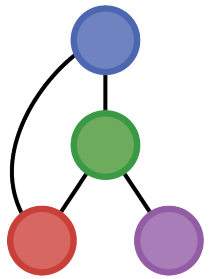
UCONN
UNIVERSITY OF CONNECTICUT

Global Dynamic Optimization Using Hardware-Accelerated Programming

Robert Gottlieb, PhD Student
Matthew Stuber, Assistant Professor

November 13th, 2022

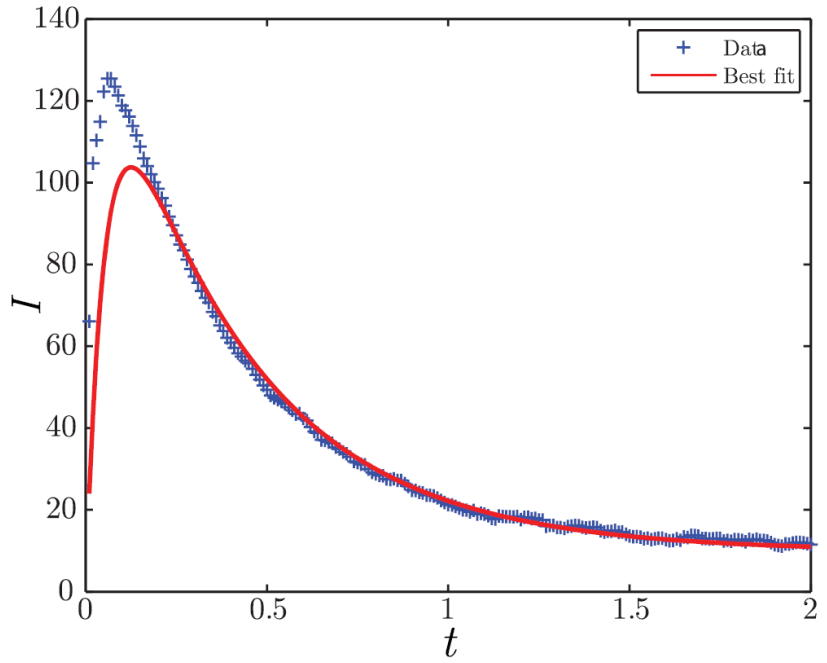
2022 AIChE ANNUAL MEETING



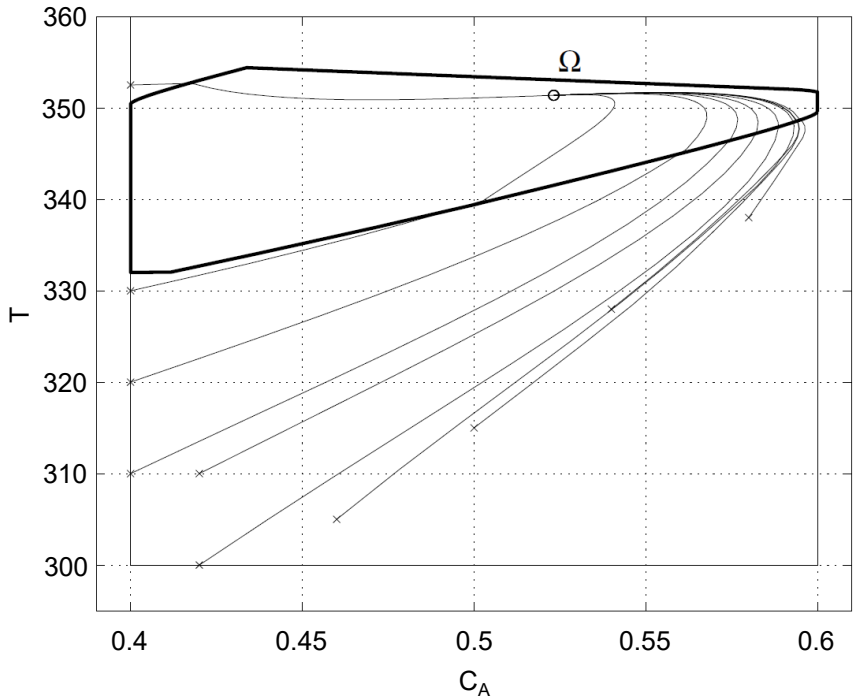
Process Systems and Operations Research Laboratory

Importance of Global Dynamic Optimization

Parameter Estimation and Model Validation[1]



Robust MPC[2]

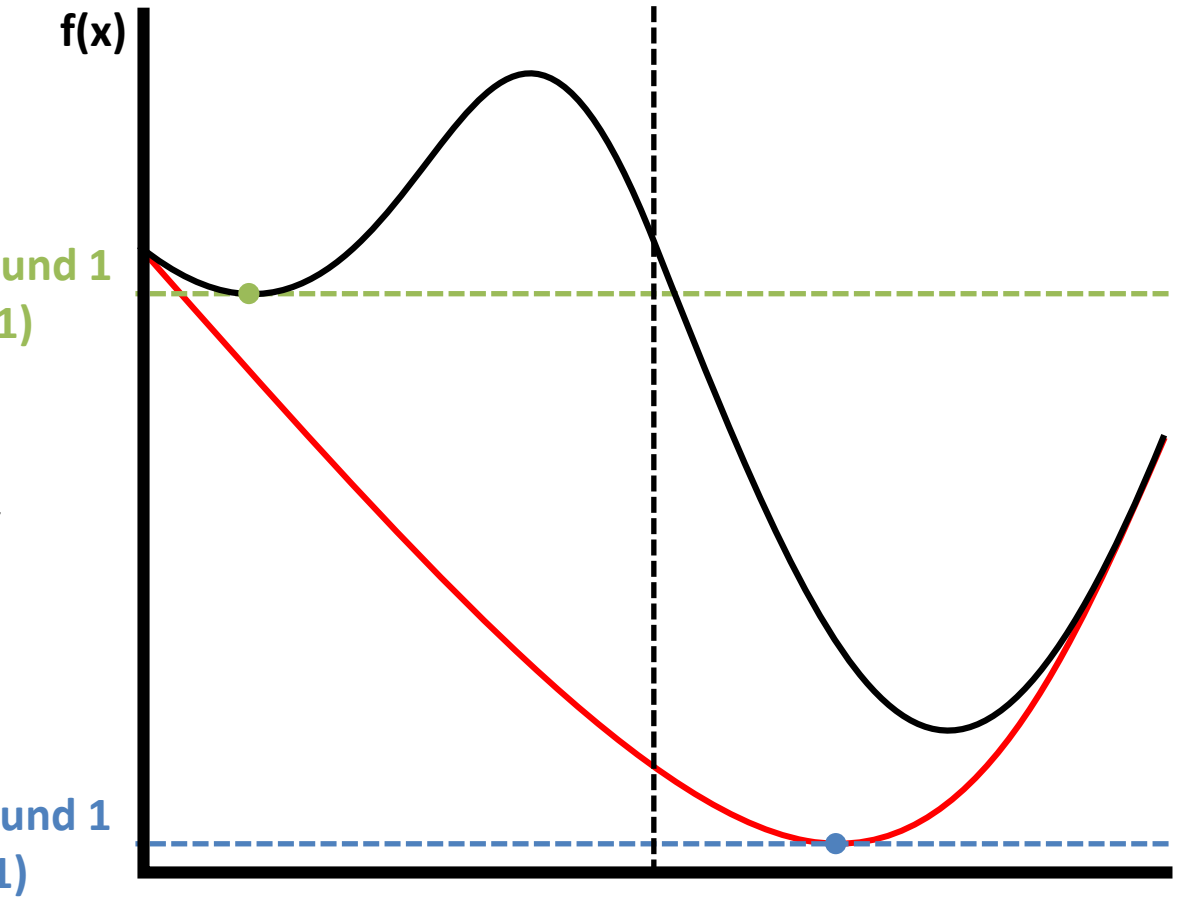


1. Stuber, M.D. et al. **Convex and concave relaxations of implicit functions**. *Optimization Methods and Software* **30**(3), 424-460 (2014).
2. Limon, D. et al. **Robust MPC of constrained nonlinear systems based on interval arithmetic.** *IEEE Proceedings – Control Theory and Applications* **152**(3), 325-332 (2005).

Deterministic Global Optimization

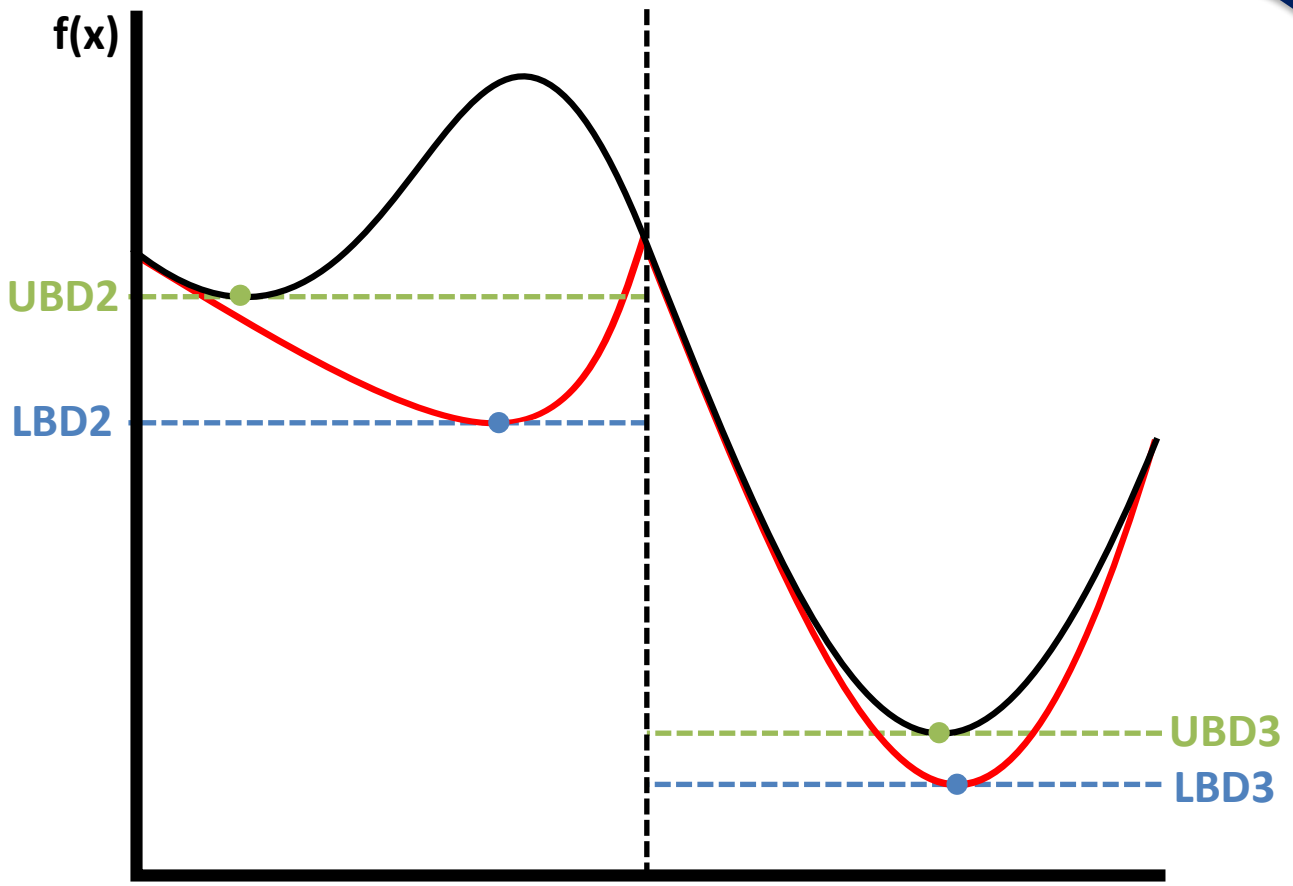
Branch and Bound (B&B)

- Deterministic method of obtaining a function's lower bound on its entire domain



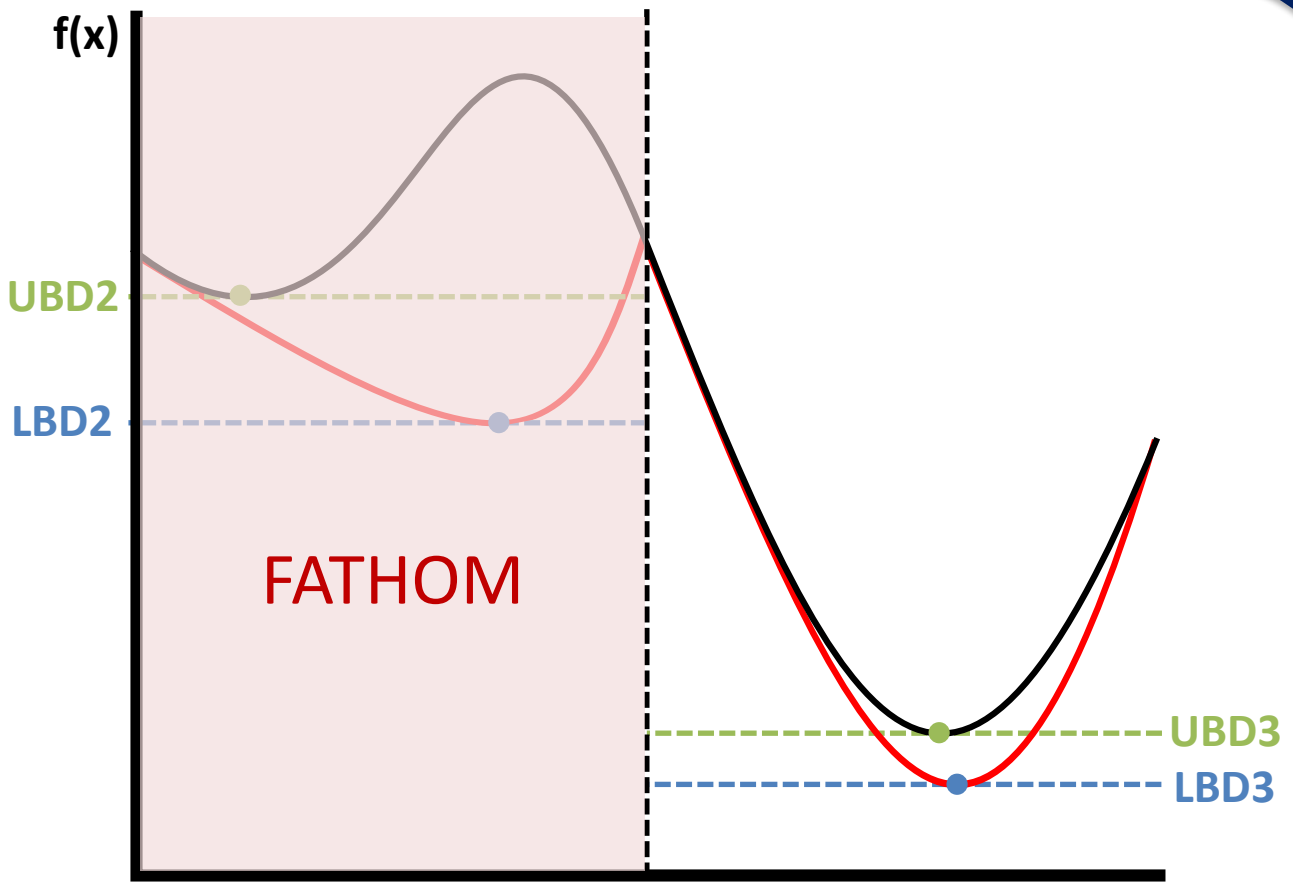
Deterministic Global Optimization

- New B&B **nodes** are created by **branching**
- Nodes are checked sequentially to calculate their LBD/UBD



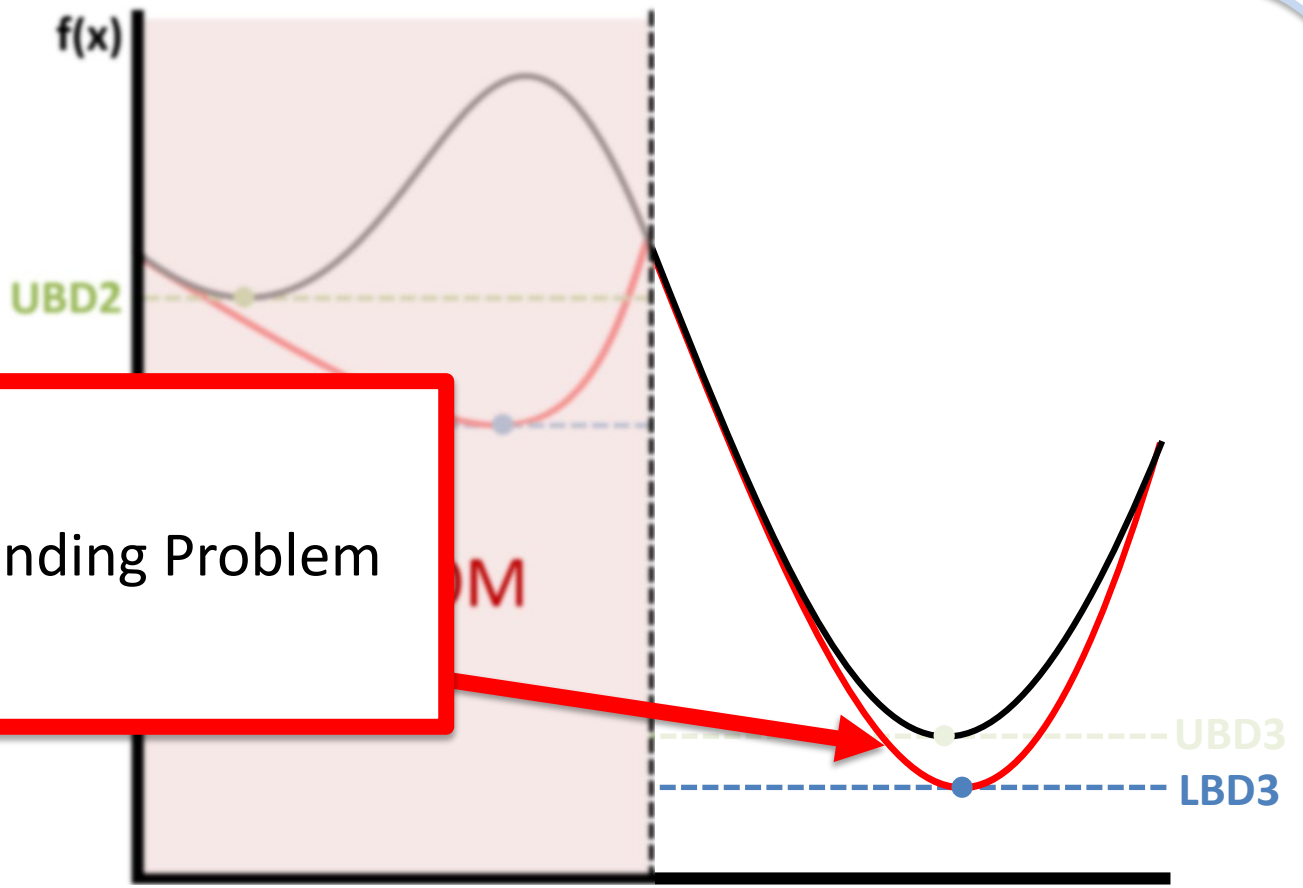
Deterministic Global Optimization

- New B&B **nodes** are created by **branching**
- Nodes are checked sequentially to calculate their LBD/UBD



Deterministic Global Optimization

- New B&B **nodes** are created by
- Nodes are sequentiall their LBD/U



Lower Bounding Problem

EAGO.jl

Deterministic global optimizer

- High performance
- Open-source and free for non-commercial use
- Extensible

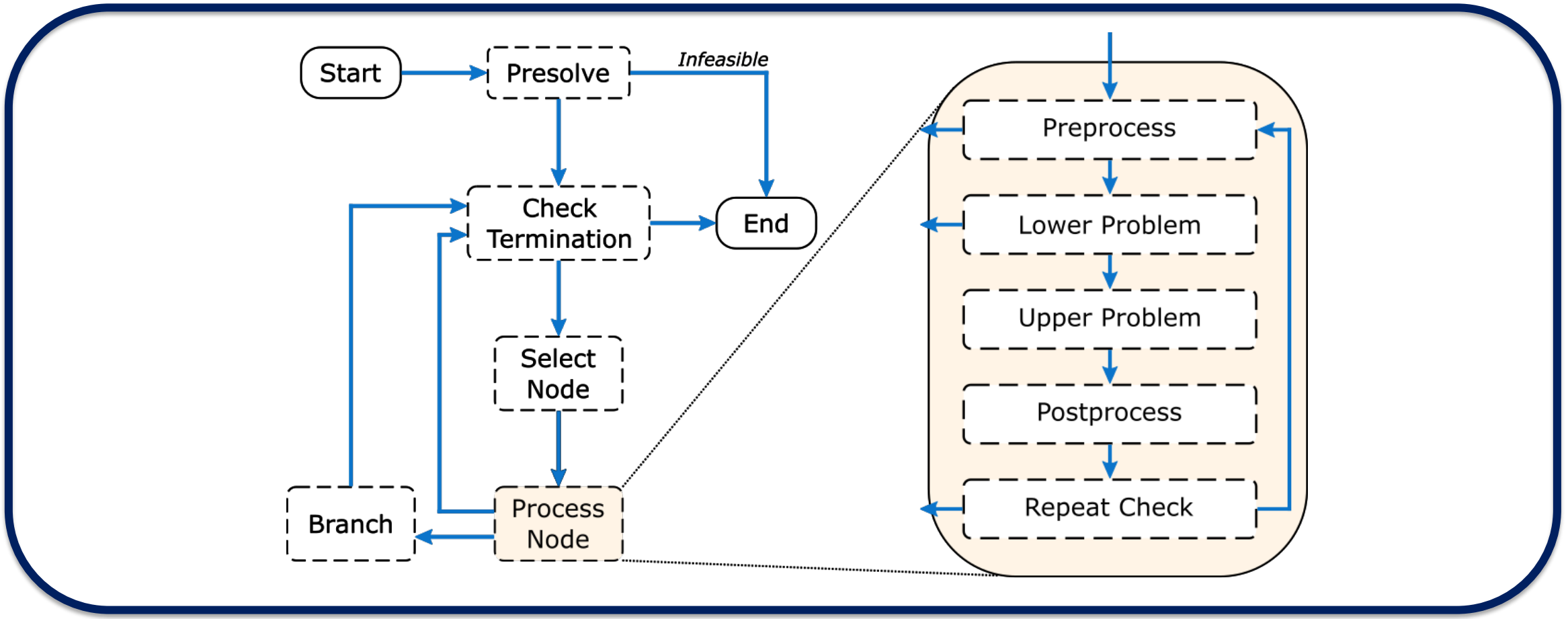
https://www.github.com/PSORLab/EAGO.jl

To learn more:

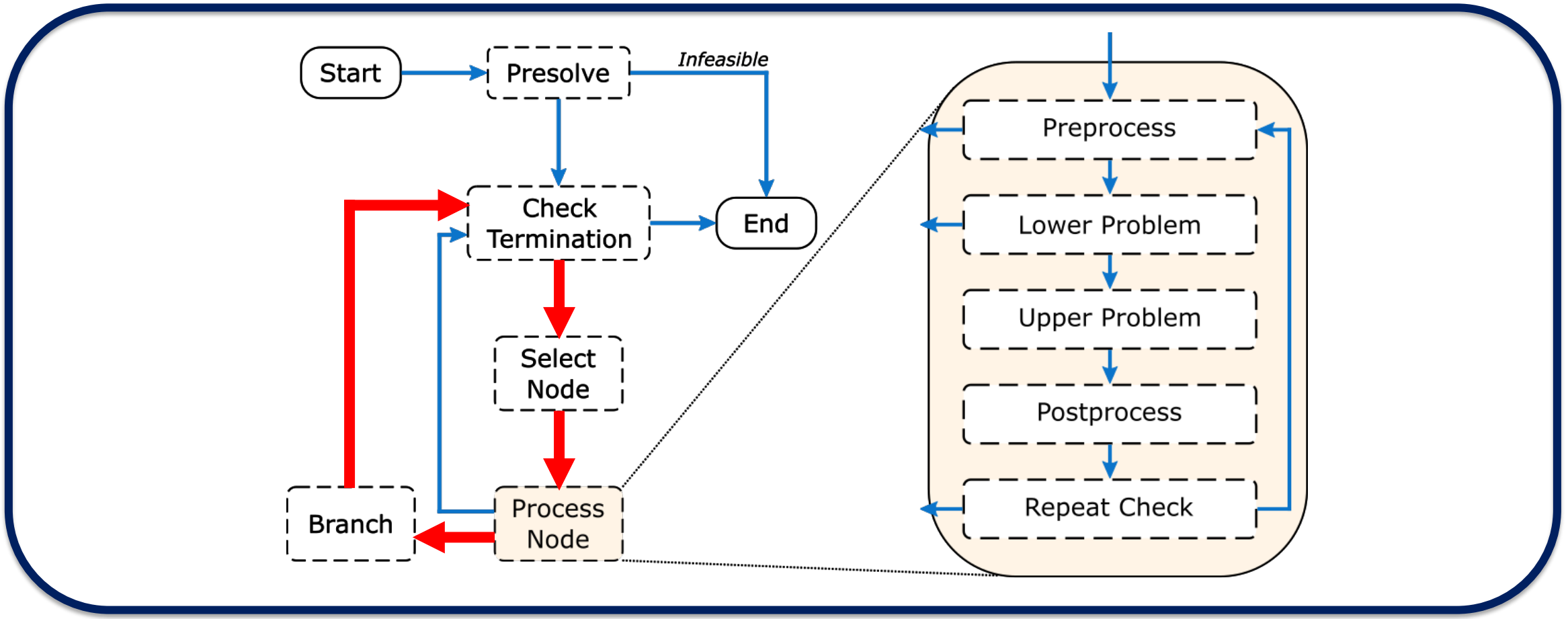
- Visit our GitHub!
- Attend my presentation!

Recent Developments in EAGO.jl (Easy Advanced Global Optimization in Julia)
Tuesday, Nov 15, 10:06 AM
W-101A

Branch-and-Bound in EAGO



Branch-and-Bound in EAGO



Motivation: Hardware Acceleration

All B&B Implementations

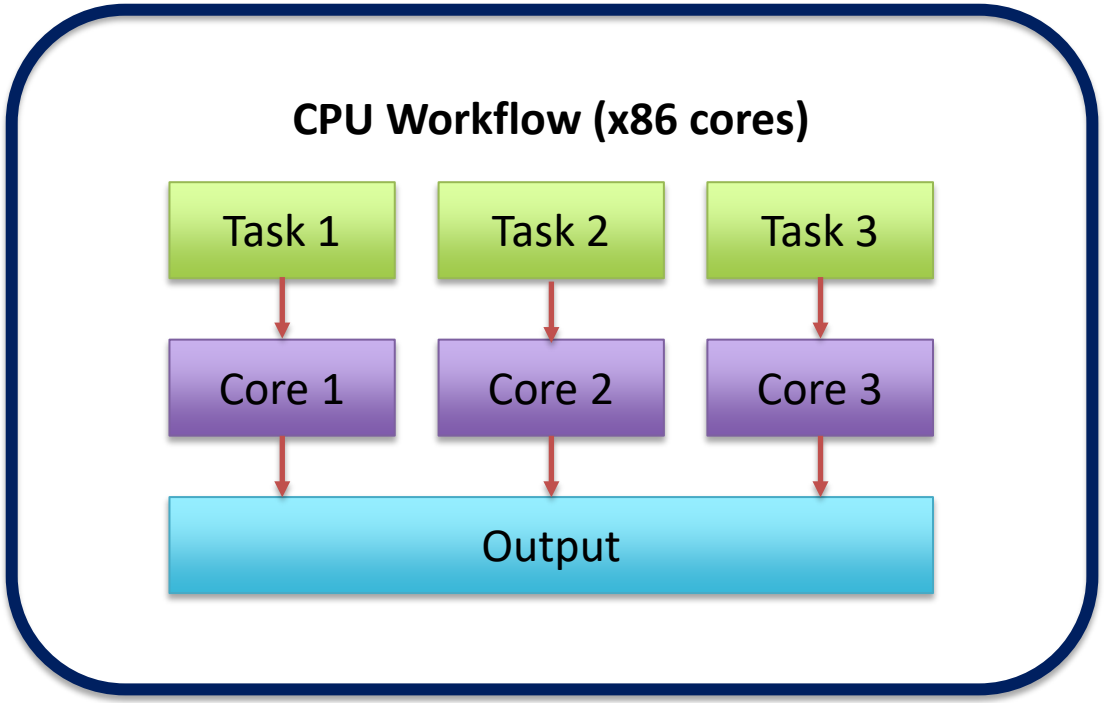
- B&B nodes checked one-at-a-time
- Calculations performed on a CPU
- **Bottleneck:** Optimization speed tied to CPU hardware advances[3]

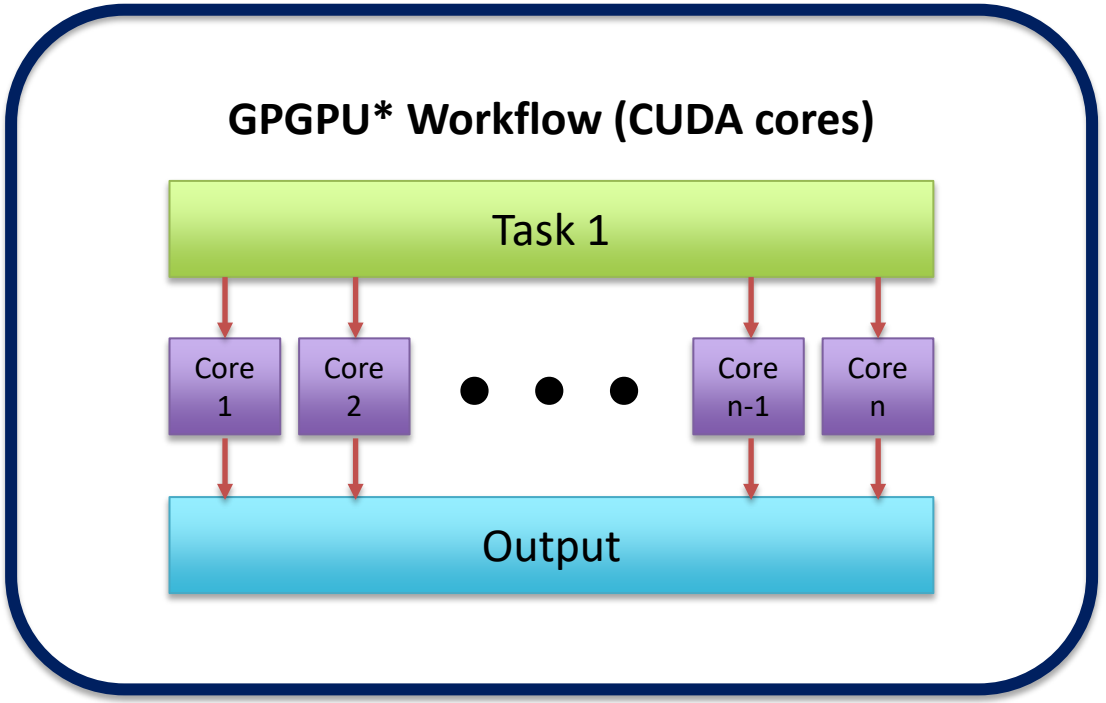
Unexplored Frontier

- Many B&B nodes checked in parallel
- Calculations performed on a GPU
- Potential to outpace CPU hardware

3. Koch, T. **Progress in mathematical programming solvers from 2001 to 2020.** Presentation at EURO 2022, Espoo, Finland (2022).

GPU Refresher





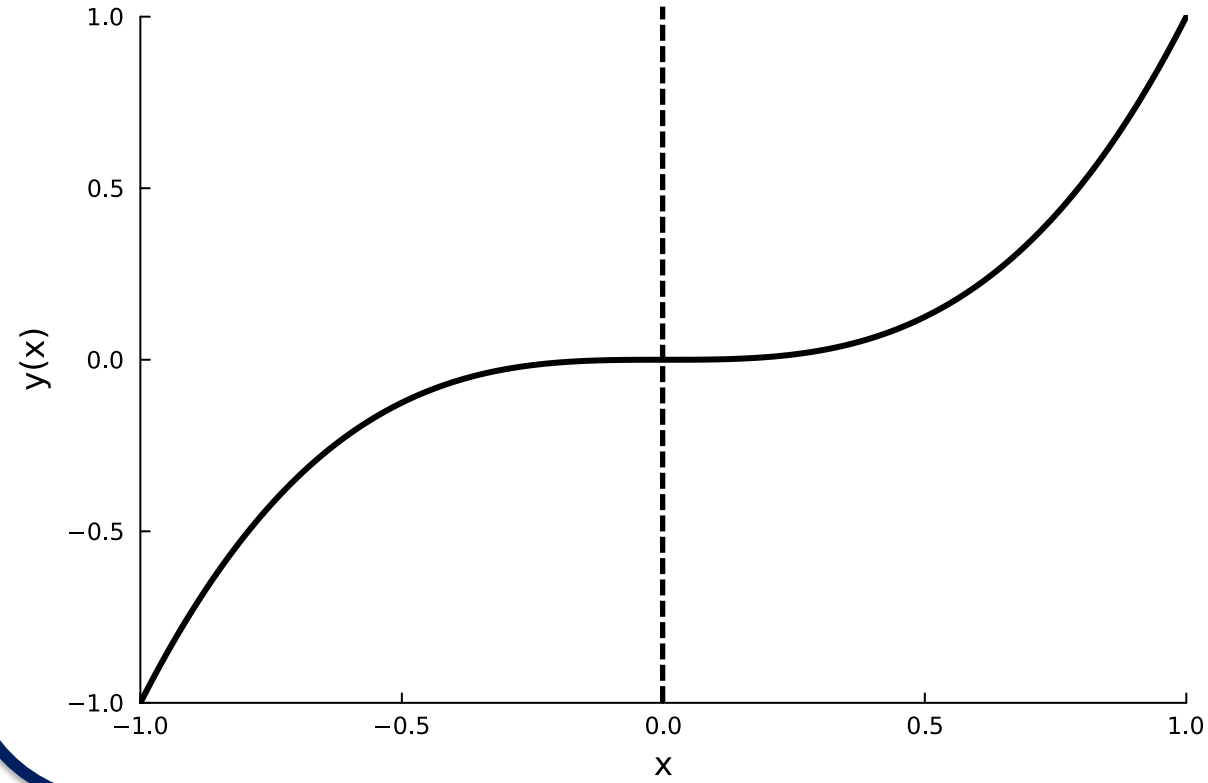
*General Purpose Graphics Processing Unit

Challenge 1) Bounds Dependence

- EAGO uses McCormick relaxations
- Structure of relaxations depend on variable domain[4,5]

Example:

$$y(x) = x^3$$



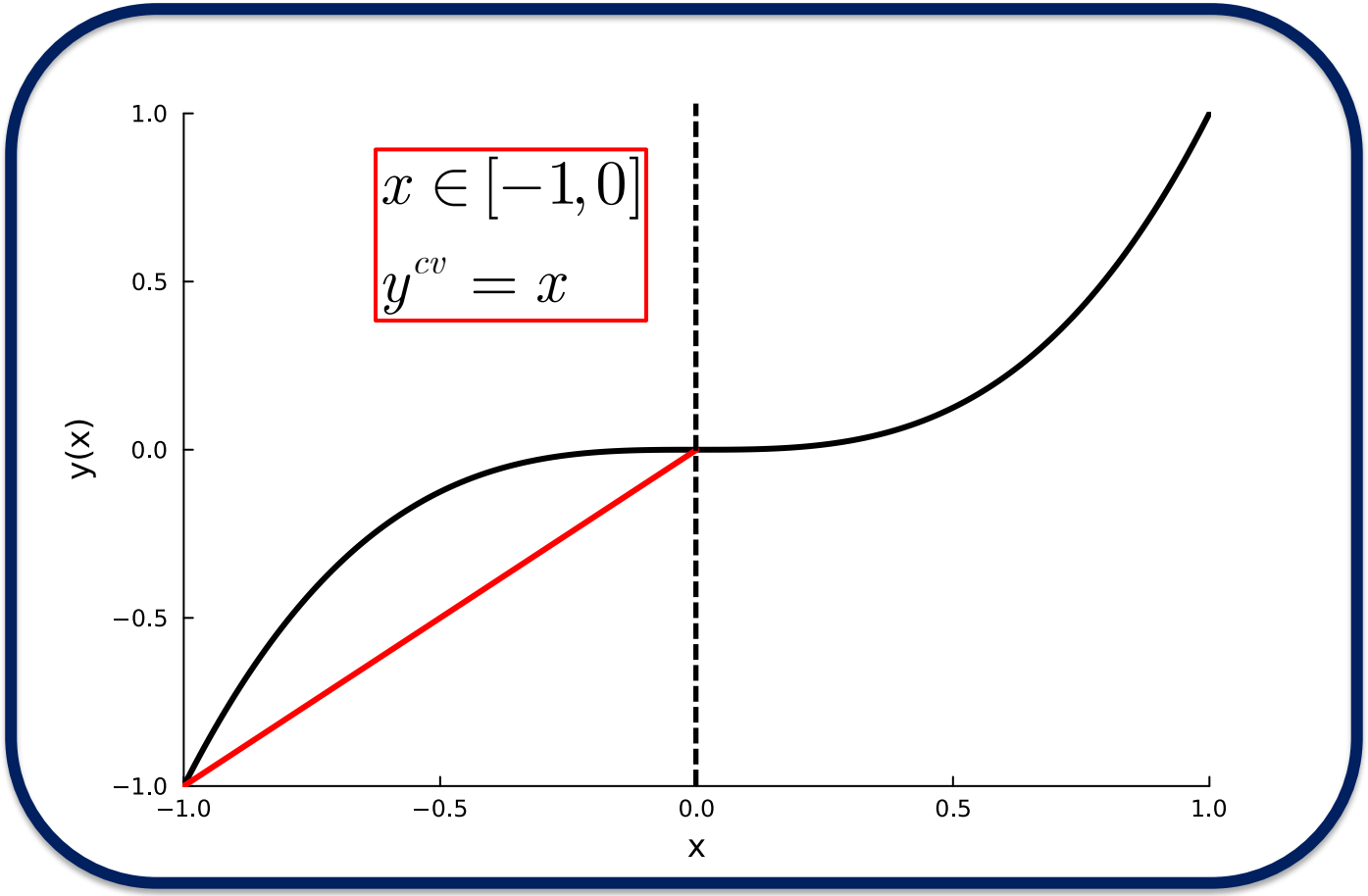
4. Mitsos, A., et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.
5. Scott, J. K., et al. **Generalized McCormick relaxations.** *Journal of Global Optimization* 51.4 (2011): 569-606.

Challenge 1) Bounds Dependence

- EAGO uses McCormick relaxations
- Structure of relaxations depend on variable domain[4,5]

Example:

$$y(x) = x^3$$



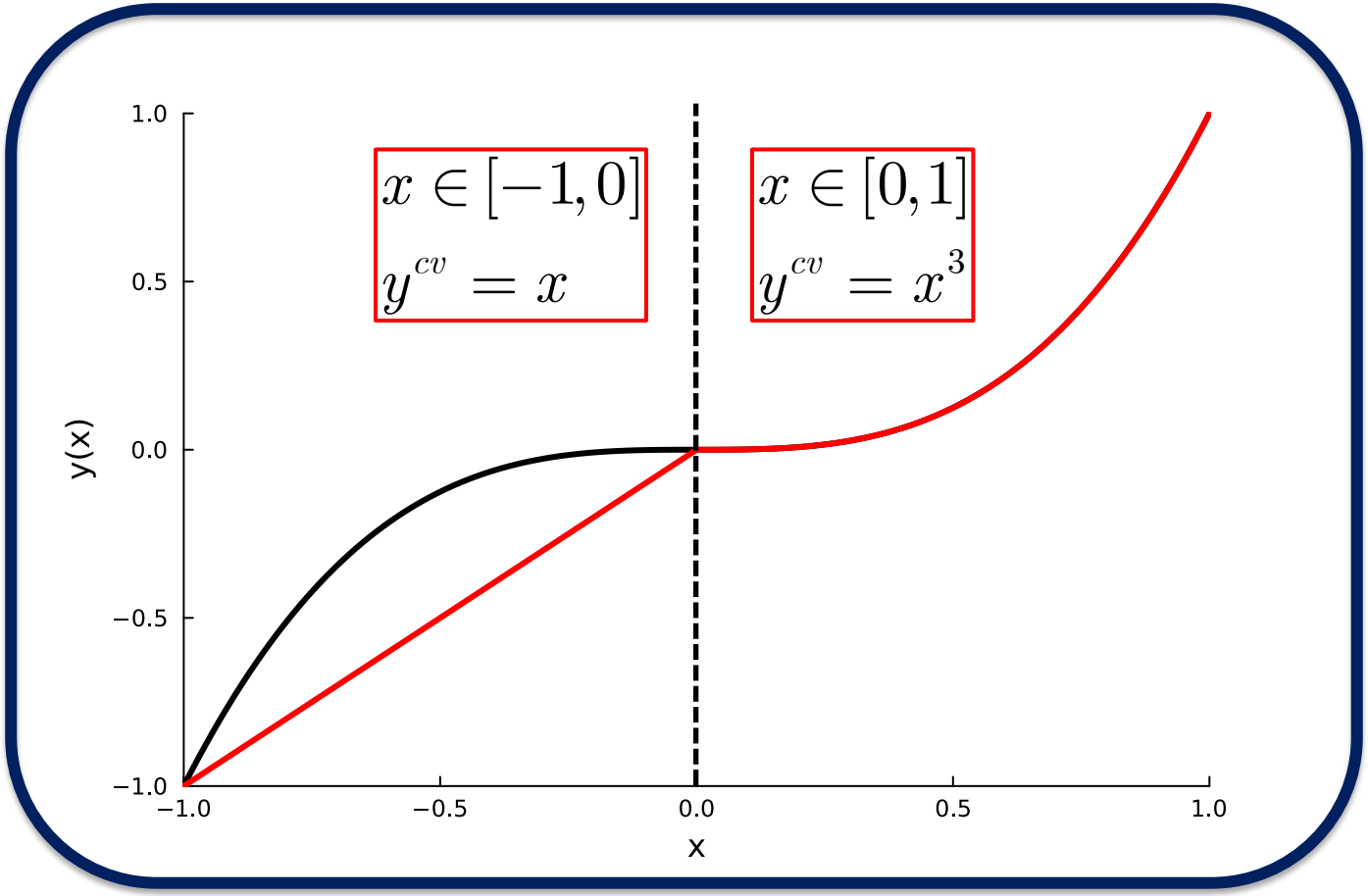
4. Mitsos, A., et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.
5. Scott, J. K., et al. **Generalized McCormick relaxations.** *Journal of Global Optimization* 51.4 (2011): 569-606.

Challenge 1) Bounds Dependence

- EAGO uses McCormick relaxations
- Structure of relaxations depend on variable domain[4,5]

Example:

$$y(x) = x^3$$



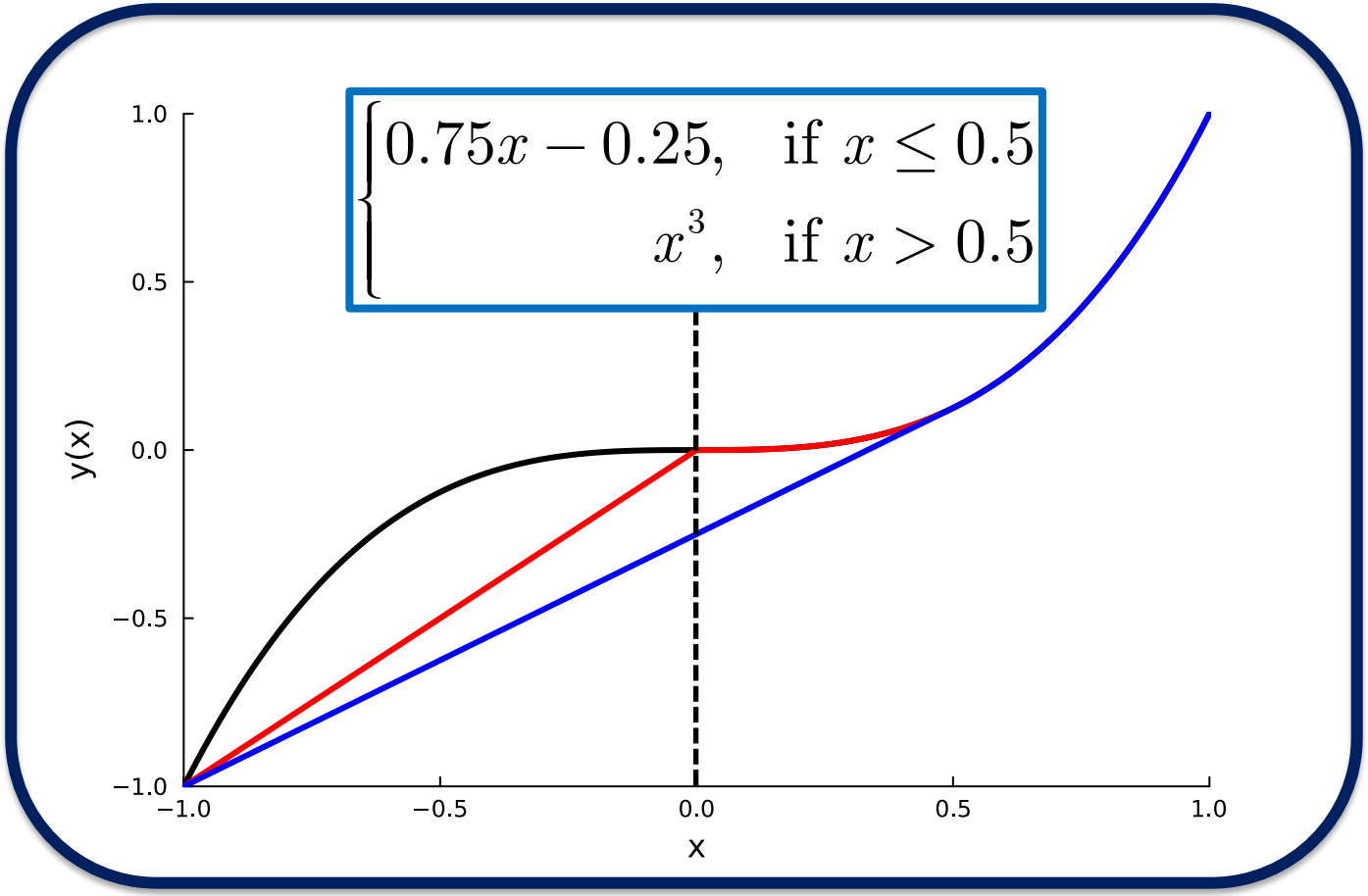
4. Mitsos, A., et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.
5. Scott, J. K., et al. **Generalized McCormick relaxations.** *Journal of Global Optimization* 51.4 (2011): 569-606.

Challenge 1) Bounds Dependence

- EAGO uses McCormick relaxations
- Structure of relaxations depend on variable domain[4,5]

Example:

$$y(x) = x^3$$

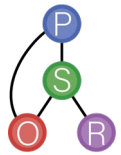


4. Mitsos, A., et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.
5. Scott, J. K., et al. **Generalized McCormick relaxations.** *Journal of Global Optimization* 51.4 (2011): 569-606.

McCormick.jl in EAGO

Step 1) Create a “McCormick object” for a variable on a domain X and at a point $x \in X$

```
xMC = MC{1,NS}(1.0, Interval(-2.0, 5.0), 1)
```

McCormick.jl in EAGO

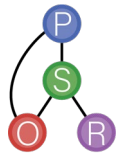
Step 1) Create a “McCormick object” for a variable on a domain X and at a point $x \in X$

```
xMC = MC{1,NS}(1.0, Interval(-2.0, 5.0), 1)
```

Step 2) Use the McCormick object in your desired expression

```
f(x) = (x-2)/(x+3)
```

```
julia> f(xMC)
MC{1, NS}(-2.125, 1.375, [-4, 3], [0.625], [-0.25], false)
```

McCormick.jl in EAGO

Step 1) Create a "McCormick object" for a variable on a domain X and at a point $x \in X$

```
xMC = MC{1,NS}(1.0, Interval(-2.0, 5.0), 1)
```

Step 2) Use the McCormick object in your desired expression

```
f(x) = (x-2)/(x+3)
```

```
julia> f(xMC)
MC{1, NS}(-2.125, 1.375, [-4, 3], [0.625], [-0.25], false)
```

Value of f^{cv} at $x = 1.0$ where $X = [-2.0, 5.0]$

McCormick.jl in EAGO

Step 1) Create a "McCormick object" for a variable on a domain X and at a point $x \in X$

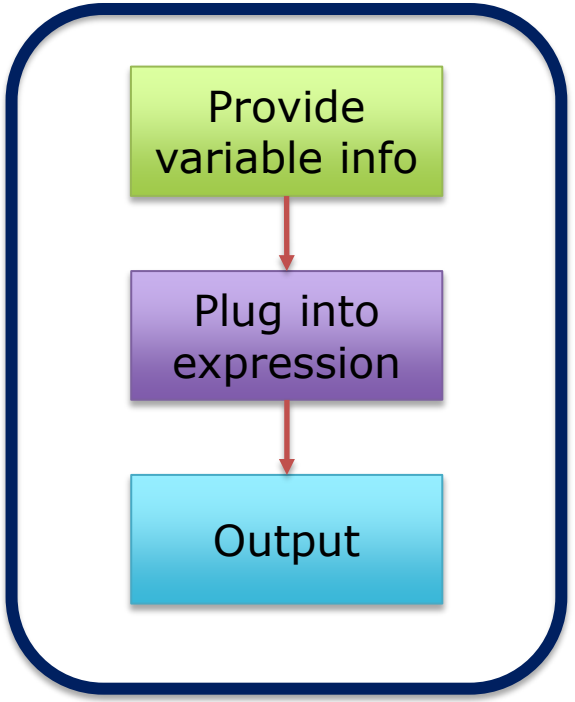
```
xMC = MC{1,NS}(1.0, Interval(-2.0, 5.0), 1)
```

Step 2) Use the McCormick object in your desired expression

```
f(x) = (x-2)/(x+3)
```

```
julia> f(xMC)
MC{1, NS}(-2.125, 1.375, [-4, 3], [0.625], [-0.25], false)
```

Value of f^{cv} at $x = 1.0$ where $X = [-2.0, 5.0]$



McCormick.jl in EAGO

Step 1) Create a "McCormick object" for a variable on a domain X and at a point $x \in X$

```
xMC = MC{1,NS}(1.0, Interval(-2.0, 5.0), 1)
```

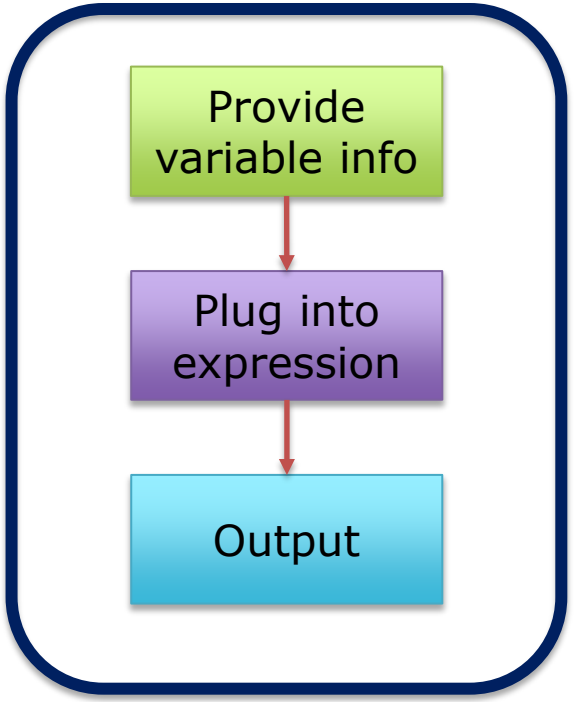
Step 2) Use the McCorm[illegible]xpression

```
f(x) = (x-2)/(x+3)
```

```
julia> f(xMC)
MC{1, NS}(-2.125, 1.375, [-4, 3], [0.625], [-0.25], false)
```

Value of f^{cv} at $x = 1.0$ where $X = [-2.0, 5.0]$

Calculations applied to *xMC* were dependent on *xMC*'s bounds!



Solution to 1) SourceCodeMcCormick.jl

SourceCodeMcCormick.jl

$$f(x) \longrightarrow f^L(x),\ f^U(x),\ f^{cv}(x),\ f^{cc}(x)$$

- Applies McCormick arithmetic using symbolic substitution with Symbolics.jl[6]
- Creates a general form of the new expressions
  - Accounts for all possible variable bounds with a single expression

6. Gowda, S., et al. **High-performance symbolic-numerics via multiple dispatch.** arXiv:2105.03949 (2021).

SourceCodeMcCormick.jl

Step 1) Create an evaluator function for your desired expression

```
@variables x
f_cv, _ = convex_evaluator((x-2)/(x+3))
```

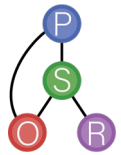
SourceCodeMcCormick.jl

Step 1) Create an evaluator function for your desired expression

```
@variables x
f_cv, _ = convex_evaluator((x-2)/(x+3))
```

Step 2) Plug in a point $x \in X$ and a domain X

```
julia> f_cv(1.0, 1.0, 5.0, -2.0)
-2.125
```

SourceCodeMcCormick.jl

Step 1) Create an evaluator function for your desired expression

```
@variables x
f_cv, _ = convex_evaluator((x-2)/(x+3))
```

Step 2) Plug in a point $x \in X$ and a domain X

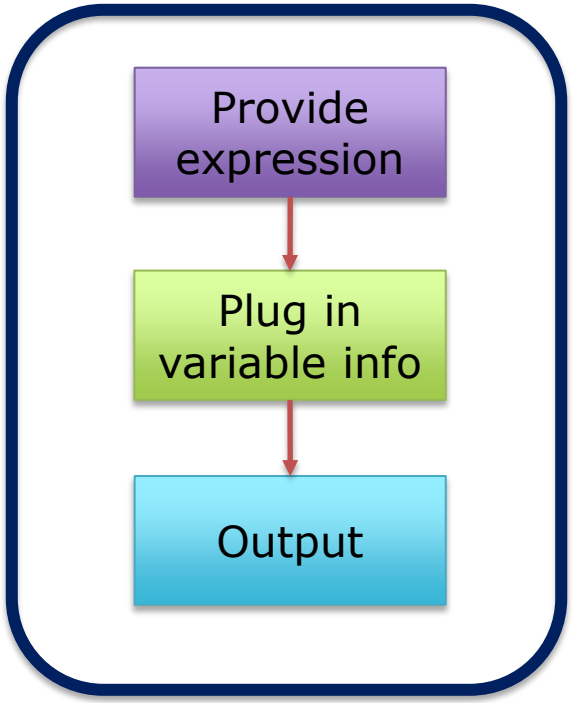
```
julia> f_cv(1.0, 1.0, 5.0, -2.0)
-2.125
```

Value of f^{cv} at $x = 1.0$ where $X = [-2.0, 5.0]$

SourceCodeMcCormick.jl

Step 1) Create an evaluator function for your desired expression

```
@variables x
f_cv, _ = convex_evaluator((x-2)/(x+3))
```



Step 2) Plug in a point $x \in X$ and a domain X

```
julia> f_cv(1.0, 1.0, 5.0, -2.0)
-2.125
```

Value of f^{cv} at $x = 1.0$ where $X = [-2.0, 5.0]$

SourceCodeMcCormick.jl

Step 1) Create an evaluator function for your desired expression

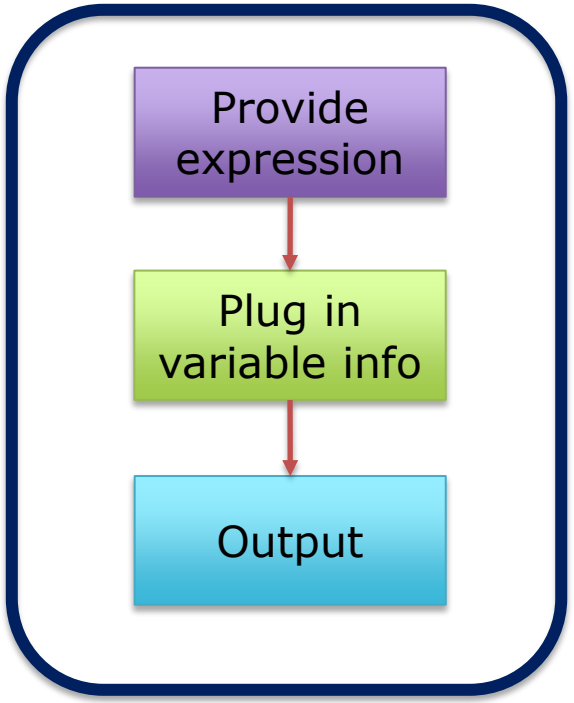
```
@variables x
f_cv, _ = convex_evaluator((x-2)/(x+3))
```

Step 2) Plug in a point $x \in X$ ar

```
julia> f_cv(1.0, 1.0, 5.0, -2.0)
-2.125
```

Value of f^{cv} at $x = 1.0$ where $X = [-2.0, 5.0]$

Calculations are the same regardless of *x*'s domain!



Challenge 2) Parallel Convex Optimization

Given a convex function, how do you find its minimum value?

Variety of Methods

Newton's Method

Interior Point Method

Cutting Plane Method

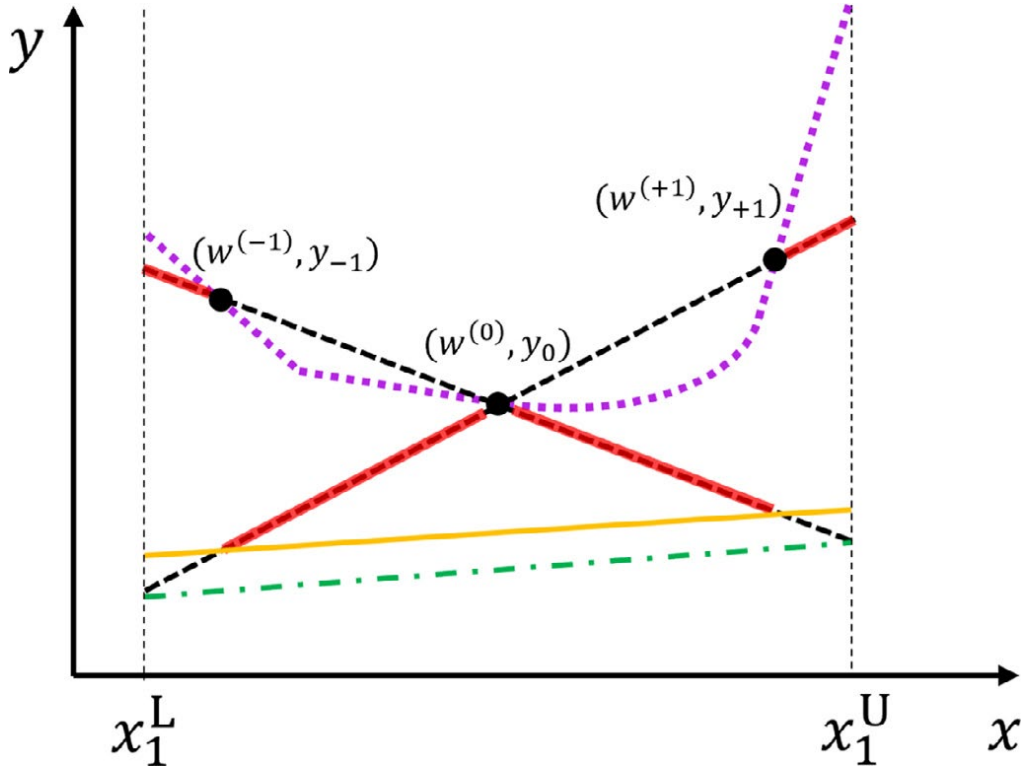
[…]

Variable number of steps
(Bad for parallelization)

Solution to 2) Black-box Sampling

Recent paper by Song et al.[7] introduced a method of obtaining a guaranteed lower bound for a convex function

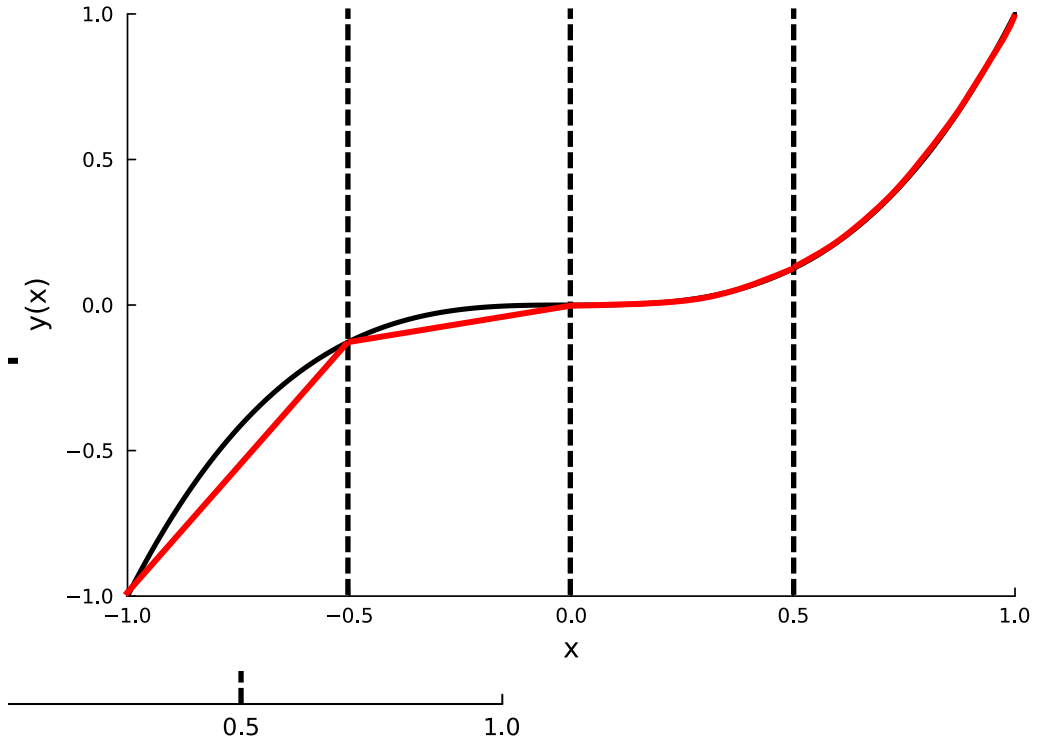
For an *n*-dimensional function, only 2*n*+1 pointwise evaluations (and one algebraic step) are required



7. Song, Y., et al. **Bounding convex relaxations of process models from below by tractable black-box sampling.** *Computers & Chemical Engineering* 153 (2021), 107413.

Parallel Evaluation

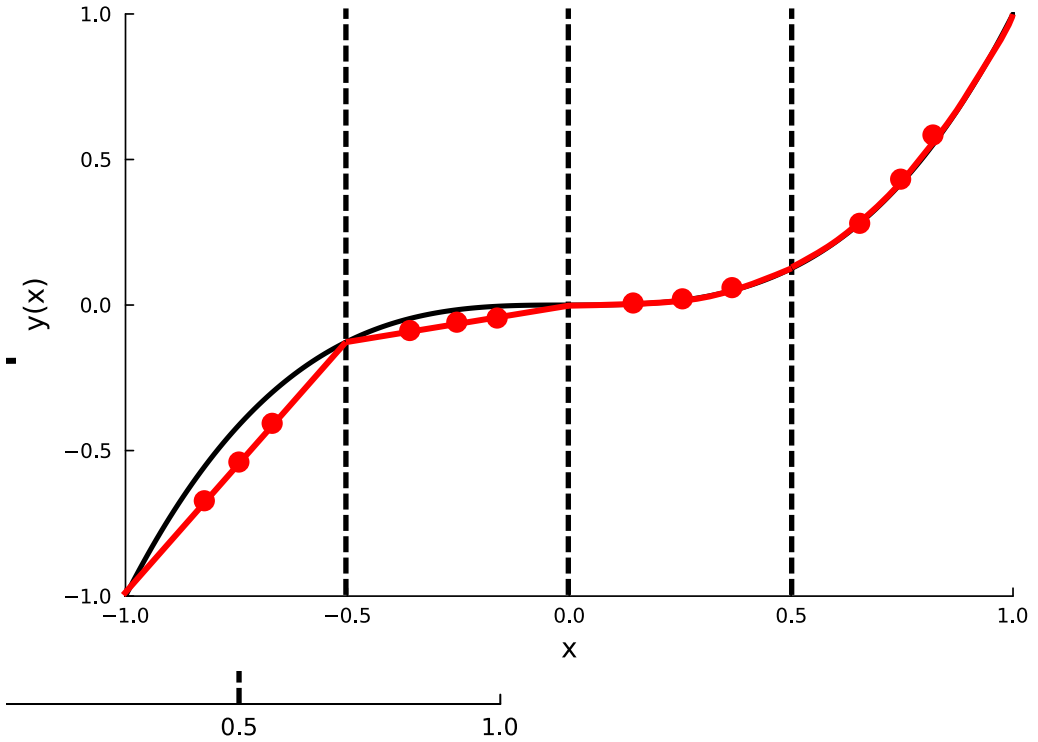
Step 1) Use SourceCodeMcCormick.jl to create a convex evaluator function of the objective



Parallel Evaluation

Step 1) Use SourceCodeMcCormick.jl to create a convex evaluator function of the objective

Step 2) For each node, identify $2n+1$ points to evaluate (based on variable bounds)

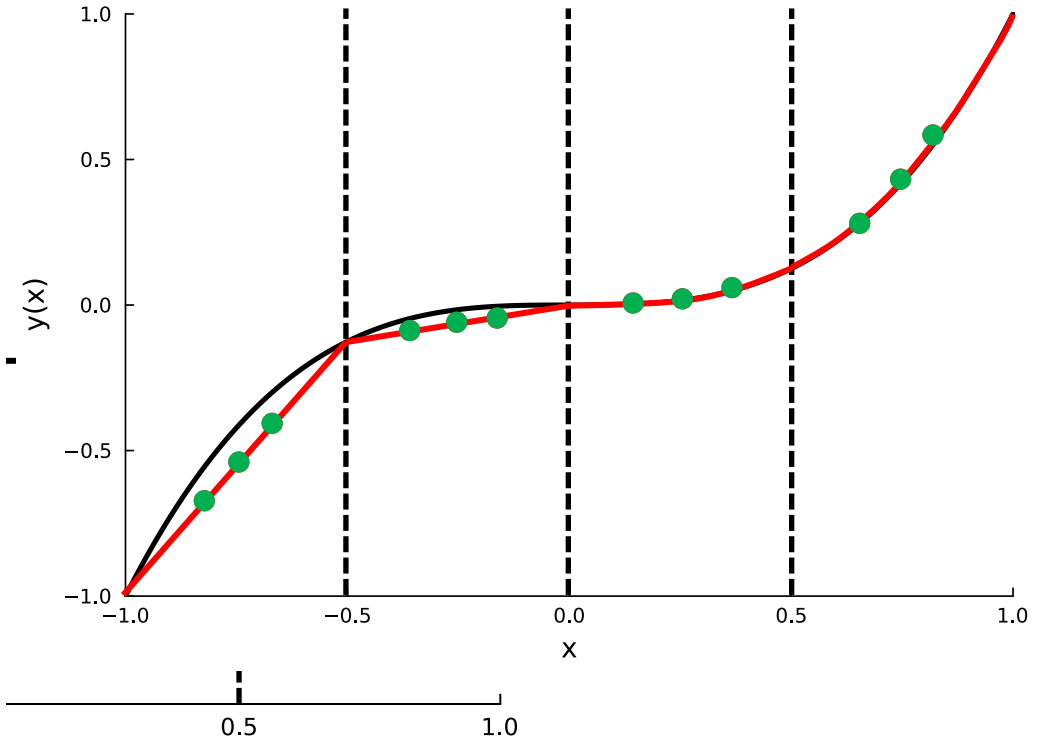


Parallel Evaluation

Step 1) Use SourceCodeMcCormick.jl to create a convex evaluator function of the objective

Step 2) For each node, identify $2n+1$ points to evaluate (based on variable bounds)

Step 3) Use the convex evaluator function to calculate the value of the convex relaxation at all $2n+1$ points for each node **simultaneously**



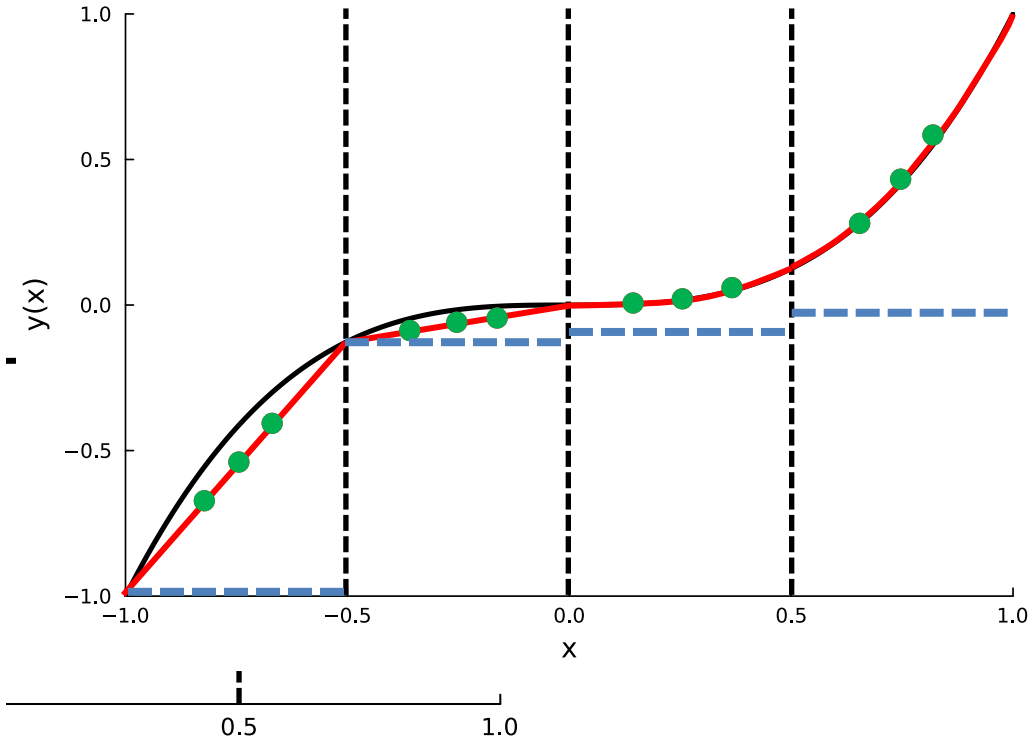
Parallel Evaluation

Step 1) Use SourceCodeMcCormick.jl to create a convex evaluator function of the objective

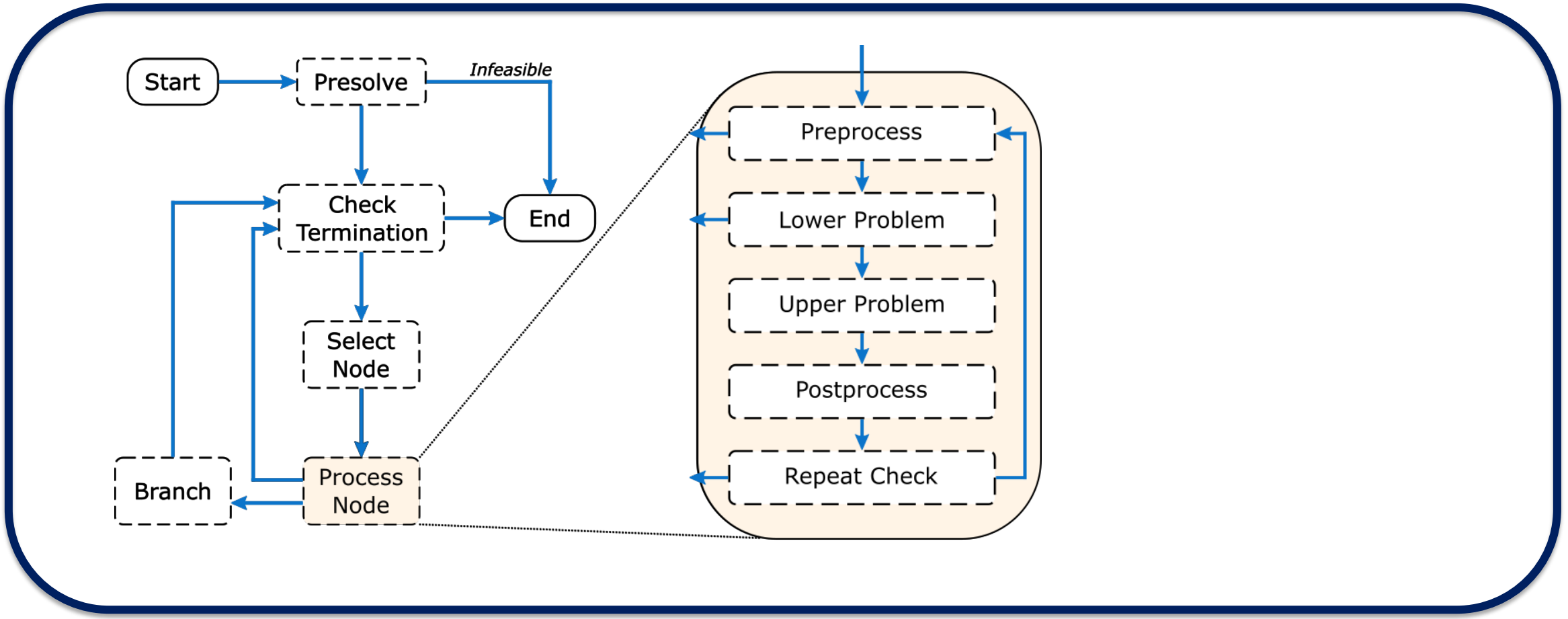
Step 2) For each node, identify $2n+1$ points to evaluate (based on variable bounds)

Step 3) Use the convex evaluator function to calculate the value of the convex relaxation at all $2n+1$ points for each node **simultaneously**

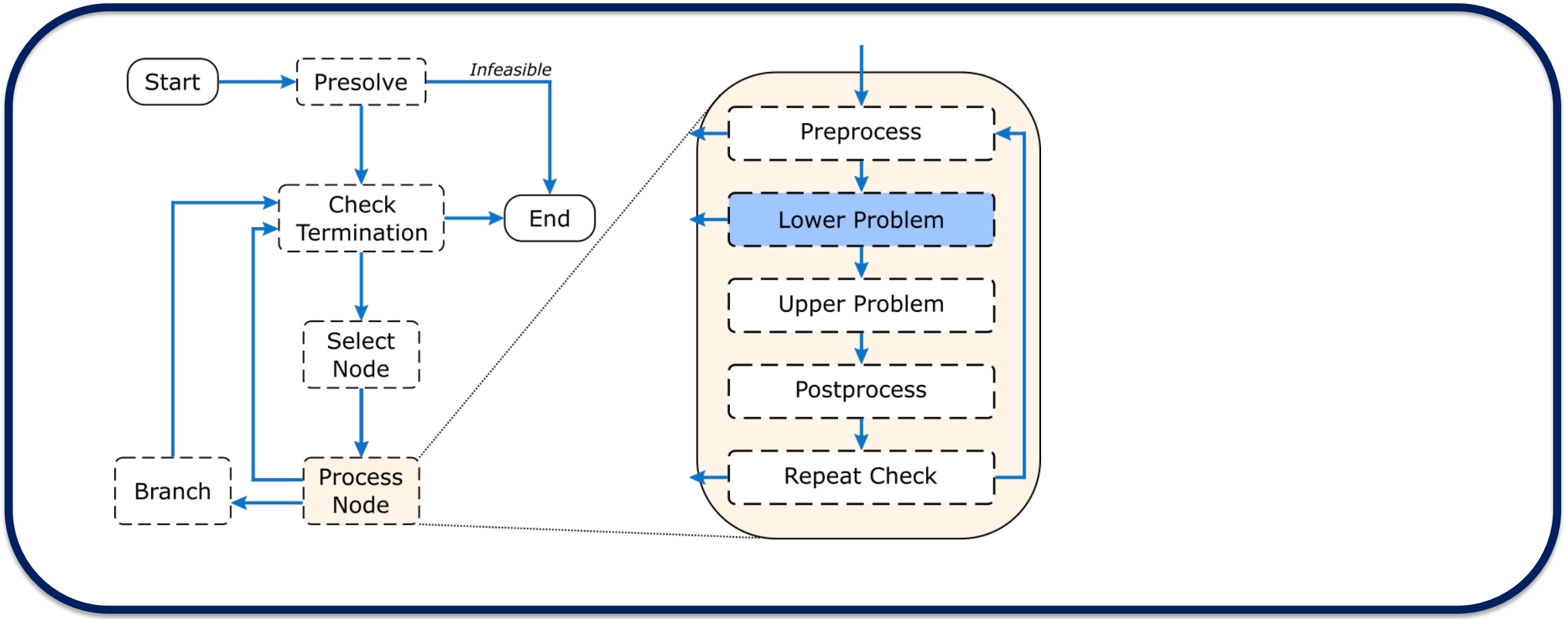
Step 4) Perform the algebraic step for each node to get guaranteed lower bounds



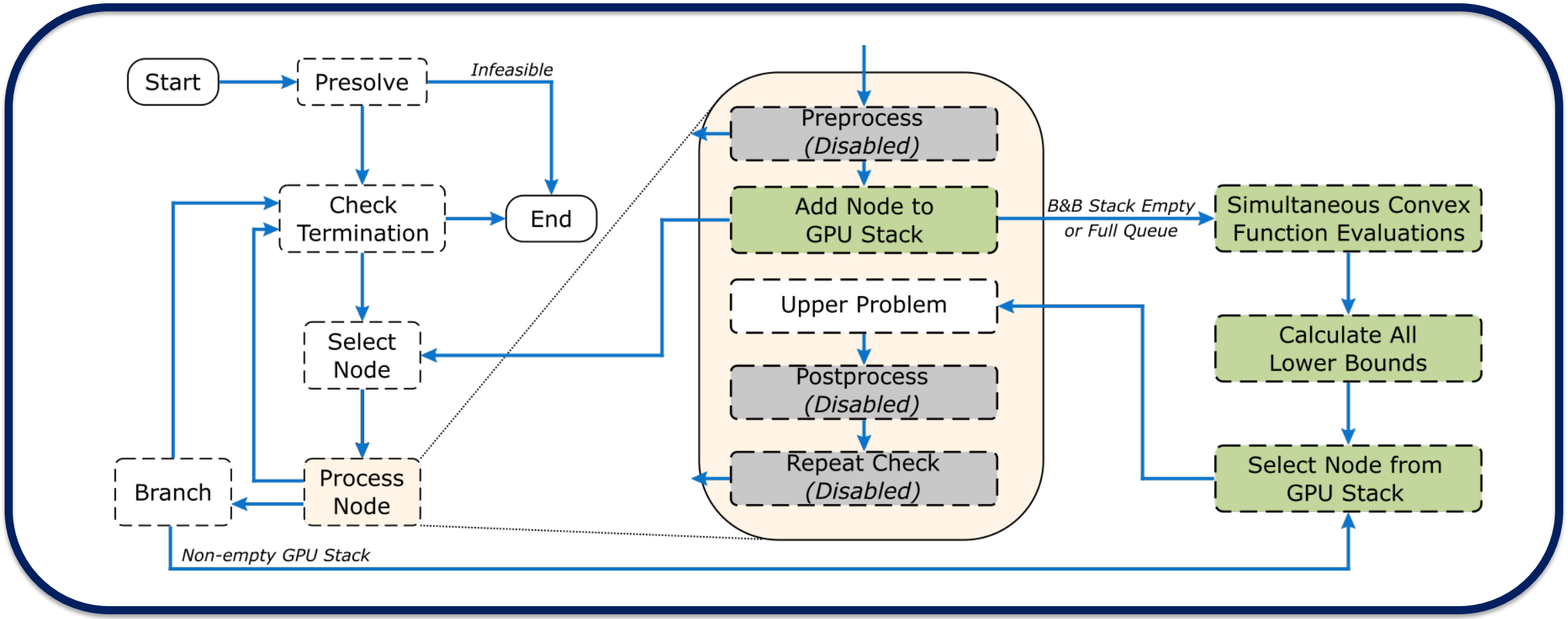
Changes to Branch-and-Bound



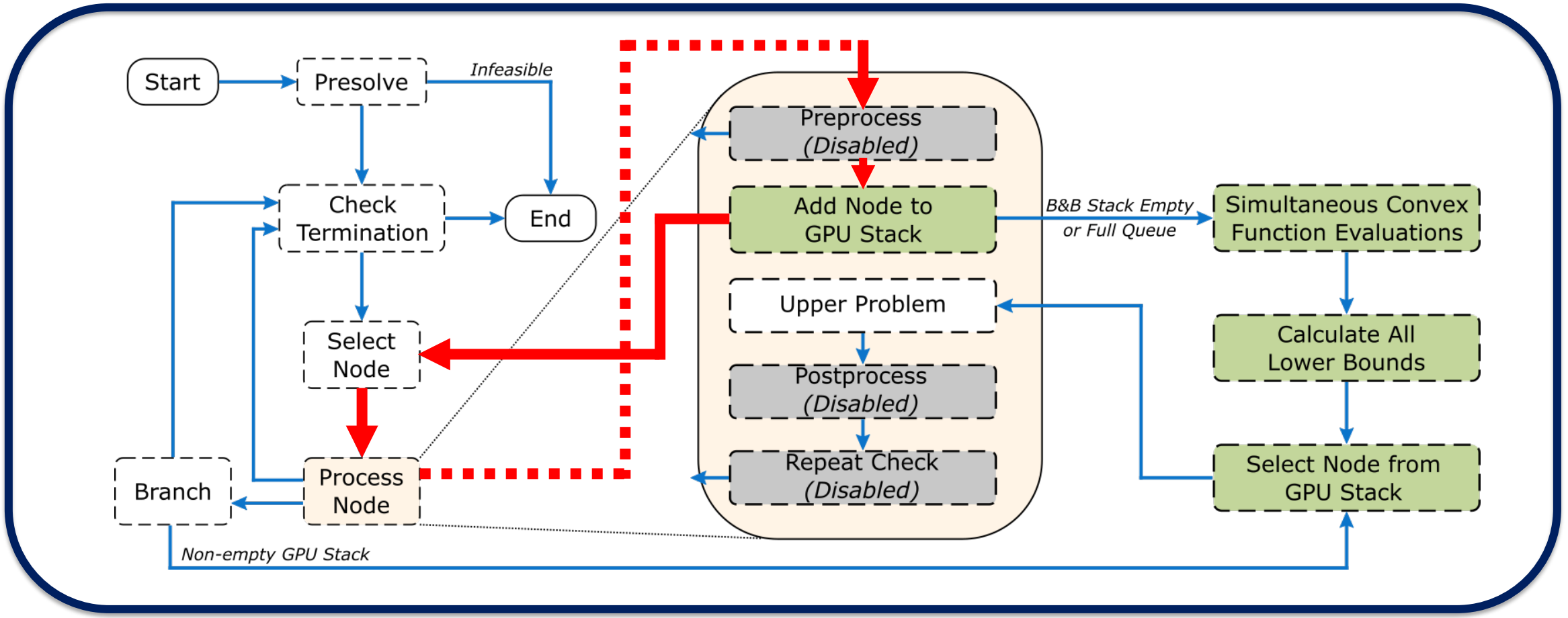
Changes to Branch-and-Bound



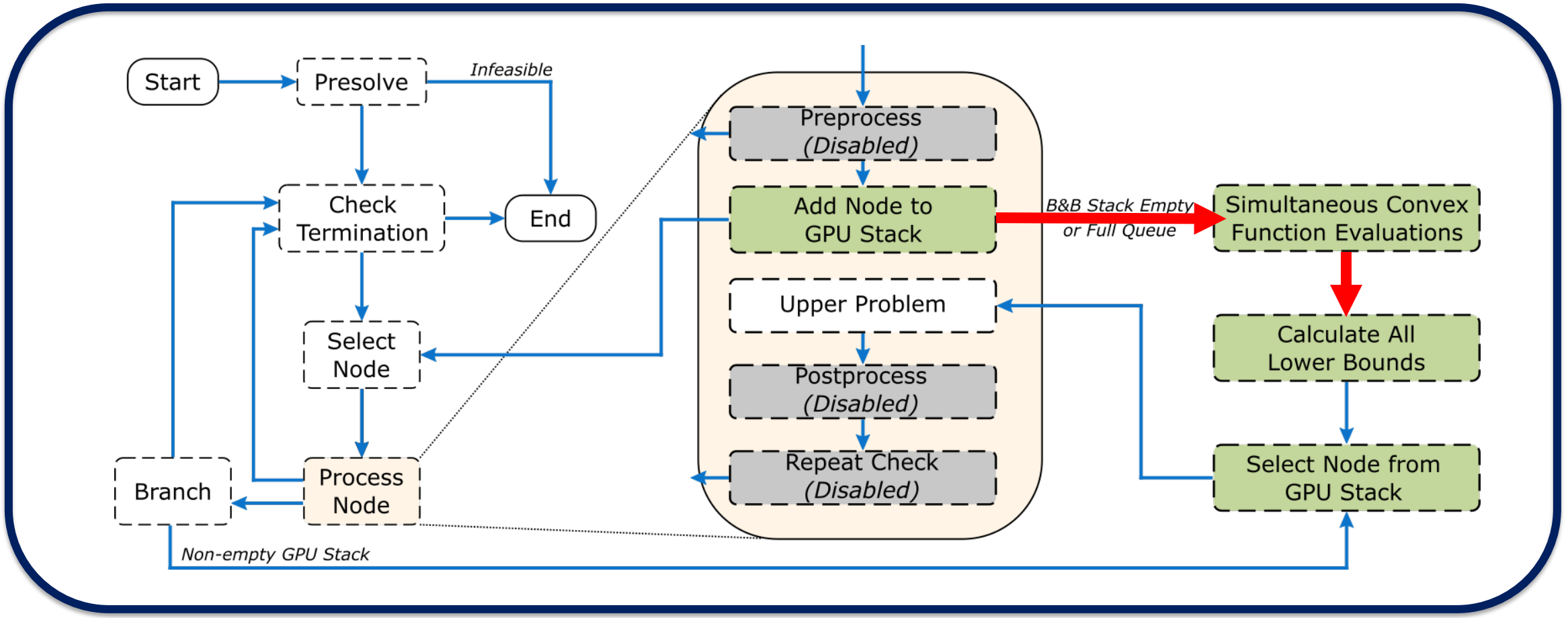
Changes to Branch-and-Bound



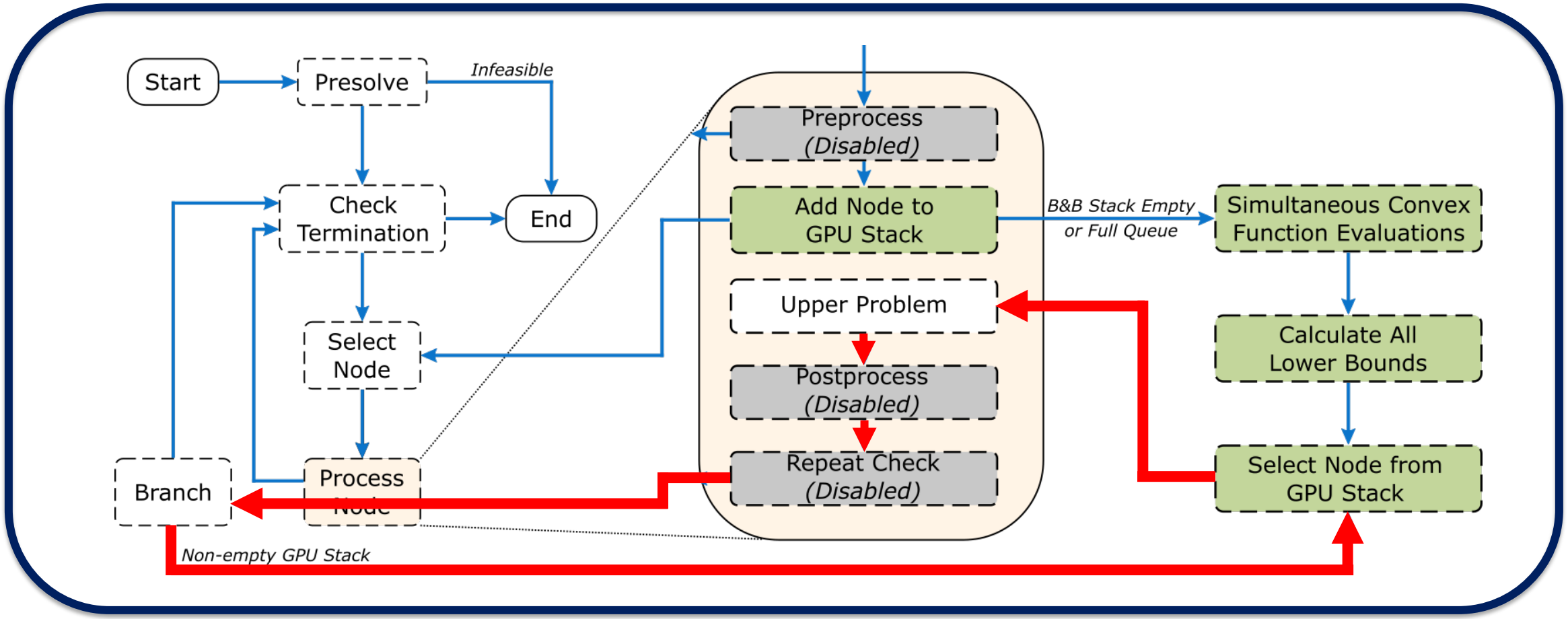
Changes to Branch-and-Bound



Changes to Branch-and-Bound



Changes to Branch-and-Bound



Kinetic Parameter Estimation

The concentrations of species $j \in \{A, B, D, Y, Z\}$ after an initial laser flash pyrolysis are modeled using the following system of ODEs:[8]

$$\frac{dx_A}{dt} = k_1 x_Z x_Y - c_{O_2}(k_{2f} + k_{3f})x_A + \frac{k_{2f}}{K_2} x_D + \frac{k_{3f}}{K_3} x_B - k_5 x_A^2,$$

$$\frac{dx_B}{dt} = c_{O_2} k_{3f} x_A - \left(\frac{k_{3f}}{K_3} + k_4\right) x_B, \qquad \frac{dx_Y}{dt} = -k_{1s} x_Z x_Y,$$

$$\frac{dx_D}{dt} = c_{O_2} k_{2f} x_A - \frac{k_{2f}}{K_2} x_D, \qquad \frac{dx_Z}{dt} = -k_1 x_z x_Y,$$

$$x_A(0) = x_B(0) = x_D(0) = 0, \quad x_Y(0) = 0.4, \quad x_Z(0) = 140.$$

8. Taylor, J. W. **Direct Measurement and Analysis of Cyclohexadienyl Oxidation.** Ph.D. thesis, Massachusetts Institute of Technology.

Kinetic Parameter Estimation

Objective: Adjust $\mathbf{p} = (k_{2\mathrm{f}}, k_{3\mathrm{f}}, k_4)$ to minimize the SSE between this model and experimental data

<u>Option 1)</u> SourceCodeMcCormick.jl + DiffEqGPU.jl[9,10]

$$\frac{dx_j}{dt} = \ldots$$

9. Rackauckas, C., et al. **Differentialequations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia.** *Journal of Open Research Software* 5:1 (2017).
10. Rackauckas, C., et al. **DiffEqGPU.jl**. GitHub repository, https://github.com/SciML/DiffEqGPU.jl (2019).
11. Wilhelm, M.E. et al. **Global optimization of stiff dynamical systems.** *AIChE Journal* **65**(12), e16836 (2019).

Kinetic Parameter Estimation

Objective: Adjust $\mathbf{p} = (k_{2f}, k_{3f}, k_4)$ to minimize the SSE between this model and experimental data

Option 1) SourceCodeMcCormick.jl + DiffEqGPU.jl[9,10]

SourceCodeMcCormick.jl

$$\frac{dx_j}{dt} = \ldots \longrightarrow \frac{dx_j^L}{dt} = \ldots, \quad \frac{dx_j^U}{dt} = \ldots, \quad \frac{dx_j^{cv}}{dt} = \ldots, \quad \frac{dx_j^{cc}}{dt} = \ldots,$$

9. Rackauckas, C., et al. **Differentialequations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia.** *Journal of Open Research Software* 5:1 (2017).
10. Rackauckas, C., et al. **DiffEqGPU.jl**. GitHub repository, https://github.com/SciML/DiffEqGPU.jl (2019).
11. Wilhelm, M.E. et al. **Global optimization of stiff dynamical systems.** *AIChE Journal* **65**(12), e16836 (2019).

Kinetic Parameter Estimation

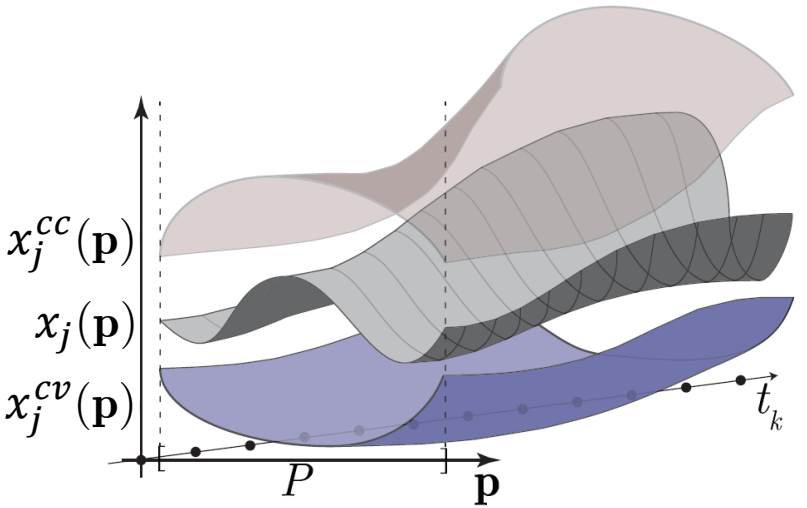
Objective: Adjust $\mathbf{p} = (k_{2f}, k_{3f}, k_4)$ to minimize the SSE between this model and experimental data

<u>Option 1)</u> SourceCodeMcCormick.jl + DiffEqGPU.jl[9,10]

SourceCodeMcCormick.jl

$$\frac{dx_j}{dt} = \ldots \longrightarrow \frac{dx_j^L}{dt} = \ldots, \quad \frac{dx_j^U}{dt} = \ldots, \quad \frac{dx_j^{cv}}{dt} = \ldots, \quad \frac{dx_j^{cc}}{dt} = \ldots,$$

DiffEqGPU.jl



9. Rackauckas, C., et al. **Differentialequations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia.** *Journal of Open Research Software* 5:1 (2017).
10. Rackauckas, C., et al. **DiffEqGPU.jl**. GitHub repository, https://github.com/SciML/DiffEqGPU.jl (2019).
11. Wilhelm, M.E. et al. **Global optimization of stiff dynamical systems.** *AIChE Journal* **65**(12), e16836 (2019).

Kinetic Parameter Estimation

Objective: Adjust $\mathbf{p} = (k_{2\mathrm{f}}, k_{3\mathrm{f}}, k_4)$ to minimize the SSE between this model and experimental data

<u>Option 2)</u> SourceCodeMcCormick.jl + Explicit Euler integration

$$\frac{dx_j}{dt} = \ldots$$

11. Wilhelm, M.E. et al. **Global optimization of stiff dynamical systems.** *AIChE Journal* **65**(12), e16836 (2019).

Kinetic Parameter Estimation

Objective: Adjust $\mathbf{p} = (k_{2\mathrm{f}}, k_{3\mathrm{f}}, k_4)$ to minimize the SSE between this model and experimental data

Option 2) SourceCodeMcCormick.jl + Explicit Euler integration

Explicit Euler

$$\frac{dx_j}{dt} = \ldots \longrightarrow x_{j,t+1} = x_{j,t} + hf_j(\mathbf{x}_t)$$

11. Wilhelm, M.E. et al. **Global optimization of stiff dynamical systems.** *AIChE Journal* **65**(12), e16836 (2019).

Kinetic Parameter Estimation

Objective: Adjust $\mathbf{p} = (k_{2f}, k_{3f}, k_4)$ to minimize the SSE between this model and experimental data

<u>Option 2)</u> SourceCodeMcCormick.jl + Explicit Euler integration

Explicit Euler

SourceCodeMcCormick.jl

$$\frac{dx_j}{dt} = \ldots \longrightarrow x_{j,t+1} = x_{j,t} + hf_j(\mathbf{x}_t) \longrightarrow x^L_{j,t+1},\ x^U_{j,t+1},\ x^{cv}_{j,t+1},\ x^{cc}_{j,t+1}$$

11. Wilhelm, M.E. et al. **Global optimization of stiff dynamical systems.** *AIChE Journal* **65**(12), e16836 (2019).

Kinetic Parameter Estimation

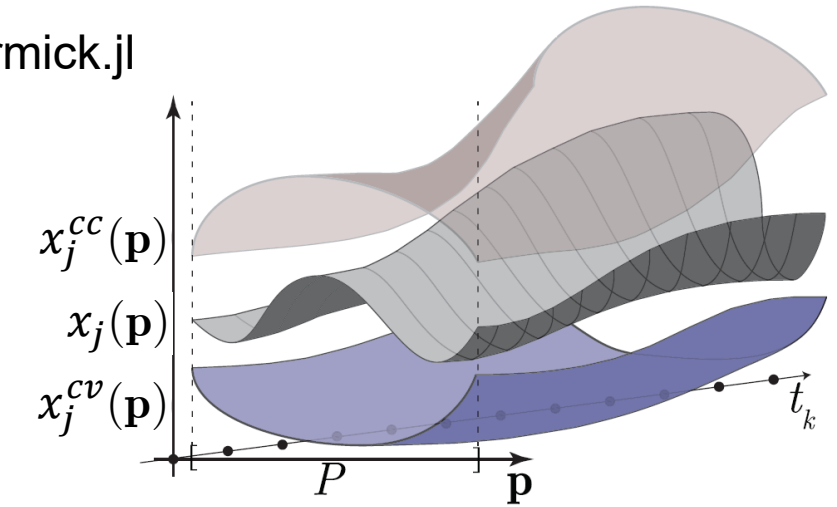
Objective: Adjust $\mathbf{p} = (k_{2f}, k_{3f}, k_4)$ to minimize the SSE between this model and experimental data

<u>Option 2)</u> SourceCodeMcCormick.jl + Explicit Euler integration

Explicit Euler

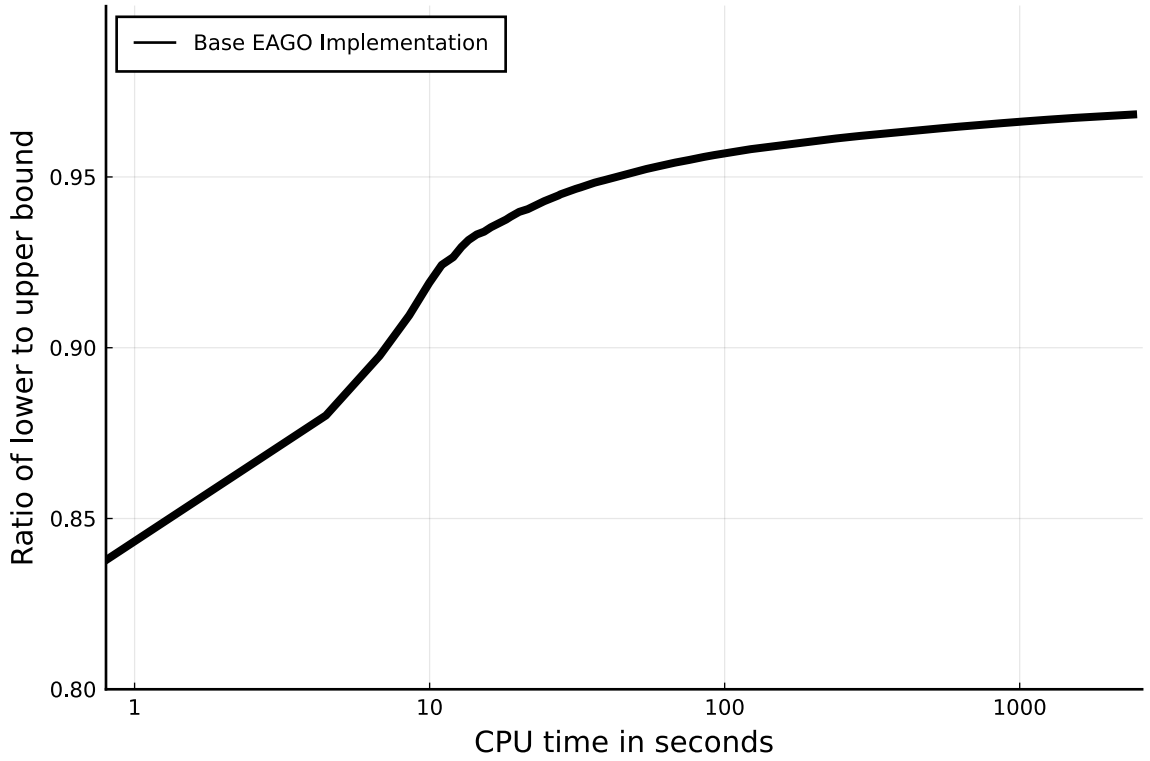
$$\frac{dx_j}{dt} = \ldots \longrightarrow x_{j,t+1} = x_{j,t} + hf_j(\mathbf{x}_t)$$

SourceCodeMcCormick.jl



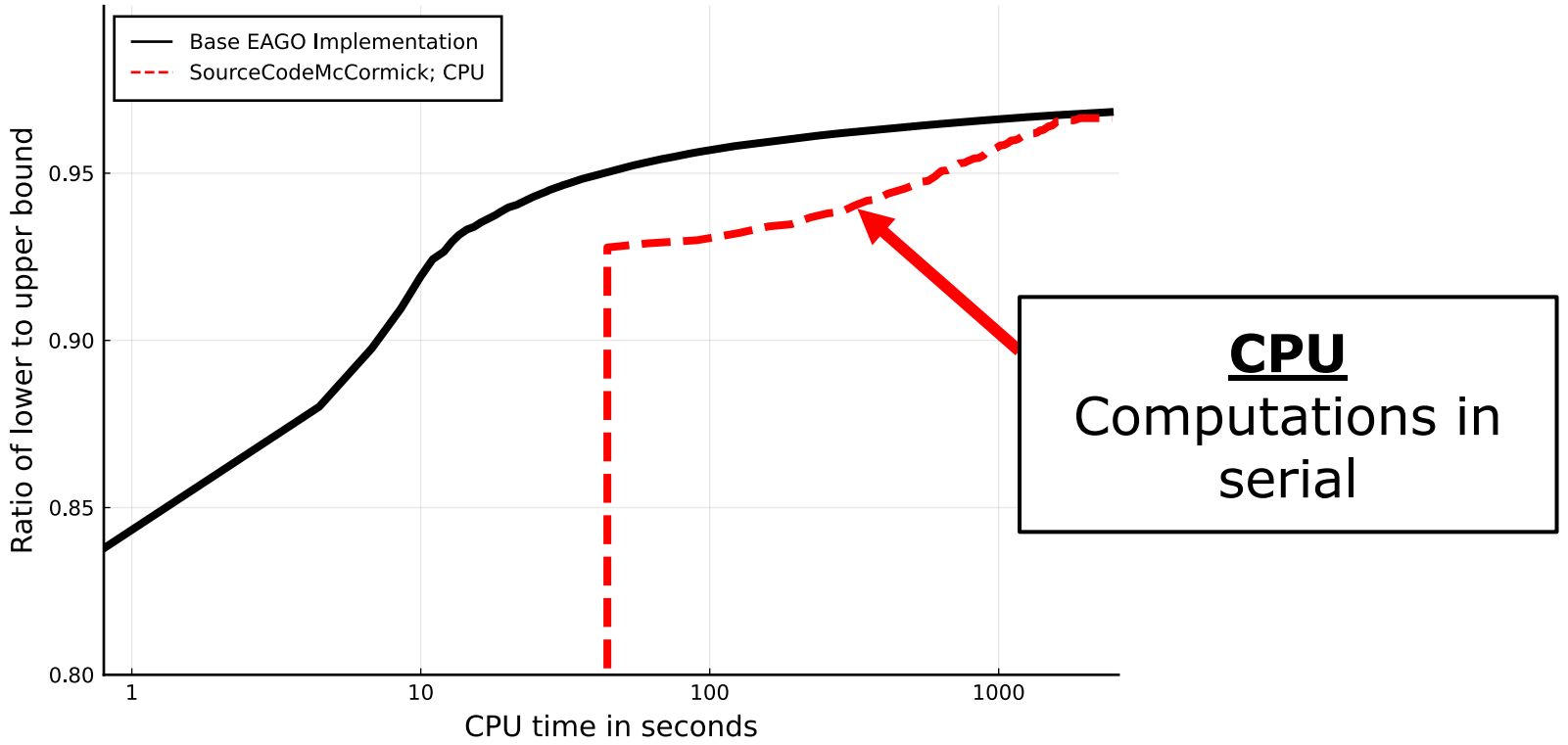
11. Wilhelm, M.E. et al. **Global optimization of stiff dynamical systems.** *AIChE Journal* **65**(12), e16836 (2019).

Results



CPU: Intel W-2195
GPU: NVIDIA Quadro GV100

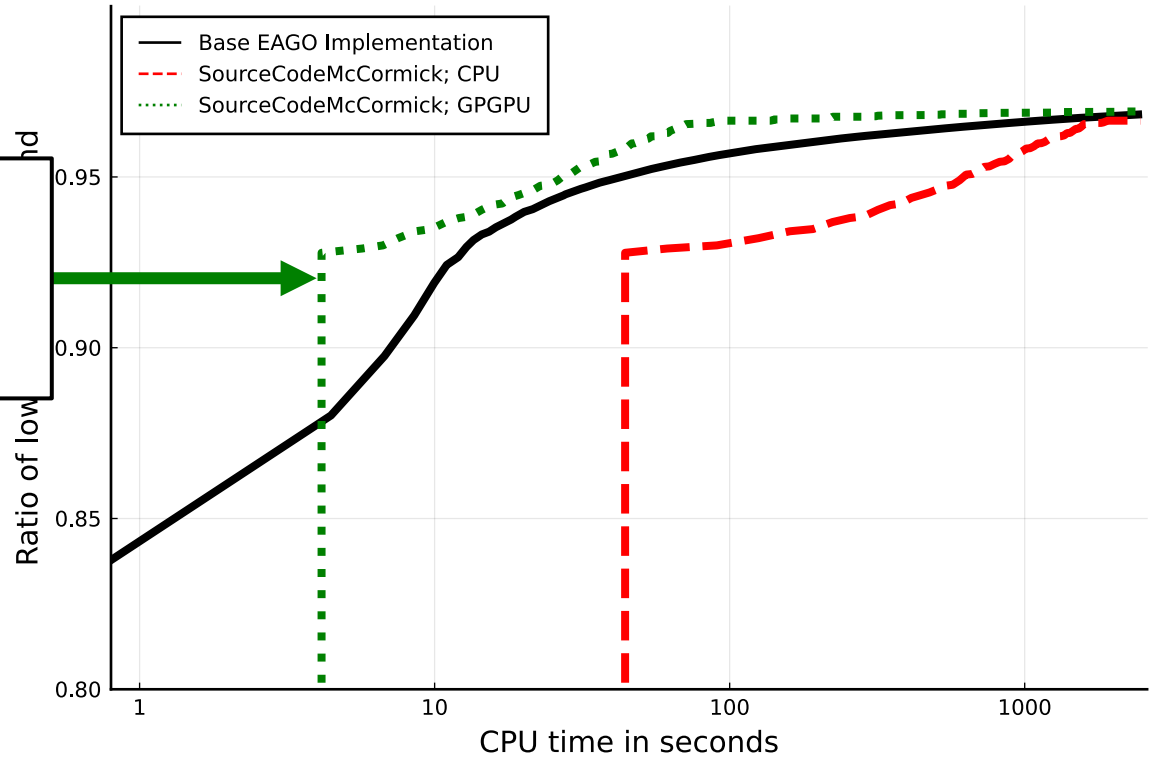
Results



CPU: Intel W-2195
GPU: NVIDIA Quadro GV100

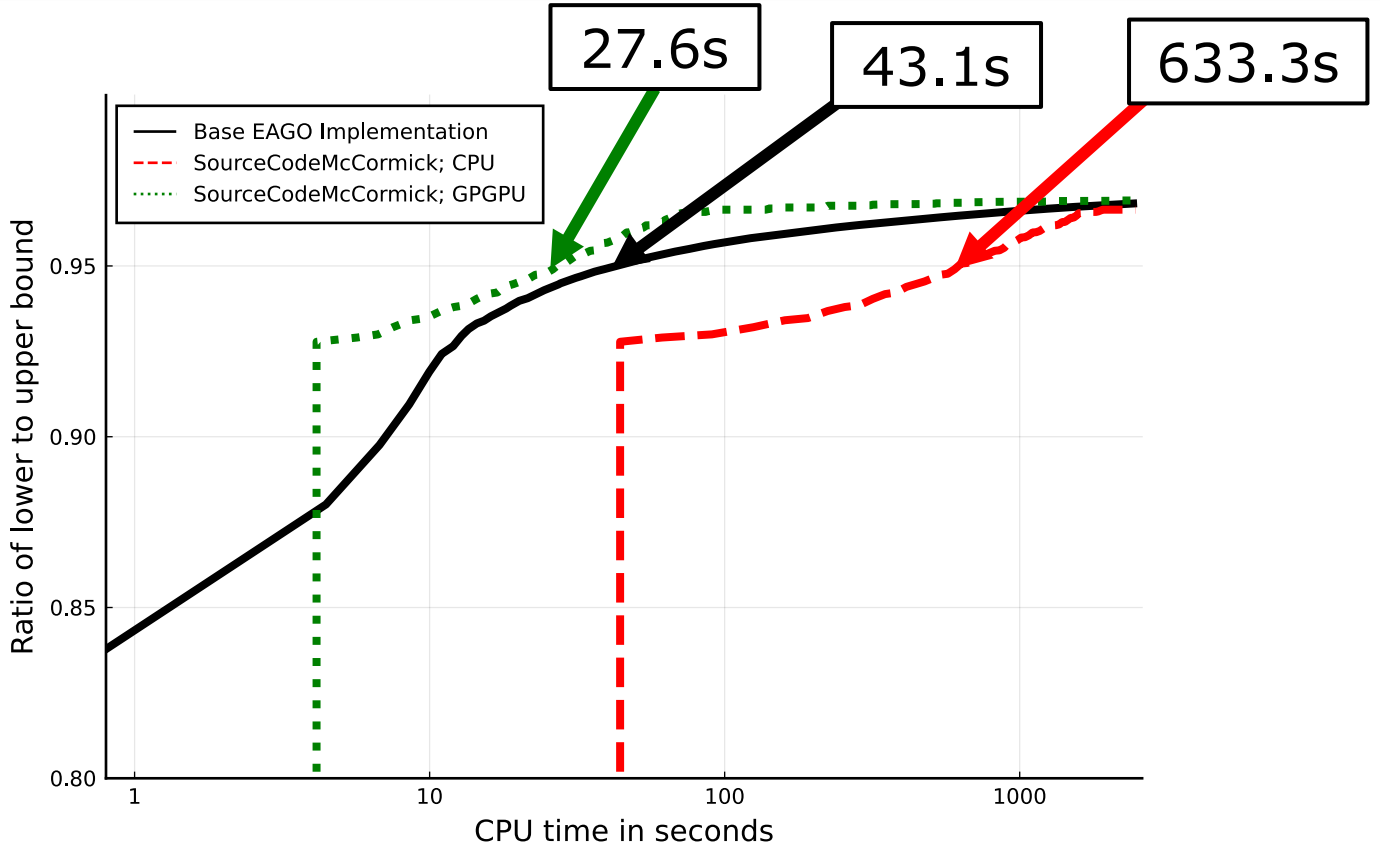
Results

GPU
50,000 nodes in parallel



CPU: Intel W-2195
GPU: NVIDIA Quadro GV100

Results



CPU: Intel W-2195
GPU: NVIDIA Quadro GV100

Conclusions

- First implementation of parallelized B&B on a GPU
- Weaker lower bounding method than base EAGO
  - Comparable performance to state-of-the-art serial CPU branch-and-bound

Future Efforts:

- Other B&B steps can benefit from parallelization
- Parallelizing alternative lower bounding methods can boost performance further

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