Reduced-Space Optimization of Data-Driven Hybrid Models in EAGO.jl

Matthew Wilhelm, PhD Candidate
Matthew Stuber, Assistant Professor
EAGO.jl: Global Deterministic Optimization of Simulations

EAGO.jl: A deterministic global optimizer in JuMP/Julia (for nonconvex MINLP via branch-and-cut)

EAGO.jl: Global Deterministic Optimization of Simulations

- EAGO.jl: A deterministic global optimizer in JuMP/Julia (for nonconvex MINLP via branch-and-cut)
- Can solve formulations with **user-defined expressions (simulations, etc.)**
- Uses composite relaxation framework which enables expansion to an esoteric set of problems

---

EAGO.jl: McCormick Relaxations

\[ y = f(g(x), ..., h(x)) \]

- Relaxations of \( g(x) \) at \( x \) in \( X \)
- Relaxations of \( h(x) \) at \( x \) in \( X \)
- Relaxations of \( f(x) \) at \( x \) in \( X \)

Akin to automatic differentiation but computes relaxations instead of derivatives

- EAGO generates relaxations of complicated nonlinear expressions using a McCormick relaxation methodology\(^2,3\)
- Intrinsic library of relaxations
- Hybrid source-code transformation approach
  - Method overloading for basic functions.
  - Store and analyze graph structure as well.

---

New Applications

**Implicit Functions**

\[ x(p) \mid h(x(p), p) = 0 \]

**ODEs and DAEs**

\[ z_k^{cc}(p), z_k(p), z_k^{cu}(p) \]

**Continuous Random Variables**

**Blackbox Functions**

---


5. Wilhelm, ME; Le, AV; and Stuber, MD. *Global Optimization of Stiff Dynamical Systems*. AIChE Journal: Futures Issue, 65 (12), 2019


New Applications

Implicit Functions

\[ x(p) \mid h(x(p), p) = 0 \]

ODEs and DAEs


INFORMS 2021 – October 24, 2021
EAGO.jl: Extended Nonlinear

New multi-graph representation:

- Extends JuMP + EAGO backend to introduce auxiliary variables
- Support for multiple-output subexpressions using new multigraph backend representation.

Composability of Nonlinear Functions (via JuMP extension):

- Facilities chaining blocks of general nonlinear expressions (such as implicitly defined functions).

```julia
function f!(d, y)
end

@variable(m, -2 <= x[i=1:2] <= 2)
@auxiliary_variable(m, z[i=1:2])

@constraint(m, sum(z) <= 0.0)

m = EAGOModel()

h_1(y, x) = 0
y = y(x)

h_2(z, y) = 0
z = z(y)

g(x) = f(z(y(x)))
```
EAGO.jl: Embedded ML

- Currently, EAGO.jl supports ML-models that are embedded with have a factorable representation.
- Additional work underway to natively support embedded Flux.jl models.
- Future support for more complex ML model structures
  - Layers with implicit functions evaluated via fixed-point methods (i.e. deep equilibrium networks).

\[
\hat{y}(u) = Cx + Du \\
x = \phi(Ax + Bu)
\]


Adapted from https://www.asimovinstitute.org/neural-network-zoo/
Peaks Function

\[ f(x, y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10 \left( \frac{x}{5} - x^3 - y^5 \right) e^{-x^2-y^2} - \frac{e^{-(x+1)^2-y^2}}{3} \]

function f(x,y)
    3*exp(-x^2-(y+1)^2)*(1-x)^2 - 10*(x/5-x^3-y^5)*exp(-x^2-y^2) - exp(-(x+1)^2-y^2)/3
end

Domain

\[-3 \leq x, y \leq 3\]

```
l = [-3; -3]
u = [3; 3]
```
Simple Surrogate Model

1. Generate Data

```
xd = sample(n, l, u, SobolSample())
yd = f.(xd)
```
Simple Surrogate Model

1. Generate Data

\[ xd = \text{sample}(n, l, u, \text{SobolSample}()) \]
\[ yd = f.(xd) \]

2. Specify Model

\[ m = \text{Chain}(\text{Dense}(2, 6, \text{tanh}), \text{Dense}(6, 1)) \]

Surrogates.jl Workflow for Model Development
Simple Surrogate Model

Surrogates.jl Workflow for Model Development

1. Generate Data
   \[ x_d = \text{sample}(n, l, u, \text{SobolSample}()) \]
   \[ y_d = f.(x_d) \]

2. Specify Model
   \[ m = \text{Chain}(\text{Dense}(2, 6, \text{tanh}), \text{Dense}(6, 1)) \]

3. Train Model
   \[ ns = \text{NeuralSurrogate}(x_d, y_d, l, u, \text{model} = m) \]
   \[ \text{surrogate_optimize}(f, \text{SRBF}(), l, u, ns, \text{SobolSample}()) \]
Optimize Surrogate Model

JuMP + EAGO.jl Workflow for Optimization

4. Create Model

\[
\text{using JuMP, EAGO}
\]

\[
m = \text{Model}(\text{EAGO.Optimizer}) \\
@\text{variable}(m, l[i] \leq x[i=1:2] \leq u[i])
\]

5. Register Function, Specify Problem

\[
@\text{register}(m, :f, 2, f) \\
@\text{NLO} \text{bjective}(m, \text{Min}, f(x[1], x[2]));
\]

6. Optimize the problem

\[
\text{optimize!(m)}
\]
Optimize Surrogate Model

- Dispatch to improved relaxations for a large library of activation functions automatically.
- Some support for “cleaning” script defined models for compatibility.
- Compatible with standard global optimization methods.

4. Create Model

```
using JuMP, EAGO

m = Model(EAGO.Optimizer)
@variable(m, l[i] <= x[i=1:2] <= u[i])
```

5. Register Function, Specify Problem

```
@register(m, :f, 2, f)
@NLObjective(m, Min, f(x[1], x[2]))
```

6. Optimize the problem

```
optimize!(m)
```
Next Steps

- Support for neural-ODE models.
- Improvements to composite relaxation forms (faster global optimization of surrogate models).
- Continually add support for additional ML forms.
  - Relaxation of implicit functions\textsuperscript{10}
  - Reverse propagation of relaxations\textsuperscript{11}

\textsuperscript{10} Stuber, MD et al. Convex and concave relaxations of implicit functions. Optimization Methods and Software (2015), 30, 424-460
Acknowledgements

- PSOR Lab @ UCONN
- INFORMS 2021 Organizers
- Funding: National Science Foundation

This material is based upon work supported by the National Science Foundation under Grant No. 1932723. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.