual subroutines can be set using a simple API.

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


6 Rustem, B. et al. Optimization Methods and Software, 2001, 16, 21-47

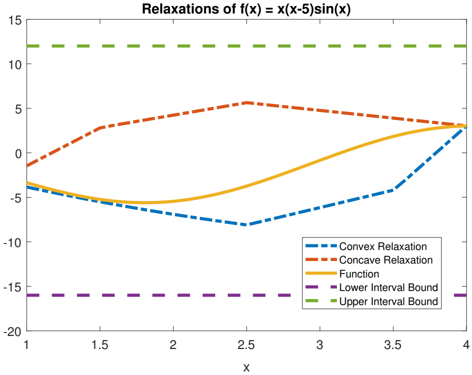
Relaxations - Framework

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]
- αBB-Type Relaxations [9]
- Interval Arithmetic [7]
- Convex/Concave Envelopes



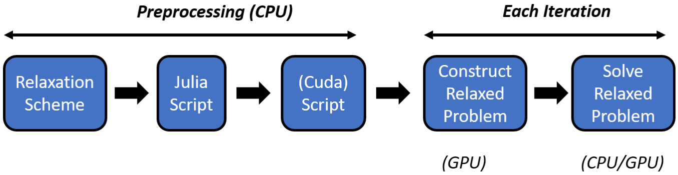
7 Moore, R. Methods and Application of Interval Analysis, SIAM, 1979

8 Tawarmalani, M. et al. Mathematical Programming, 99, 2004, 563-591

9 Adjiman, C.S et al. Computers & Chemical Engineering, 20, 1996, 419-424

Relaxations - GPU Programming

- 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.

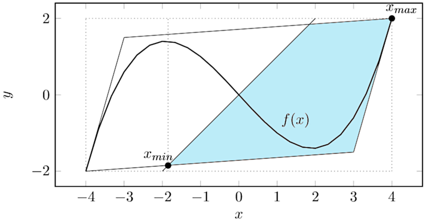


10 Besard, T. et al. IEEE Transactions on Parallel and Distributed Systems, 2018

Domain Reduction - OBBT

- Optimization-based bound tightening with filtering and greedy-ordering [11].
- Additional algorithms for: Poor Man's LP/NLP [12], Newton/Krawczyk methods [13], and more.
- Integrates with new relaxation framework.

$$\min_{x,y} x - y$$
$$y = 0.1x^3 - 1.1x$$
$$-4 \leq x \leq 4$$
$$-2 \leq y \leq 2$$



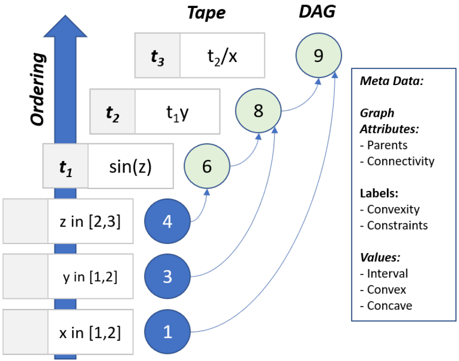
11 Gleixner, A. et al. J. Global Optim, 67, 2017, 731-757

12 Puranik, Y. et al. Constraints, 22, 2017, 338-376

13 Stuber, M. et al. Optimization Methods & Software, 30, 2015, 424-460

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

- JuMP extension for DAE models: Enabled by new MathOptInterface Bridge feature.
- 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].

13 Stuber, M. et al. Optimization Methods & Software, 30, 2015, 424-460

Acknowledgements

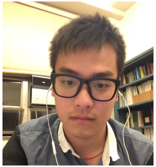
EAGO

UCONN
SCHOOL OF ENGINEERING

Process Systems and Operations Research Laboratory

- 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

Prof. Matthew Stuber



Chenyu Wang

William Hale

Connor Dion

Jacob Chicano

Abiha Jafri

Shameless Promotion...

Other EAGO Talks at AIChE

Session: **Advances in Determininstic 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

Fin...

- 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?