UCONN
UNIVERSITY OF CONNECTICUT

EAGO.jl: Next Generation Global & Robust Optimization in Julia, Revisited

Matthew Wilhelm, PhD Candidate
Robert Gottlieb, PhD Student
Matthew Stuber, Assistant Professor

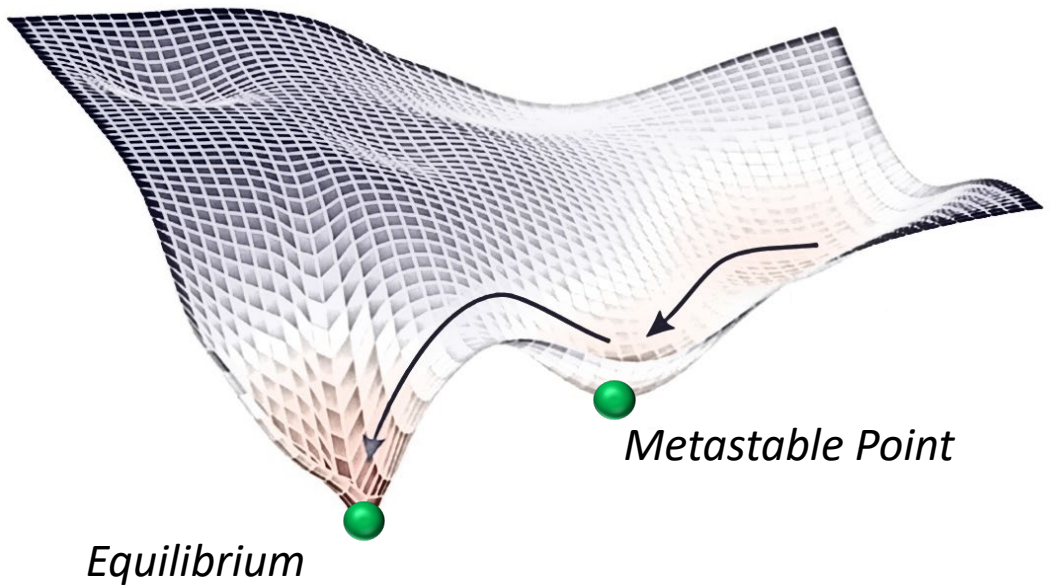
November 7th, 2021

2021 AIChE ANNUAL MEETING

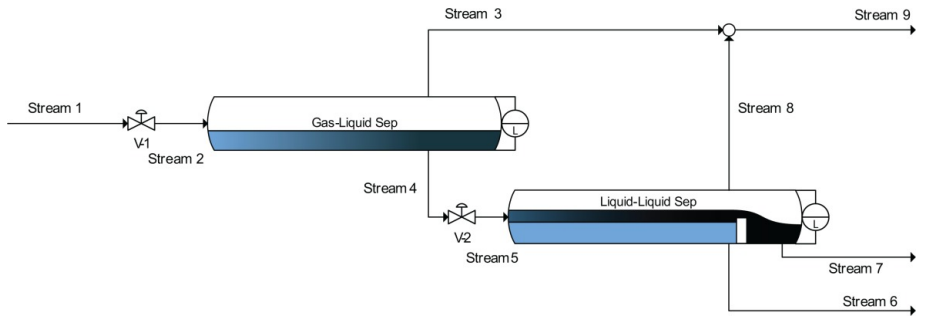
Process Systems and Operations Research Laboratory

Importance of Global Optimization

Physically Meaningful Computations[1]



Safety-Critical Systems[2]

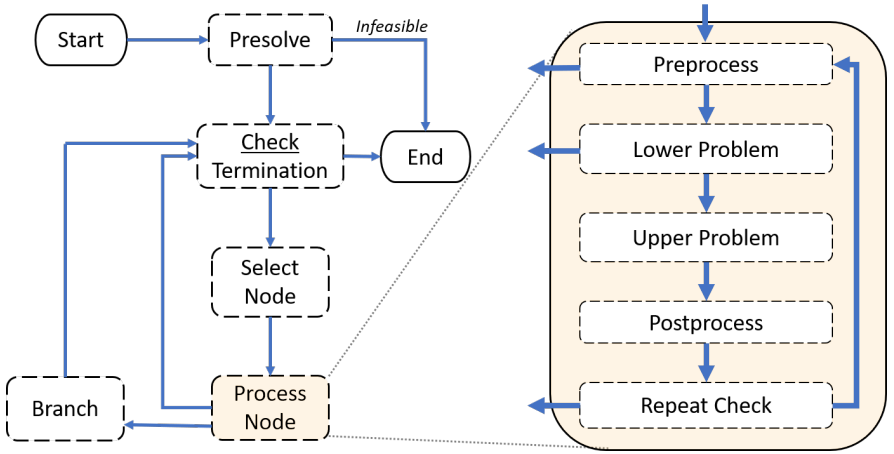


Better quality solutions than local methods…

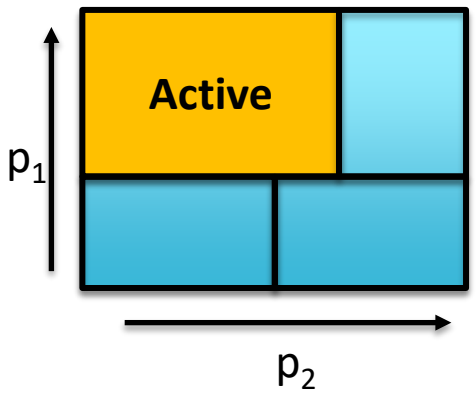
1. Grajcarova, L. **Simulations of structural phase transitions in crystals using ab initio metadynamics**. INIS-IAEA (2013)
2. Stuber, MD et al. **Worst-case design of subsea production facilities using semi-infinite programming.** *AIChE Journal* (2014): 2513-2524.

Global Optimization

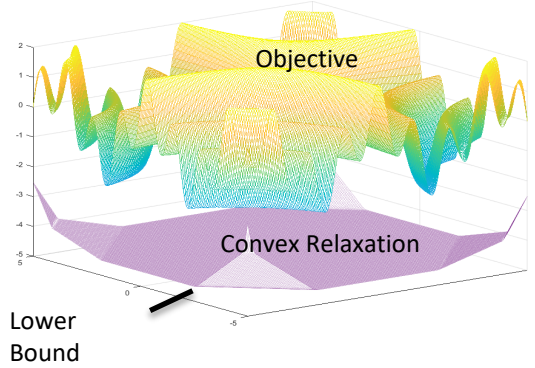
- Nonconvex MINLP formulations naturally arise in many applications.
- MINLP solvers generally rely on some variation of spatial branch-and-bound[3,4].
- Relaxed subproblems are used to compute bounds and are often derived from relaxed functions[3,4].



(Select Region)



(Relax Problem + Compute Bound)



3. Wilhelm, M.E., and Stuber, M.D.. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 1-26.
4. Horst, Reiner, and Hoang Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

McCormick Relaxations of Factorable Functions

$$\mathbf{y} = \mathbf{f}(\mathbf{g}(\mathbf{x}), \dots, \mathbf{h}(\mathbf{x}))$$

Auxiliary Variable Method

$$\mathbf{z_1} = \mathbf{g}(\mathbf{x})$$

$$\vdots$$

$$\mathbf{z_n} = \mathbf{h}(\mathbf{x})$$

$$\mathbf{y} = \mathbf{f}(\mathbf{z_1}, \dots, \mathbf{z_n})$$

→ **Relaxations of *f at (x, z) in (X, Z)***

McCormick Relaxation[5,6]

Relaxations of g at x in X

⋮

Relaxations of h at x in X

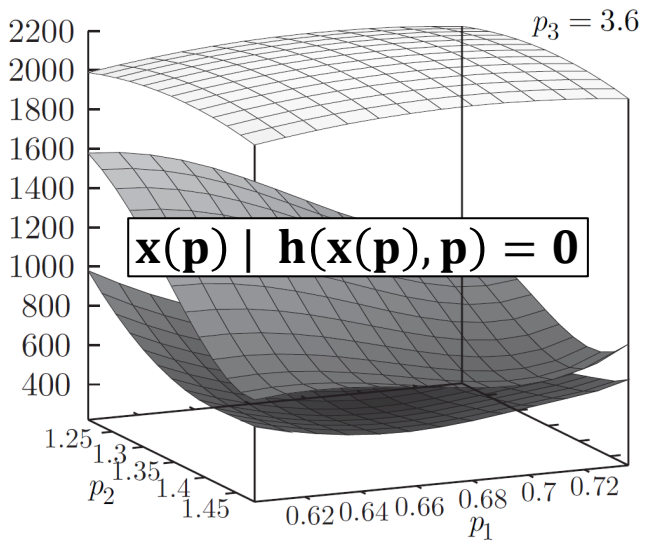
Apply **f** *composite relaxation rules* →

Relaxations of *f at x in X*

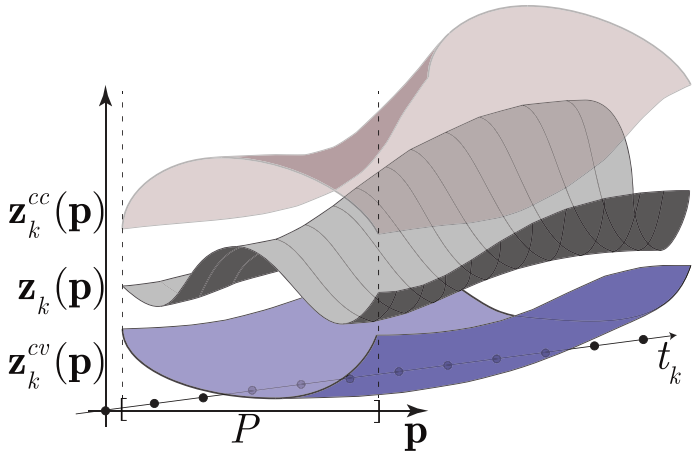
5. Mitsos, A, et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.
6. Scott, JK, et al. **Generalized McCormick relaxations.** *Journal of Global Optimization* 51.4 (2011): 569-606.

Reduced Space Relaxations

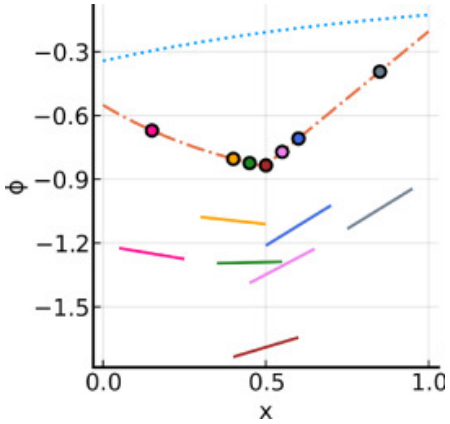
Implicit Functions[7]



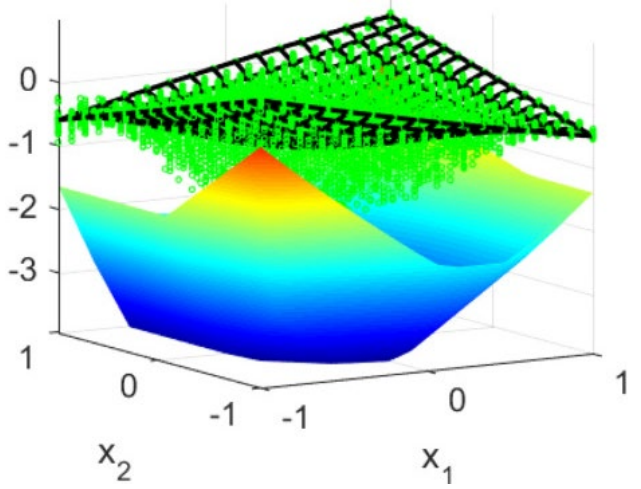
ODEs and DAEs[8]



Continuous Random Variables[9]



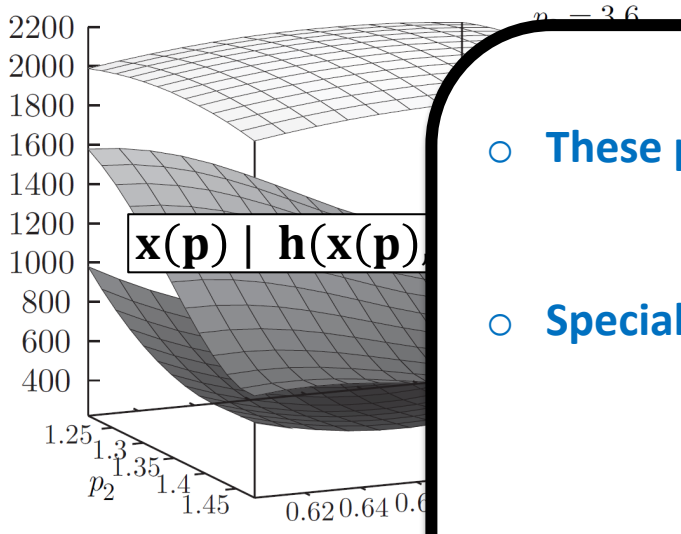
Blackbox Functions[10]



7. Stuber, MD et al. **Convex and concave relaxations of implicit functions.** *Optimization Methods and Software* 30, (2015), 424-460
8. Wilhelm, ME; Le, AV; and Stuber. MD. **Global Optimization of Stiff Dynamical Systems.** *AIChE Journal: Futures Issue,* 65 (12), *2019*
9. Shao, Y and Scott JK. **Convex relaxations for global optimization under uncertainty described by continuous random variables**, *AIChE Journal*, (2018): 3023 – 3033.
10. Song, Y; Cao, H; Mehta, C; and Khan KA. **Bounding Convex Relaxations of Process Models from Below by Tractable Black-Box Sampling**, *Computers & Chemical Engineering*, In Press, (2021).

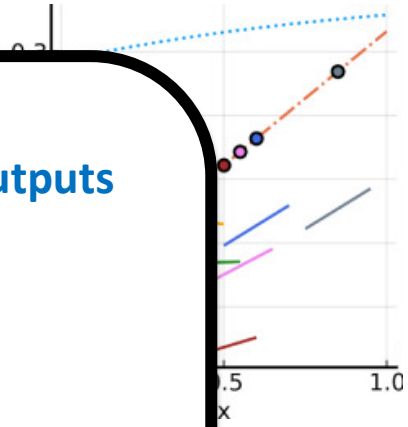
Reduced Space Relaxations

Implicit Functions[7]

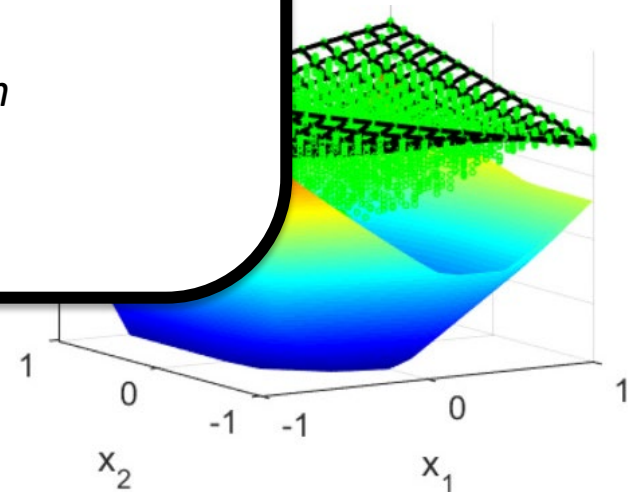


ODEs and DAEs[8]

Continuous Random Variables[9]



x Functions[10]



- **These problem types often involve vector-valued outputs**
- **Specialized relaxation routines required**
  - *Often limiting step in calculations*
  - *Need to extend problem representation*
  - *High performance is beneficial*

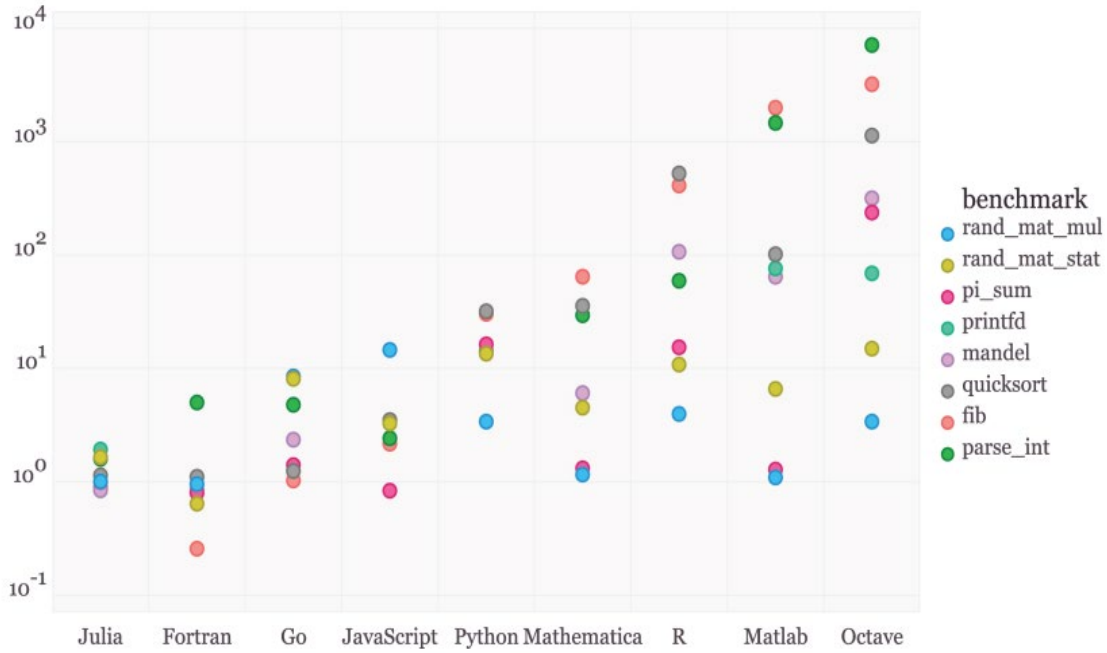
7. Stuber, MD et al. **Convex and** *Software* 30, (2015), 424-460
8. Wilhelm, ME; Le, AV; and Stub *Futures Issue,* 65 (12), *2019*
9. Shao, Y and Scott JK. **Convex relaxa continuous random variables**, *AIChE Journal*, (2018): 3023 – 3033.
10. Song, Y; Cao, H; Mehta, C; and Khan KA. **Bounding Convex Relaxations of Process Models from Below by Tractable Black-Box Sampling**, *Computers & Chemical Engineering*, In Press, (2021).

Language/Solver Capabilities

julia / JUMP

- Performance for low-level routines
- Multiple dispatch & contextual programming allow for ready extensibility (i.e., GPU, parallelism)
- Ease of setup and distribution

- Run-time benchmark comparison[11]



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

11. **Julia: A Fresh Approach to Numerical Computing**. Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah (2017) *SIAM Review*, 59: 65–98.

Original EAGO Capabilities

Extendable Support for Script Defined Functions

Parameter Estimation for Nonideal Two-Liquid Mixture

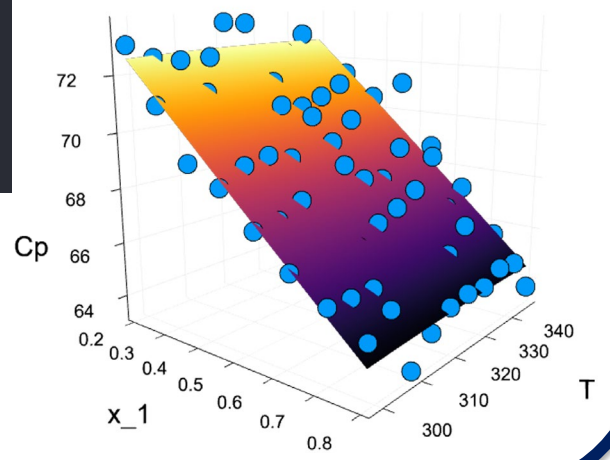
```
exGibbs(T,x1,p) = R*T*(x1*(1-x1)^2*(p[1]*T+p[2]*T^2+p[3]*log(T))+
                       (1-x1)*x1^2*(p[1]*T+p[2]*T^2+p[3]*log(T)))
GibbsA(T) = CpA*(T-T0)-T*CpA*log(T/T0)
GibbsW(T) = CpW*(T-T0)-T*CpW*log(T/T0)
Gibbs(T,x1,p) = x1*GibbsA(T)+(1-x1)*GibbsW(T)+
                R*T*(x1*log(x1)+(1-x1)*log(1-x1))+exGibbs(T,x1,p)
Cp(T,x1,p) = -T*ForwardDiff.derivative(T->ForwardDiff.derivative(T->Gibbs(T,x1,p),T),T)
```

```
function objective(T::Vector,x1::Vector,Cp_exp::Matrix,p...)
    SSE = 0.0
    for i = 1:length(T)
        for j = 1:length(x1)
            SSE += (Cp(T[i],x1[j],p)-Cp_exp[i,j])^2
        end
    end
    return SSE
end
```

```
fobj(p...) = objective(Tdata,x1data,Cp_exp,p...)
```

$$\min_{\mathbf{p}\in P} \sum_i \left(C_P^{mod}(T_i, x_i, \mathbf{p}) - C_P^{exp}(T_i, x_i)\right)$$

$$C_P^{mod}(T_i, x_i, \mathbf{p}) = -T_i \left.\frac{\partial^2 G}{\partial T^2}\right|_P (T_i, x_i, \mathbf{p})$$



Performant Subroutines

- Nonconvex MINLP & SIP solver
- Bounds Tightening Routines:
  - Optimization-based
  - Feasibility-based
- Preprocessing Routines:
  - Algebraic Rearrangements
  - Subexpression Elimination
  - Regular Problem Classification

Language/Solver Capabilities

- ❑ Extendable Graph Representation
- ❑ Dynamic Optimization
- ❑ Machine Learning in EAGO + Julia

New Nonlinear Framework

Old Framework

Evaluator

Directed Tree Structure

Compute & query value functions tied to each attribute:

- *Interval Bounds*
- *Relaxations*
- *Relaxation (Sub)gradients*

New Nonlinear Framework

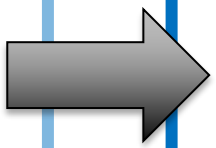
Old Framework

Evaluator

Directed Tree Structure

Compute & query value functions tied to each attribute:

- *Interval Bounds*
- *Relaxations*
- *Relaxation (Sub)gradients*

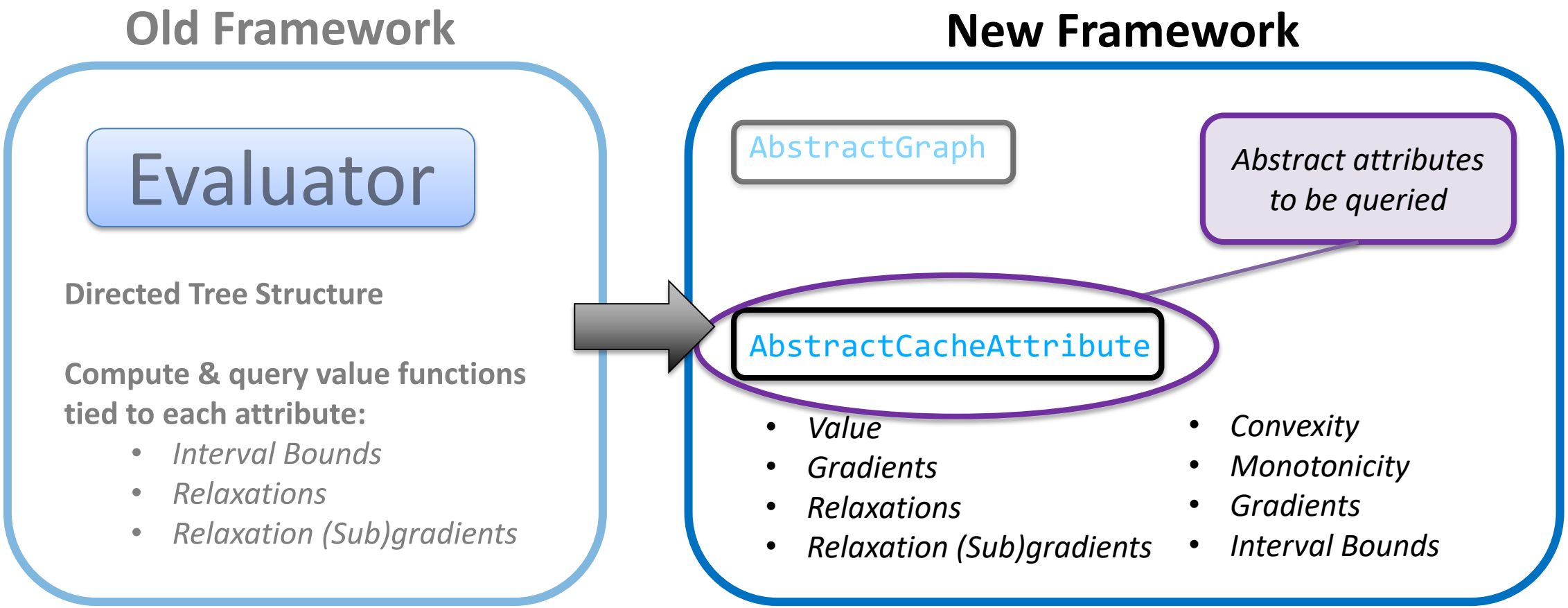


New Framework

`AbstractGraph`

Abstract graph representation of problem

New Nonlinear Framework



New Nonlinear Framework

Old Framework

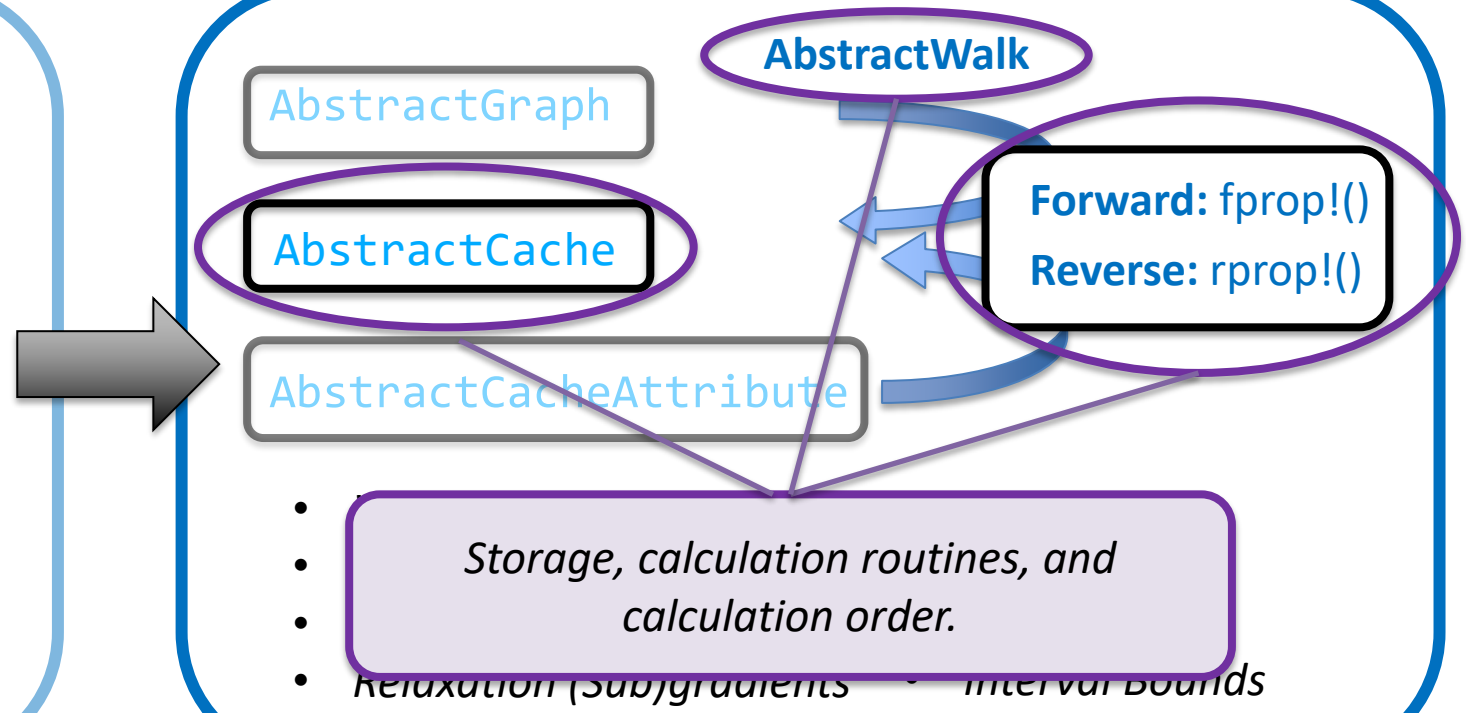
Evaluator

Directed Tree Structure

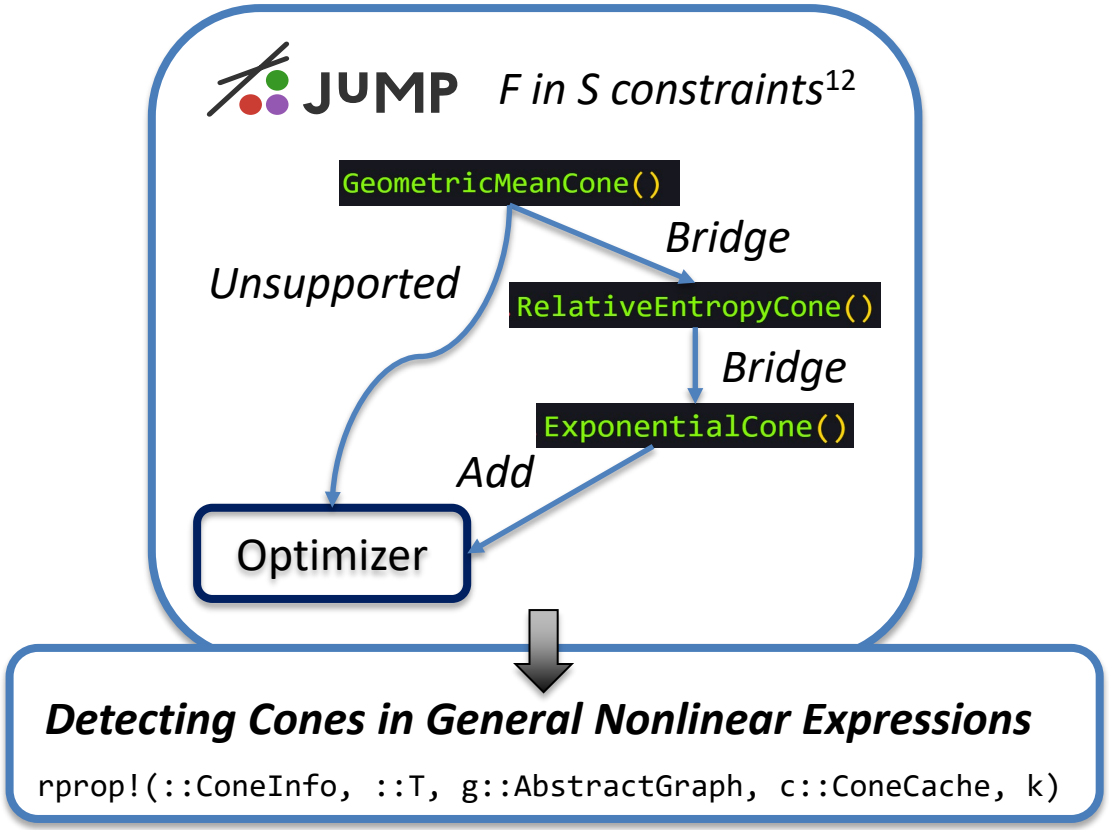
Compute & query value functions tied to each attribute:

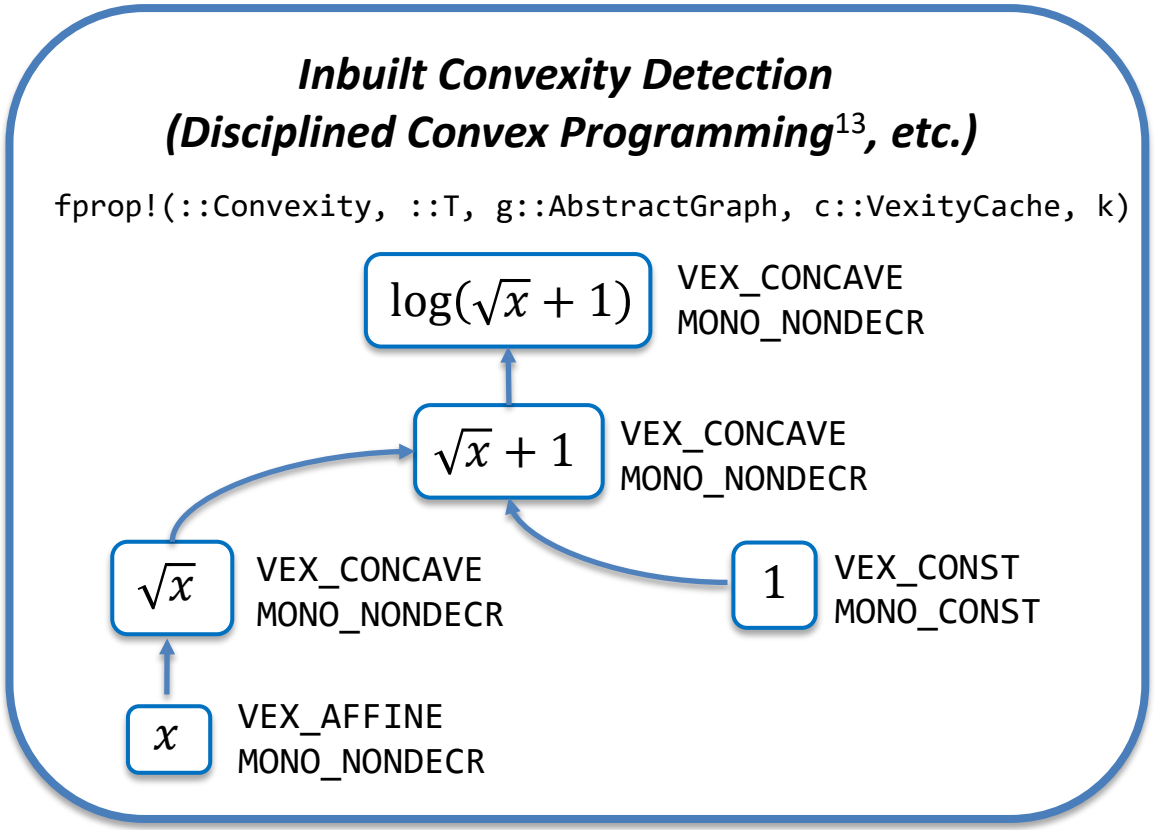
- *Interval Bounds*
- *Relaxations*
- *Relaxation (Sub)gradients*

New Framework



New Graph Utilities





12. Legat, B., Dowson, O., Garcia, J. D., & Lubin, M. (2021). **MathOptInterface: a data structure for mathematical optimization problems.** *INFORMS Journal on Computing*.
13. Grant, M., Boyd, S., & Ye, Y. (2006). **Disciplined convex programming.** In *Global optimization* (pp. 155-210). Springer, Boston, MA.

ML Support + Performance

Build a MLP via UDF (Surrogates.jl)

Build Model

```
xd = sample(n, l, u, SobolSample())
yd = f.(xd)
m = Chain(Dense(2,6,tanh), Dense(6,1))
ns = NeuralSurrogate(xd, yd, l, u, model = m)
surrogate_optimize(f, SRBF(), l, u, ns, SobolSample())
```

Optimize Model

```
using JuMP, EAGO

m = Model(EAGO.Optimizer)
@variable(m, l[i] <= x[i=1:2] <= u[i])
register(m, :surrogate, 2, ns)
@NLobjective(m, surrogate(x[1],x[2]))
optimize!(m)
```

Method/Context Overloading Extension for Specialized

```
function (a::Dense)(x::AbstractVecOrMat)
    W, b, σ = a.weight, a.bias, a.σ
    return σ.(W*x .+ b)
end


function (a::Dense)(mcb::MCBox{N,T}) where {N,T}
    W, b, σ = a.weight, a.bias, a.σ
    n = length(mcb)
    yp = zeros(MC{N,T}, n)
    yn = zeros(MC{N,T}, n)
    for i = 1:n
        for j = 1:n
            Wij = @inbounds W[i,j]
            if Wij > 0.0
                yp[i] += Wij*mcb.x[j]
            else
                yn[i] += Wij*mcb.x[j]
            end
        end
        yp[i] = set_value_post(yp[i], x.v)
        yn[i] = set_value_post(yn[i], x.v)
    end
    @__dot__ yp += yn + b
    return set_value_post(σ(yp), x.v)
end
```

ML Support + Performance

Build a MLP via UDF (Surrogates.jl)

Build Model

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

```
m = Chain(Dense(2,6,tanh), Dense(6,1))
```

```
ns = NeuralSurrogate(xd, yd, l, u, model = m)
surrogate_optimize(f, SRBF(), l, u, ns, SobolSample())
```

Optimize Model

```
using JuMP, EAGO

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

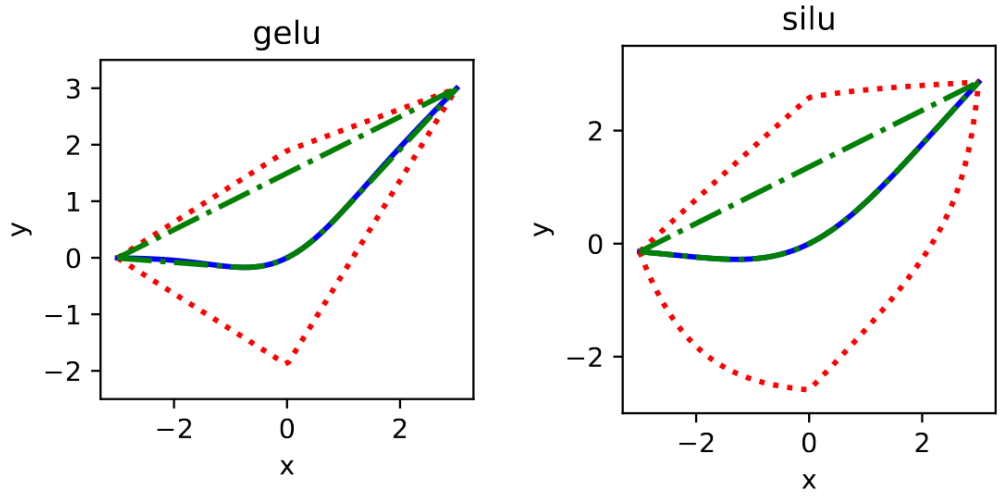
```
register(m, :surrogate, 2, ns)
@NLobjective(m, surrogate(x[1],x[2]))
```

```
optimize!(m)
```

Additional Support for ML Specific Functions

```
function (a::Dense)(x::AbstractVecOrMat)
    W, b, σ = a.weight, a.bias, a.σ
    return σ.(W*x .+ b)
end
```

Envelopes of Relaxation Functions




```
end
```

EAGO.jl and Dynamics

Abstract Layer

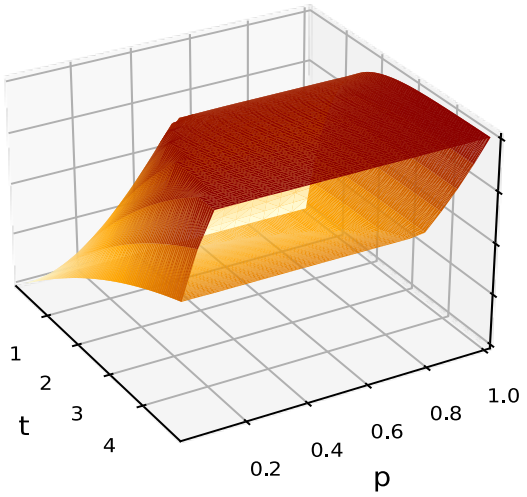
DynamicBounds.jl[14]

Extendable Global Optimizer[15]

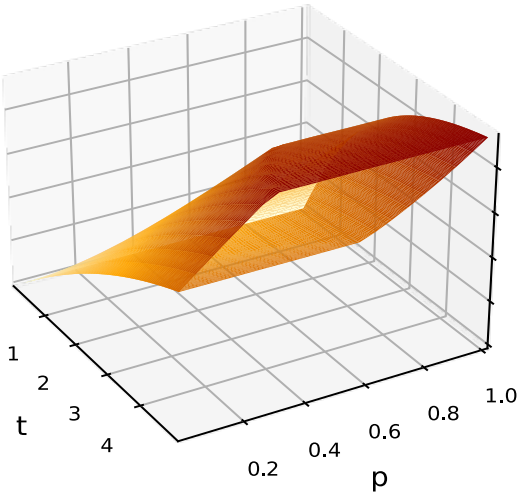
Simple ODE Relaxation

$$\frac{dx}{dt} = \exp(p)\sin(x)(2-x),$$
$$x(0) = 1, \quad p \in [0.01,1], \quad t \in [0,5]$$

Standard McCormick Relaxation



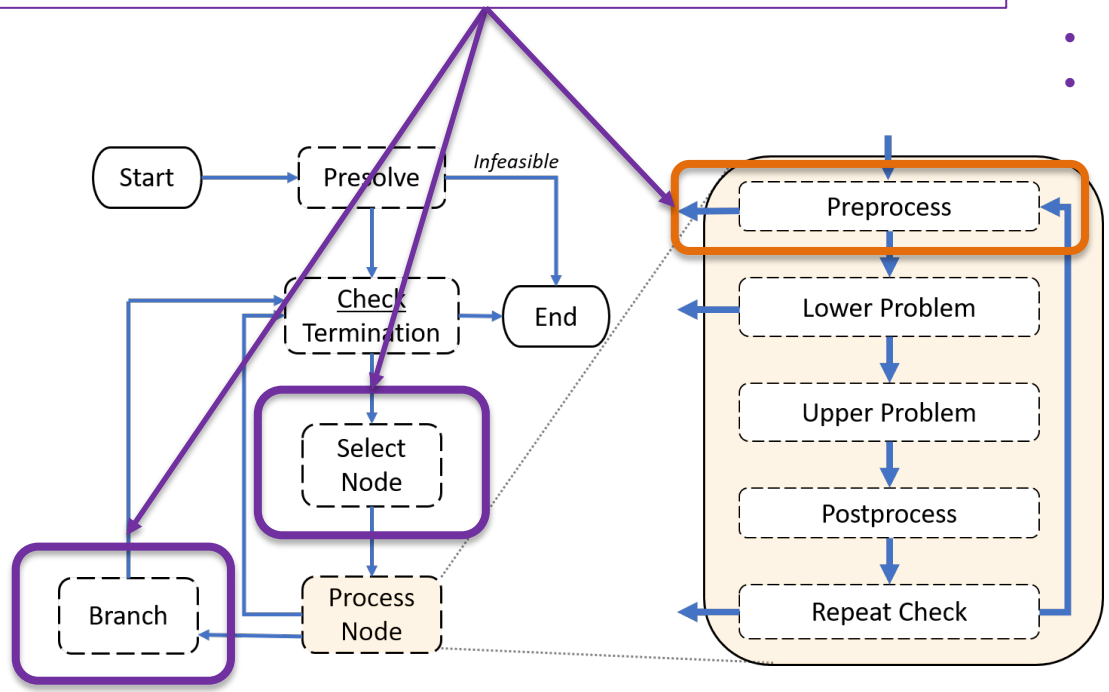
Improved Trilinear Term



14. Wilhelm, M. E., **DynamicBounds.jl**, (2020), GitHub repository, https://github.com/PSORLab/DynamicBounds.jl
15. Wilhelm, M. E., and M. D. Stuber. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software* (2020): 1-26.

Improving EAGO with Machine Learning

Typically done by theoretically motivated heuristic



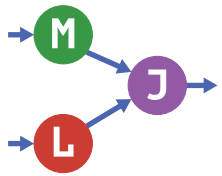
- Introduce auxiliary variable for subexpression?
- Select branch point or node?
- Use LP, SOCP, Convex relaxation?

EAGO.jl*: Extendable Low-Level Routines*

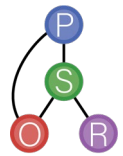
MINLPLib.jl*: Extendable Low-Level Routines*

Evolutionary.jl: *Genetic Programming*

MLJ.jl: *Random Forest Regression*
+ Interpretable Tools



3. Wilhelm, M.E., and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 1-26.
16. Nagarajan, H. **MINLPLib.jl**, (2020), GitHub repository, https://github.com/lanl-ansi/MINLPLib.jl
17. Wild, A. **Evolutionary.jl**, (2014), GitHub repository, https://github.com/wildart/Evolutionary.jl
18. Bloam, A.D. et al. **MLJ: A Julia package for composable machine learning.** *Journal of Open Source Software*, 5 (55), p2704.



Conclusion

EAGO - an extensible deterministic global optimizer

- ❖ Architected for user-defined functions and routines
- ❖ High performance solver
- ❖ Open-source and free for non-commercial use

Future Outlook

- ❖ Parallel computing capability
- ❖ Further extension to domain specific Julia packages
- ❖ Performance improvements in core EAGO algorithms

Acknowledgements

Members of the process systems and operations research laboratory at the University of Connecticut (https://psor.uconn.edu/)

Funding:

National Science Foundation, Award 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.

Questions?

Process Systems and Operations Research Laboratory

https://www.psor.uconn.edu

EAGO

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