UCONN
UNIVERSITY OF CONNECTICUT

GPU-Accelerated Deterministic Global Optimization

Robert Gottlieb, PhD Student
Matthew Stuber, P&W Associate Professor in Advanced Systems Engineering

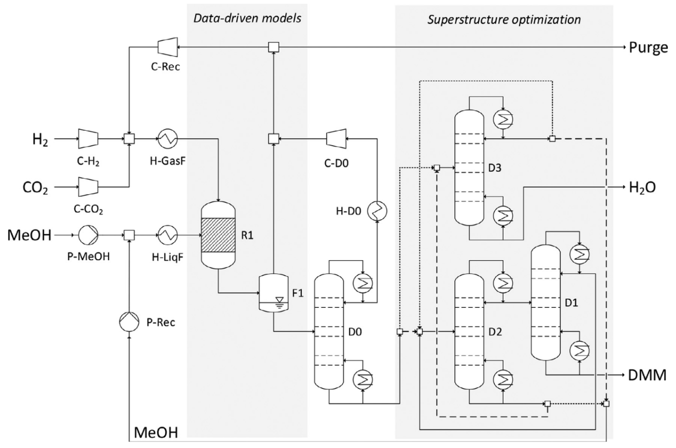
July 23, 2024

{ ISMP 2024 }

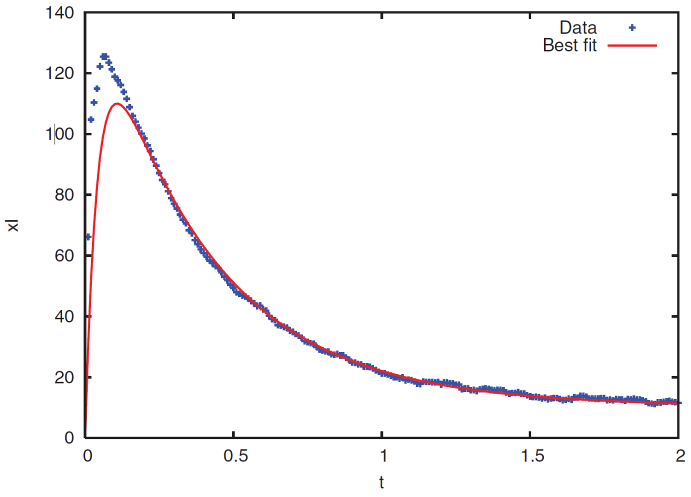
Process Systems and Operations Research Laboratory

Deterministic Global Optimization

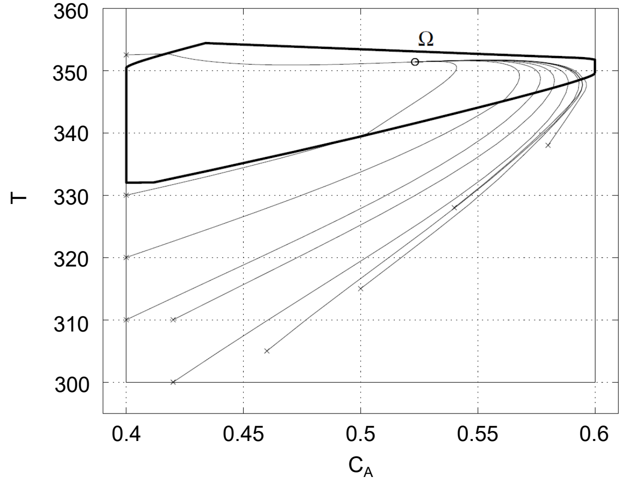
Design Improvements[1]



Parameter Estimation and Model Validation[2]



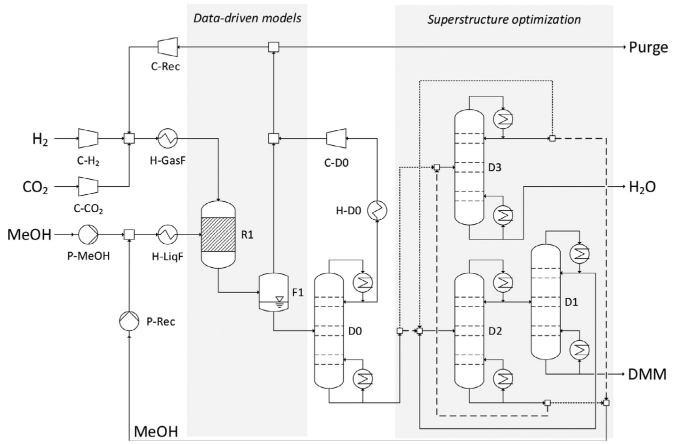
Safety and Robust Control[3]



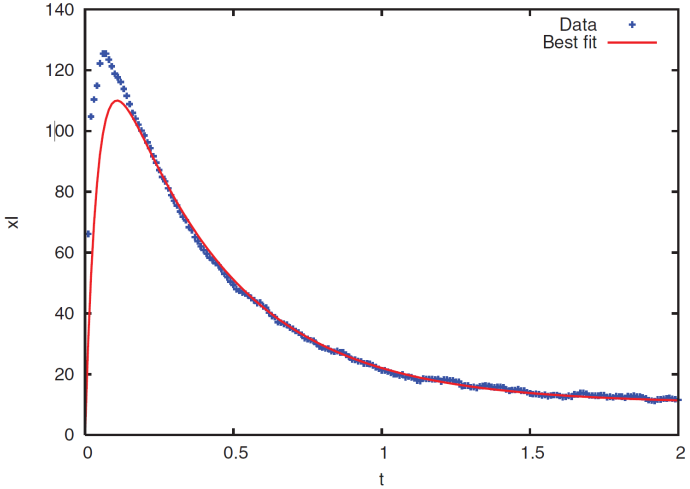
1. Burre, J., *et al*. **Global flowsheet optimization for reductive dimethoxymethane production using data-driven thermodynamic models.** *Computers & Chemical Engineering*, (2022): 107806.
2. Mitsos, A., et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.
3. 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

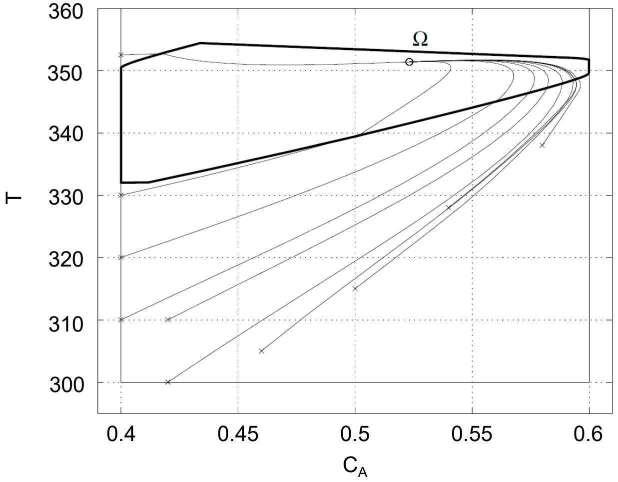
Design Improvements[1]



Parameter Estimation and Model Validation[2]



Safety and Robust Control[3]



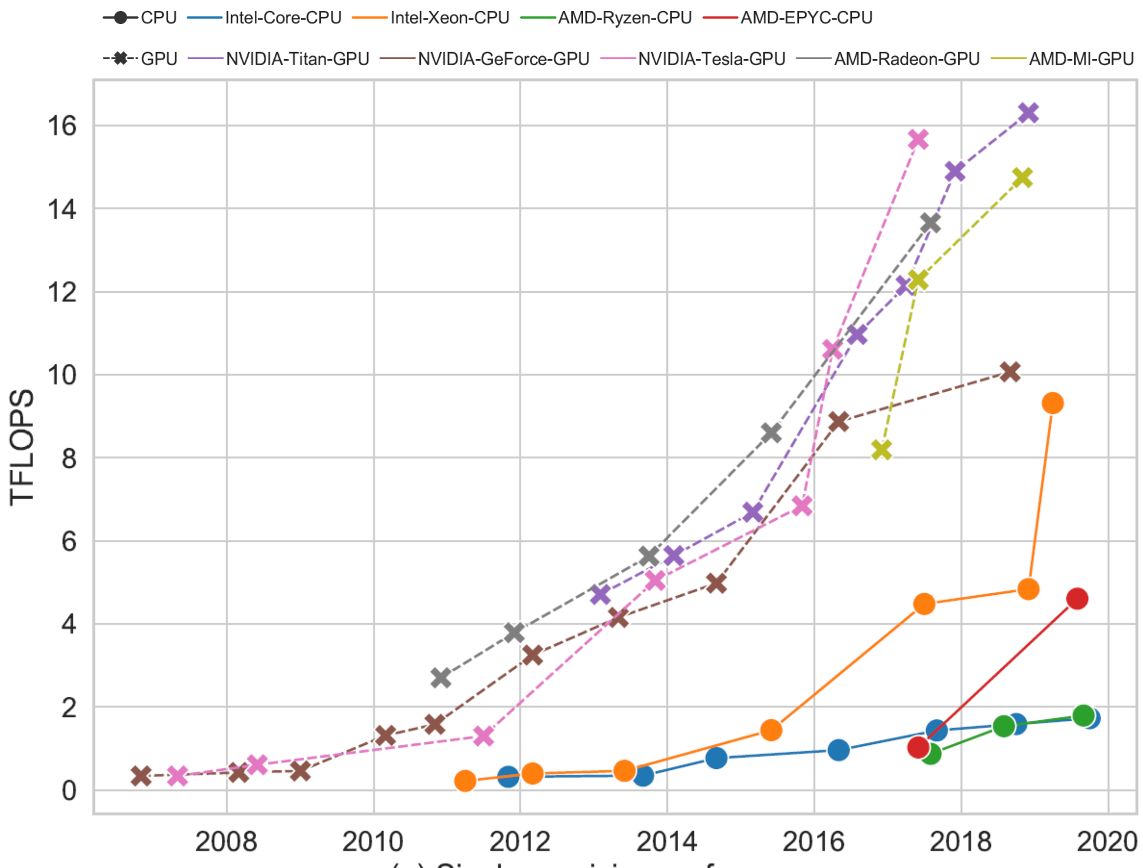
Problems are NP-hard: worst-case exponential runtime

1. Burre, J., *et al*. **Global flowsheet optimization for reductive dimethoxymethane production using data-driven thermodynamic models.** *Computers & Chemical Engineering*, (2022): 107806.
2. Mitsos, A., et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.
3. 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).

High-Performance Computing

Graphics Processing Units (GPUs)

- Graphics rendering
- Machine learning model training
  - Generative AI
- Data analysis
- Large-scale simulations
  - Molecular dynamics
  - CFD modeling
- Supercomputing
- [...]

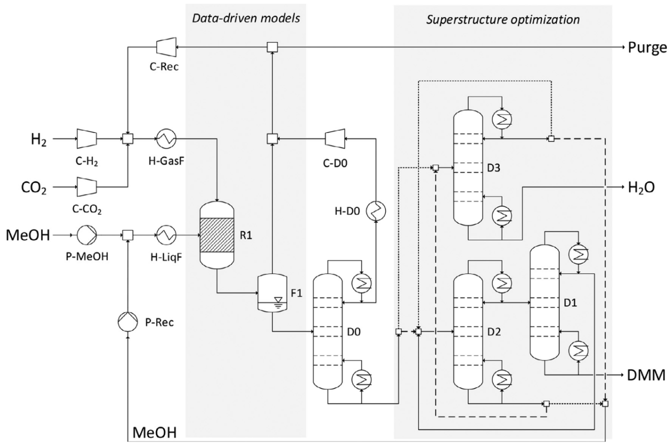


(a) Single-precision performance.

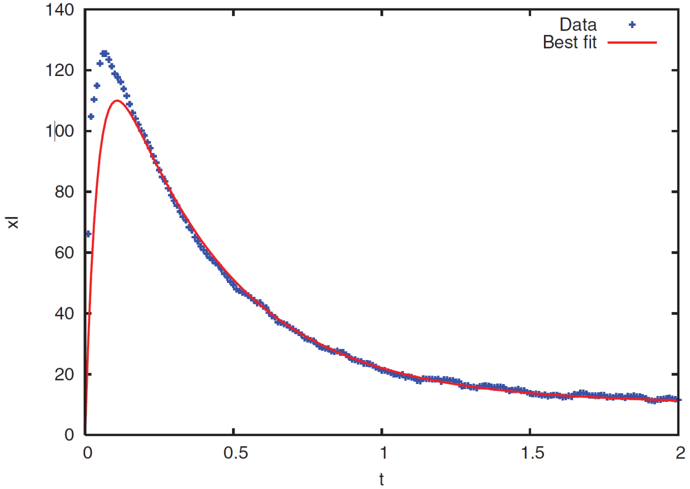
4. Sun, Y., et al. **Summarizing CPU and GPU design trends with product data.** *arXiv*, nov 2019. arXiv: 1911.11313

GPUs for Global Optimization?

1



2



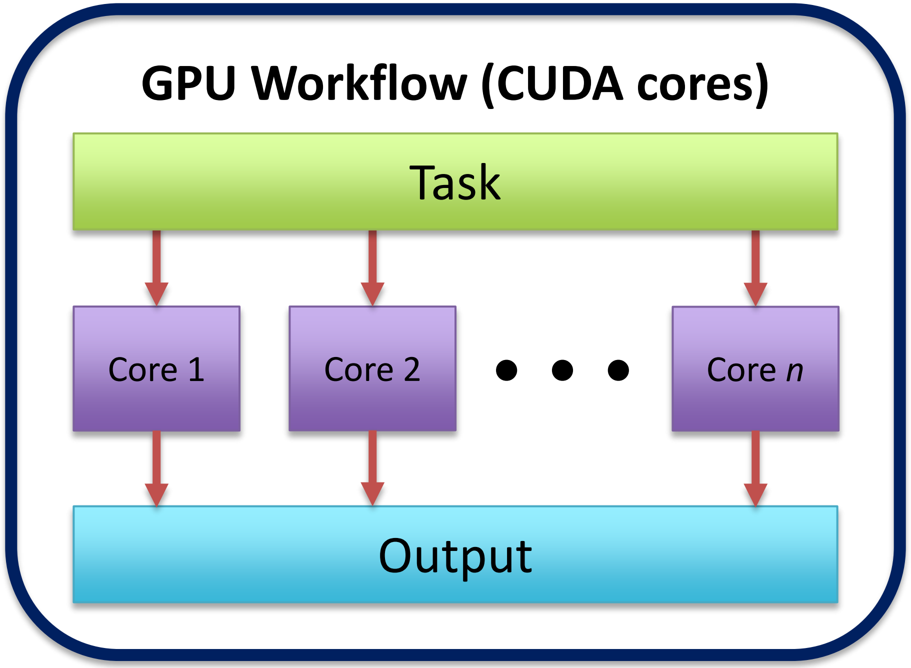
Accelerate using **GPUs?**

Energy.gov

1. Burre, J., *et al*. **Global flowsheet optimization for reductive dimethoxymethane production using data-driven thermodynamic models.** *Computers & Chemical Engineering*, (2022): 107806.
2. Mitsos, A., et al. **McCormick-based relaxations of algorithms.** *SIAM Journal on Optimization*, SIAM (2009) 20, 73-601.

GPU Concepts

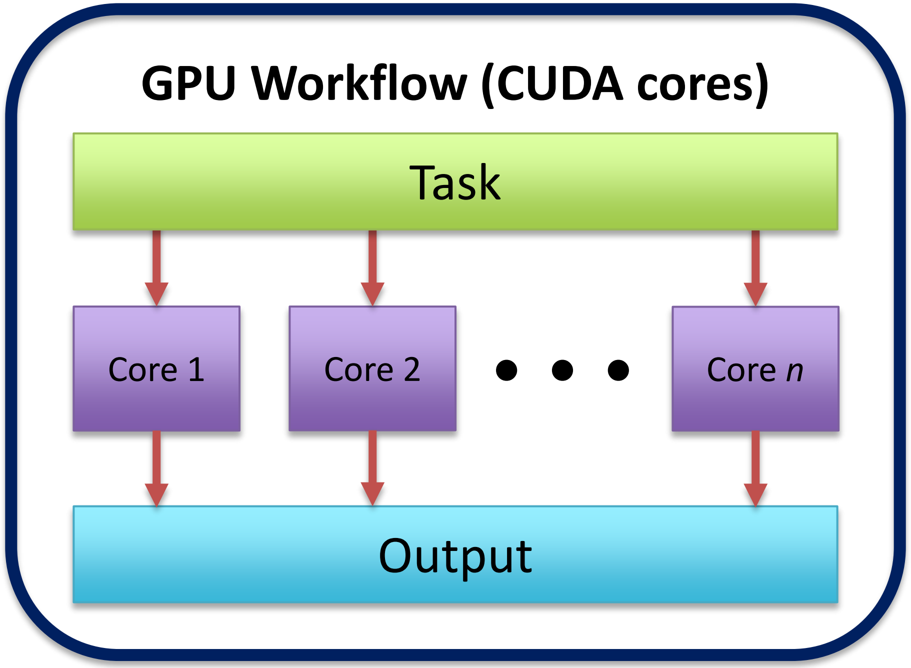
- GPUs are built for **data parallelism (SIMD)**
  - Thousands of GPU cores (threads) execute tasks simultaneously
  - Large chunks of **identical** data processing (I.e., the **inputs** change, not the math)



GPU Concepts

- GPUs are built for **data parallelism (SIMD)**
  - Thousands of GPU cores (threads) execute tasks simultaneously
  - Large chunks of **identical** data processing (I.e., the **inputs** change, not the math)

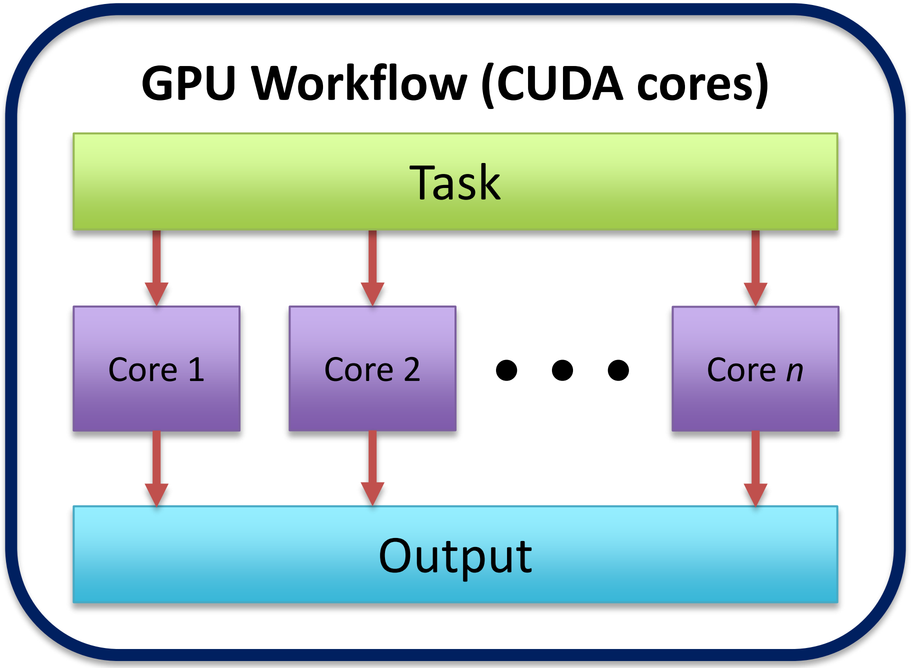
<u>Branch-and-bound (B&B) nodes:</u>
- **Same** optimization problem
- **Same** processing technique
- Different **domains**

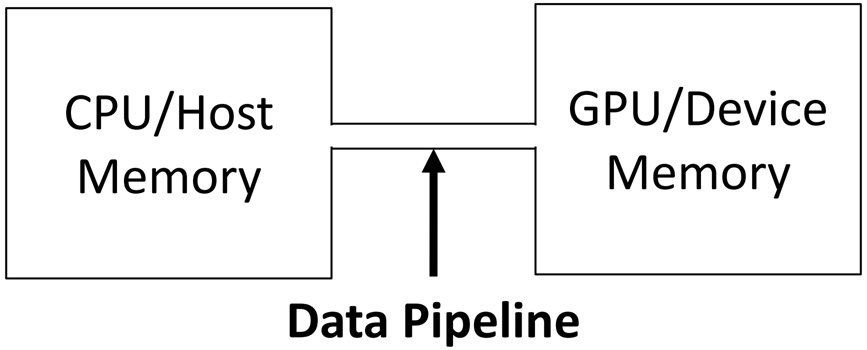


GPU Concepts

- GPUs are built for **data parallelism (SIMD)**
  - Thousands of GPU cores (threads) execute tasks simultaneously
  - Large chunks of **identical** data processing (I.e., the **inputs** change, not the math)

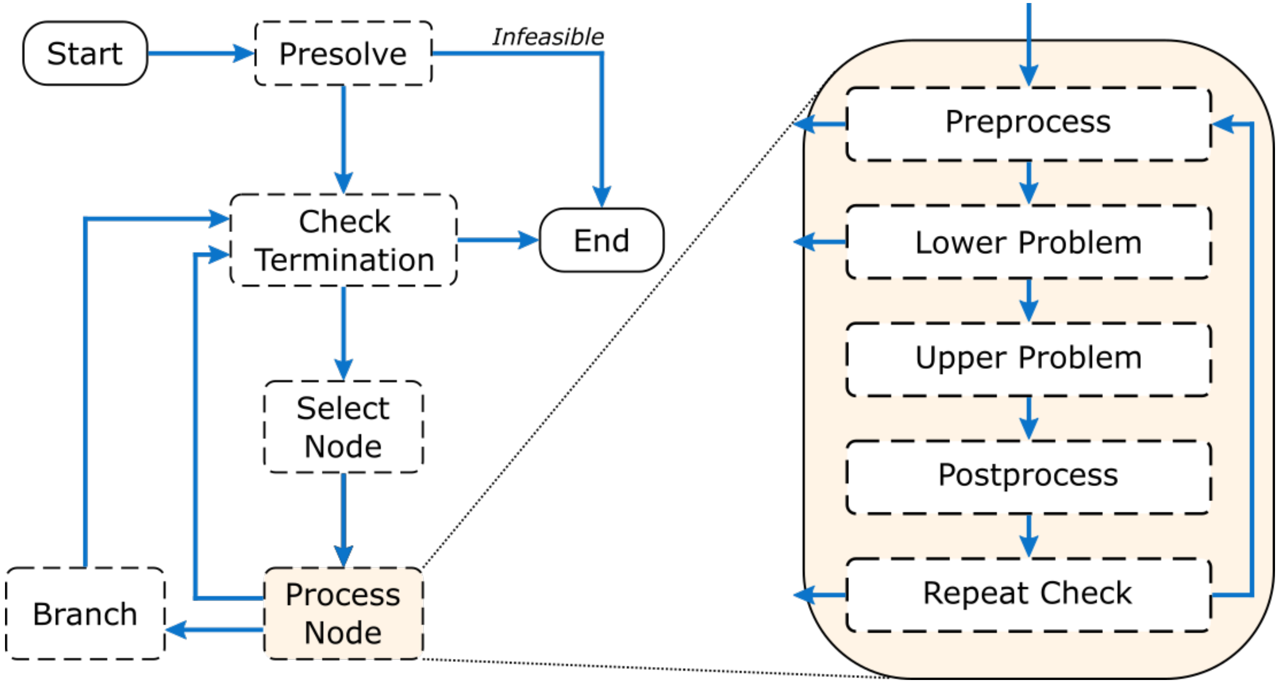
- **Major GPU bottlenecks:**
  - Code branches bad for performance
  - High data transfer overhead cost





Adapting Branch-and-Bound (B&B)

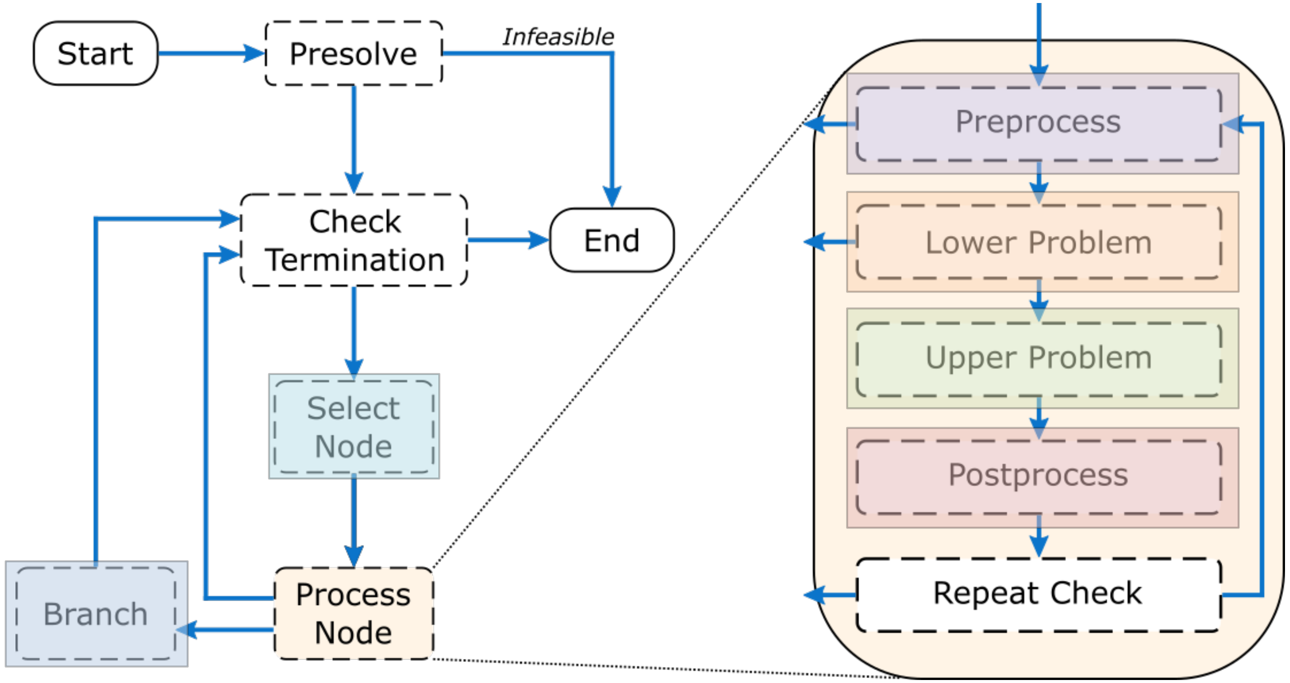
What aspects of B&B should be parallelized?



5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

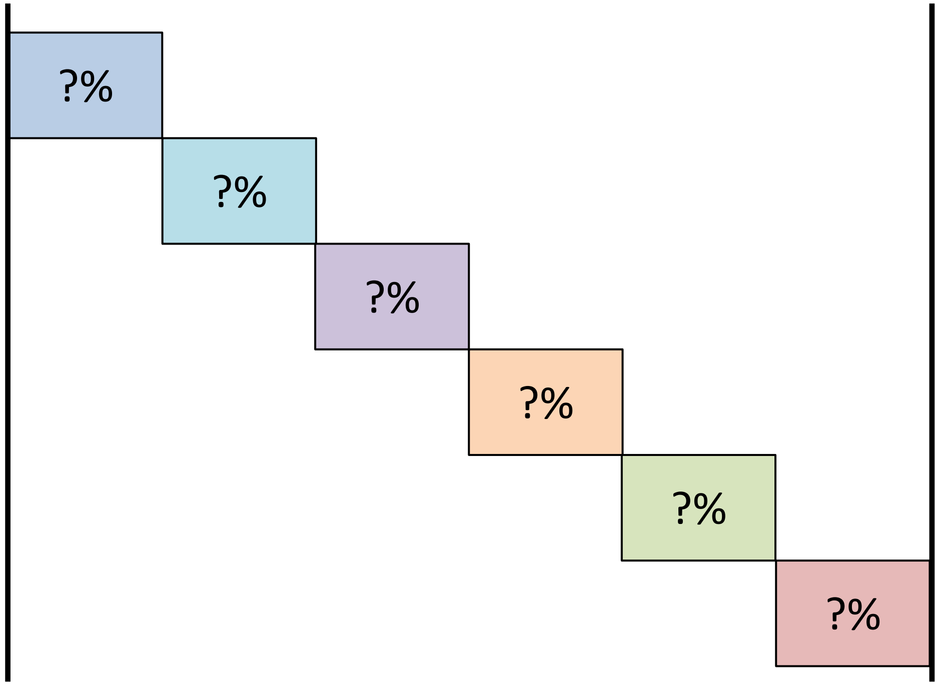
Adapting Branch-and-Bound (B&B)

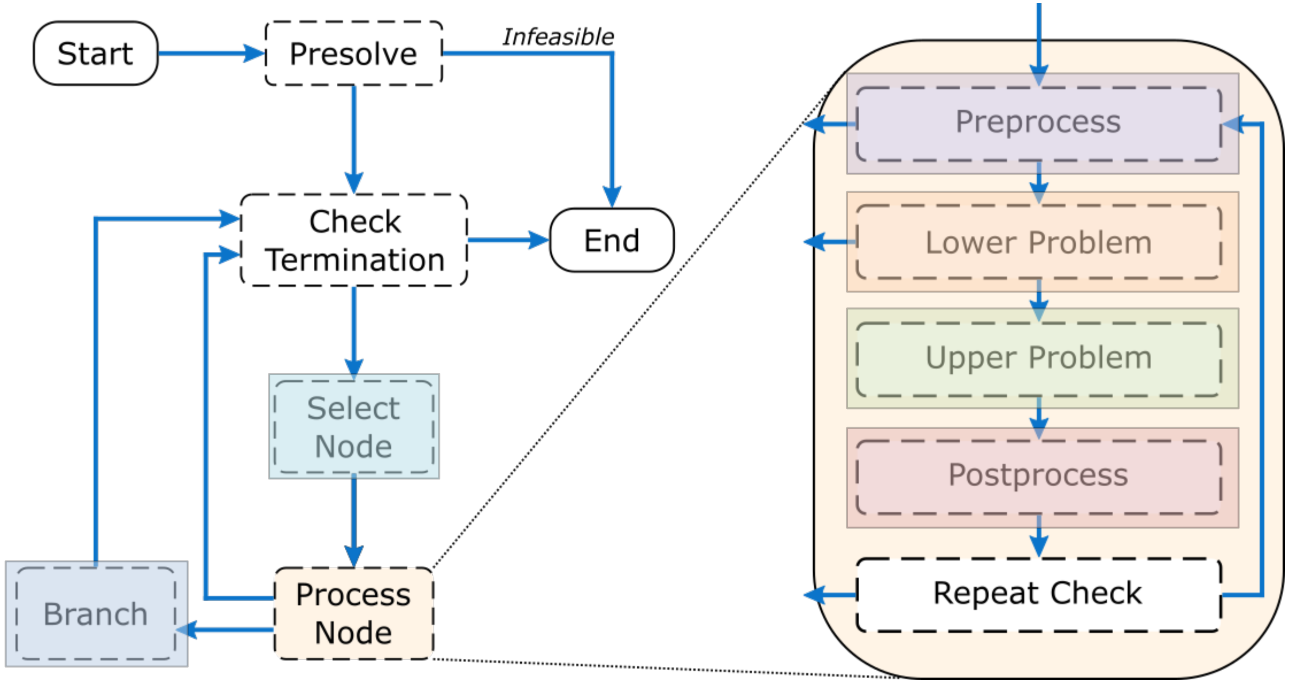
What aspects of B&B should be parallelized?



5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

B&B Step Timing



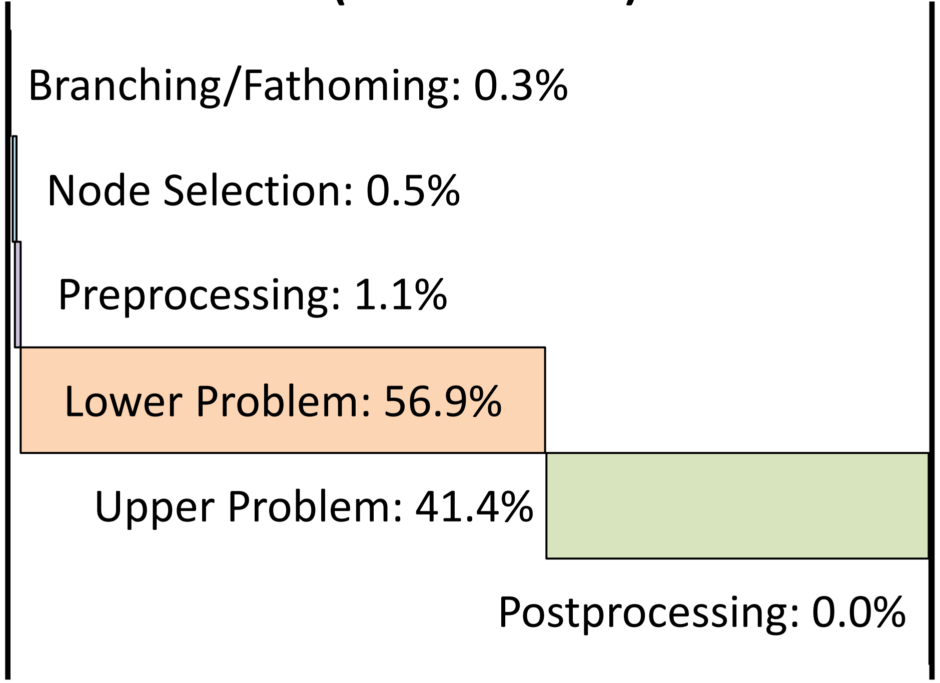


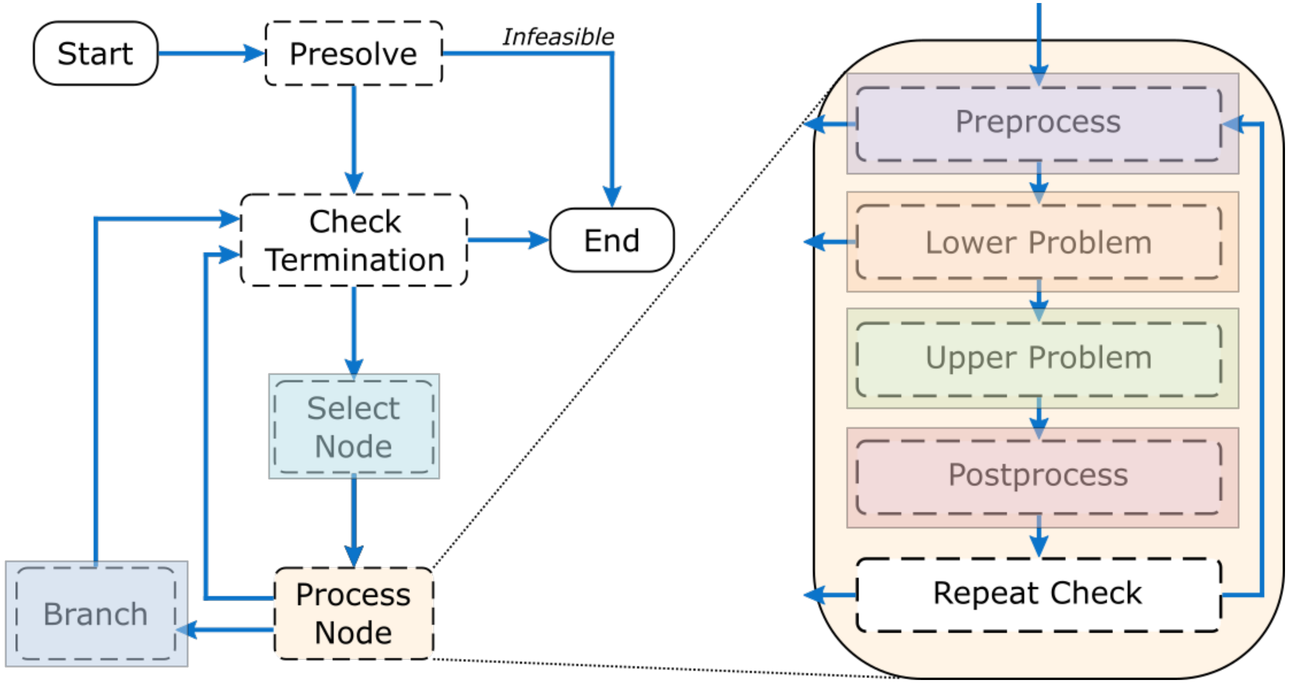
5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

B&B Step Timing

Example 1

Typical "Short" Problem (<60s runtime)





5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

B&B Step Timing

Example 2

Typical "Long" Problem (>2h runtime)

Branching/Fathoming: 0.2%

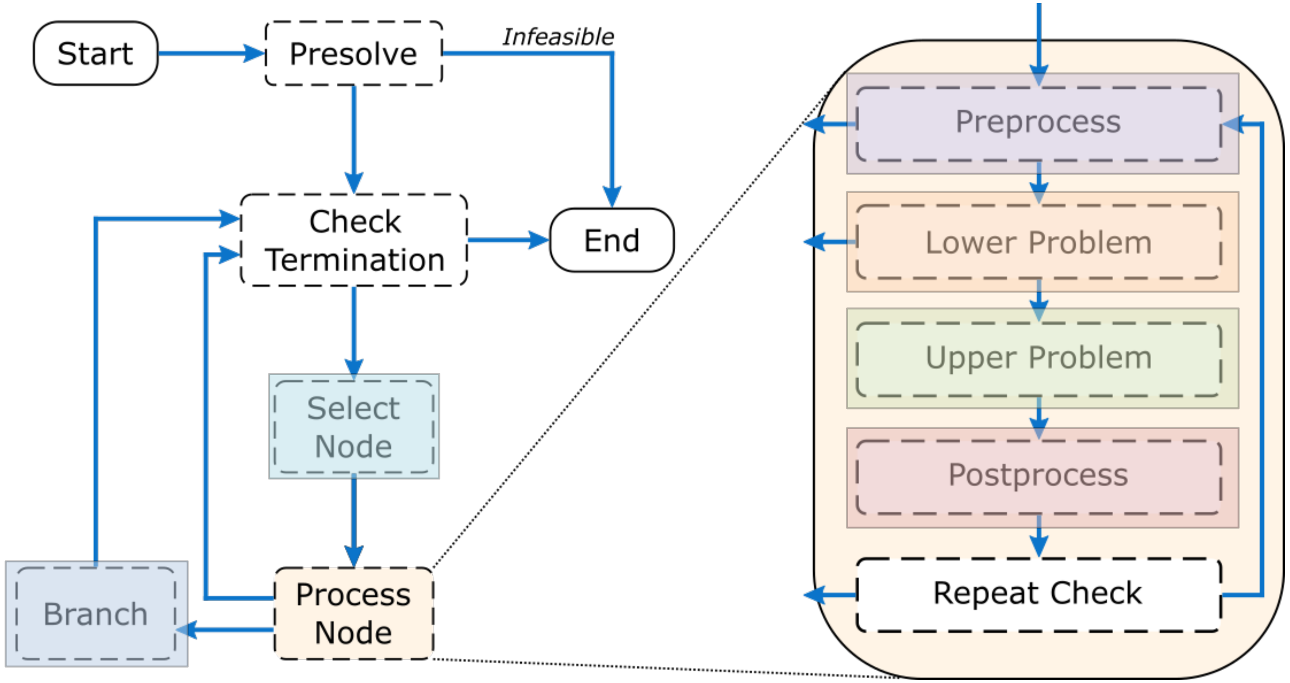
Node Selection: 0.3%

Preprocessing: 0.3%

Lower Problem: 98.9%

Upper Problem: 0.3%

Postprocessing: 0.0%

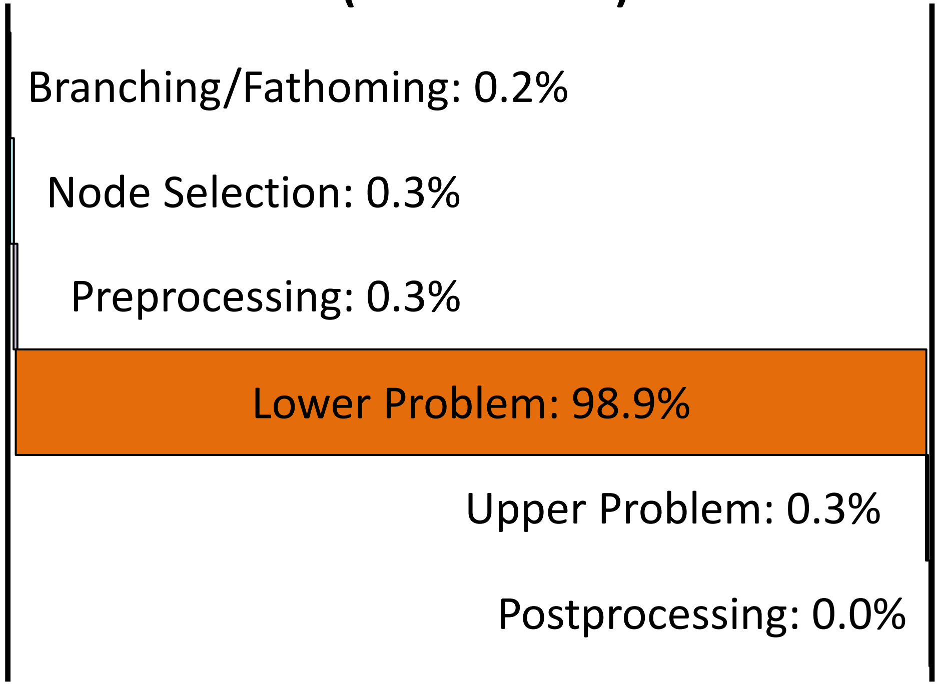


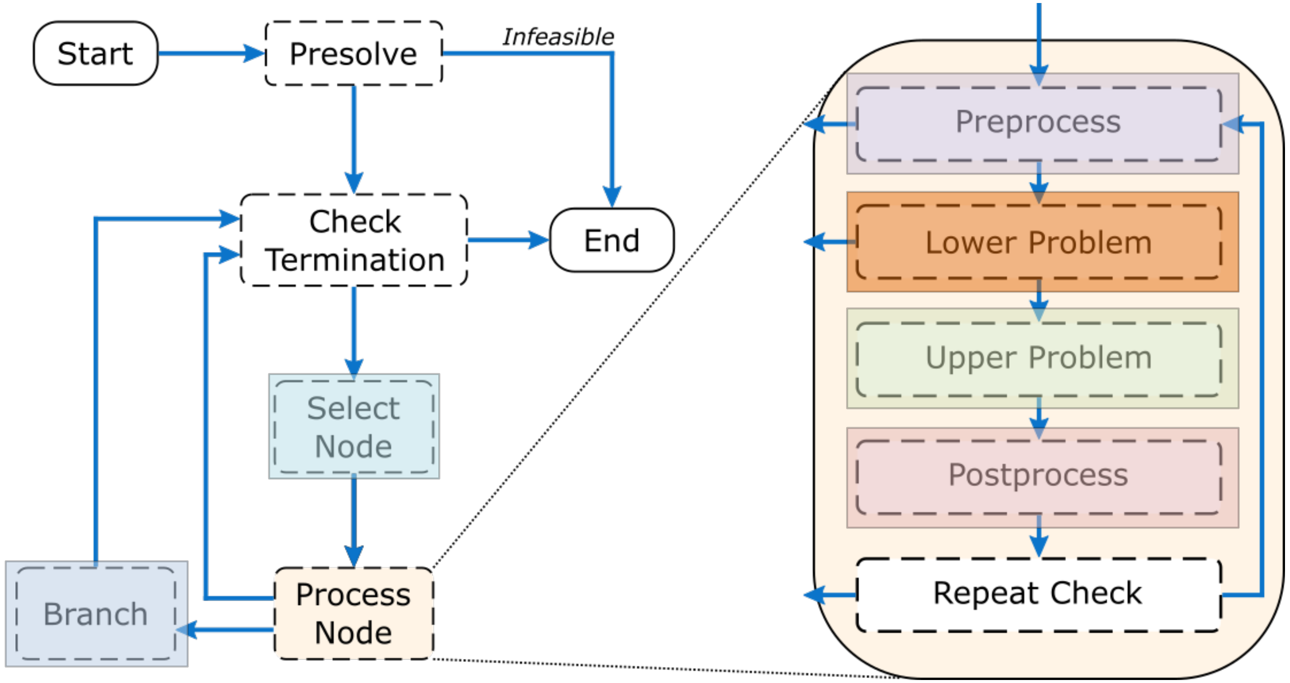
5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

B&B Step Timing

<u>Example 2</u>

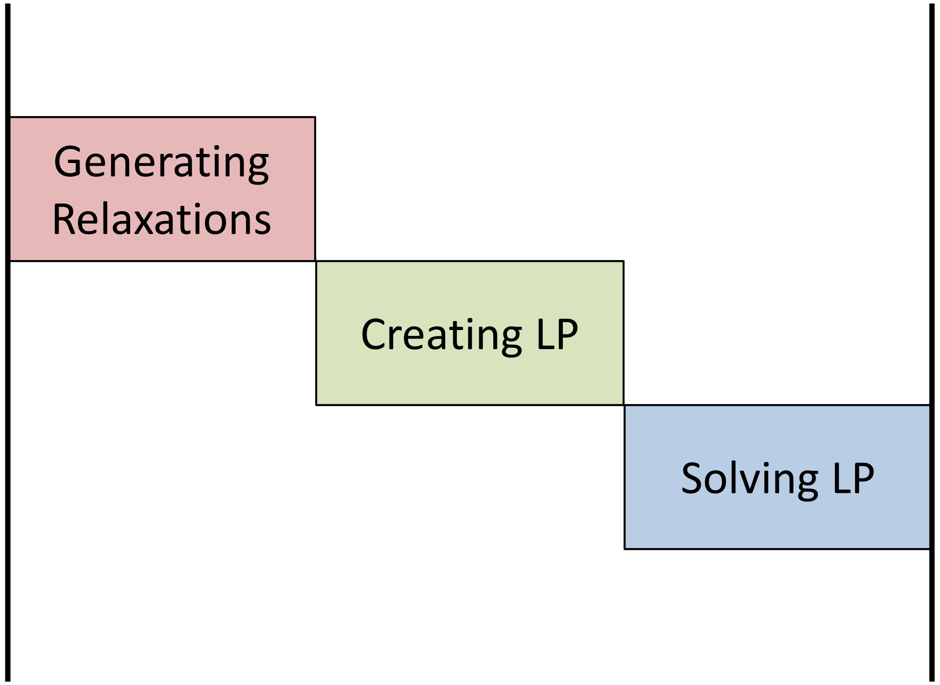
Typical "Long" Problem (>2h runtime)

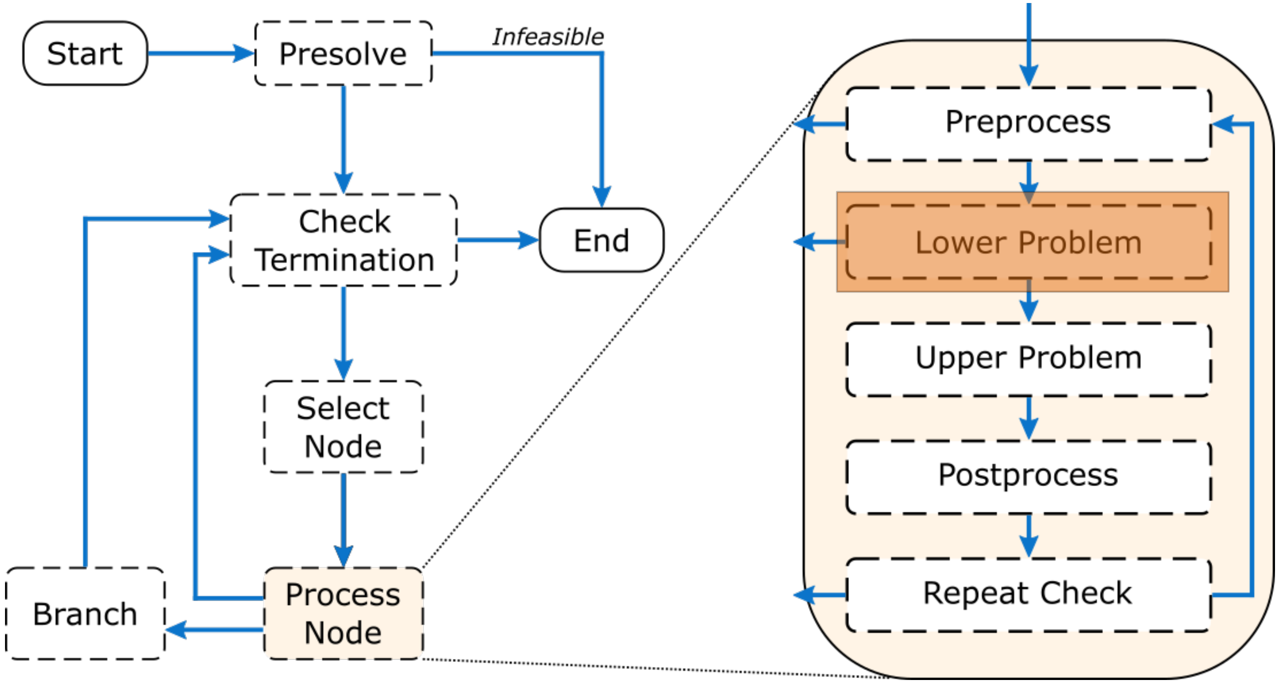




5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

B&B Step Timing





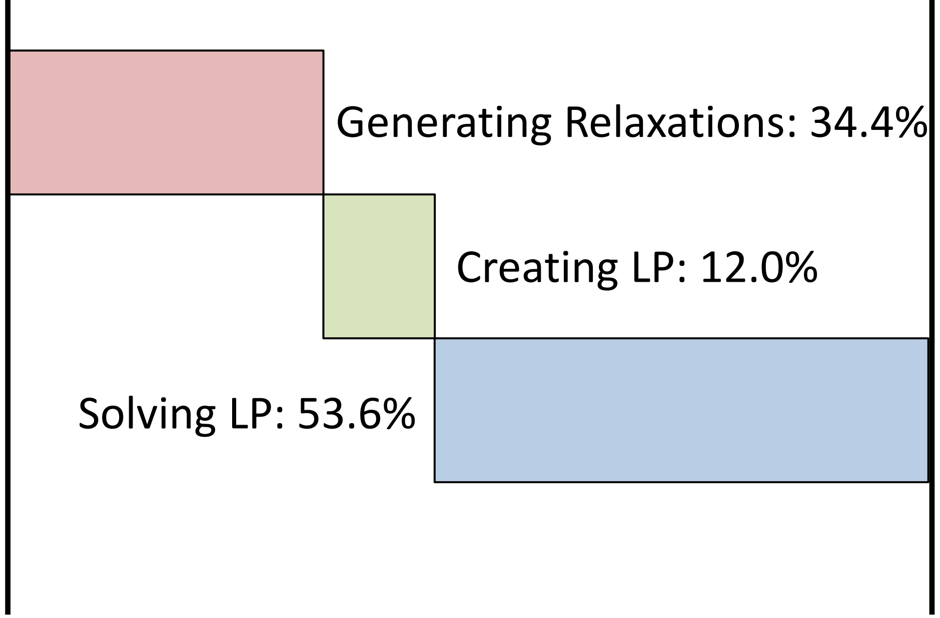
5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

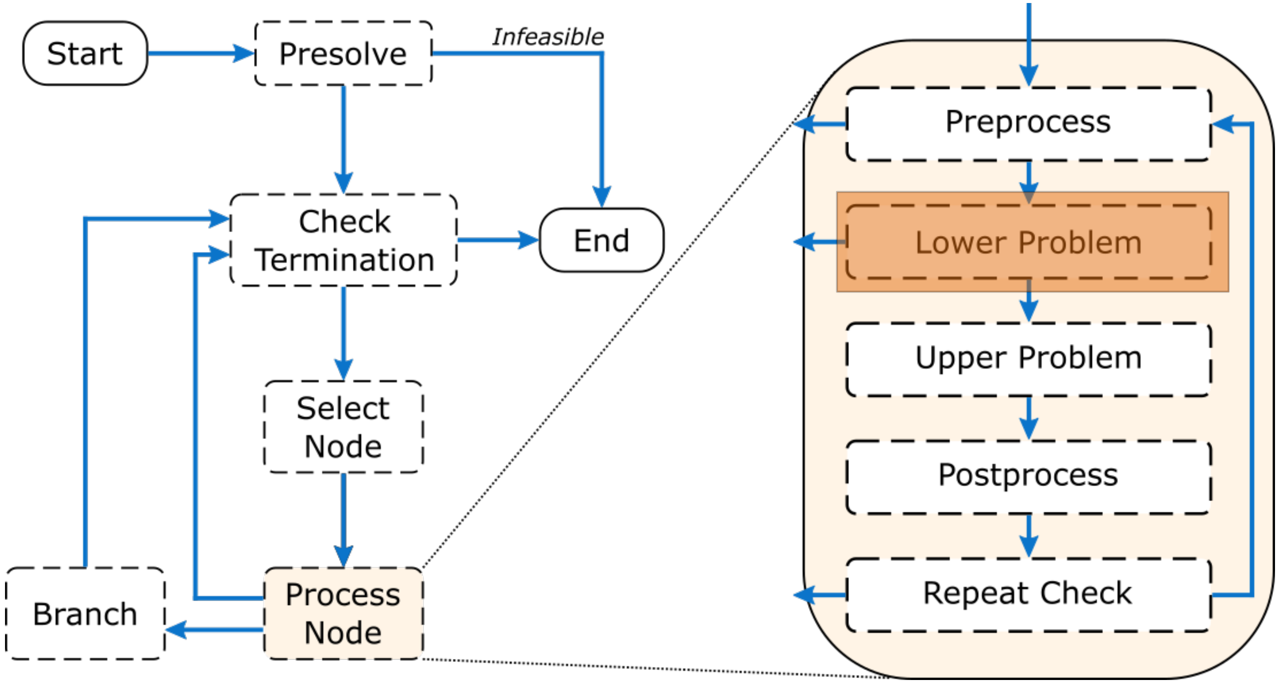
B&B Step Timing

Example 1

Simple Expressions

Higher Dimensionality





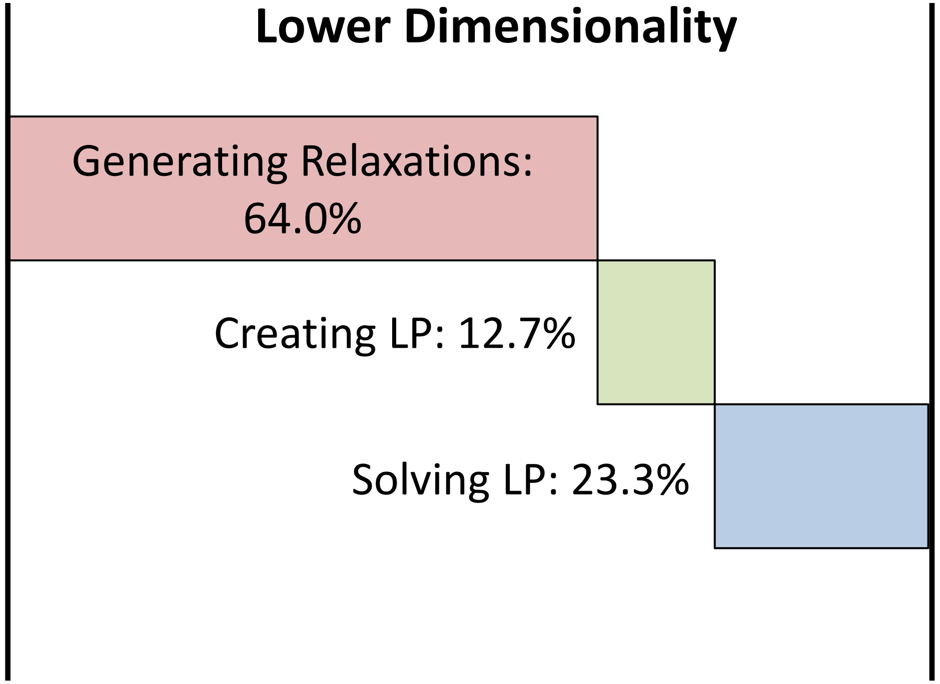
5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

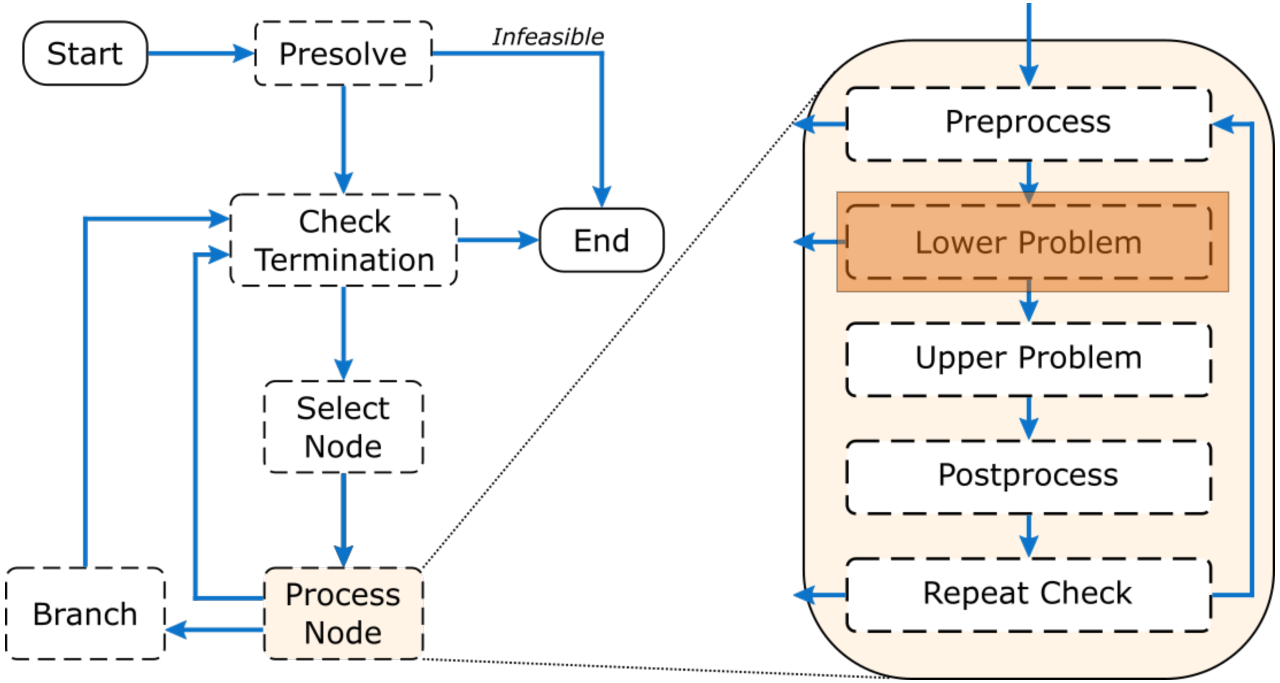
B&B Step Timing

Example 2

Complex Expressions

Lower Dimensionality

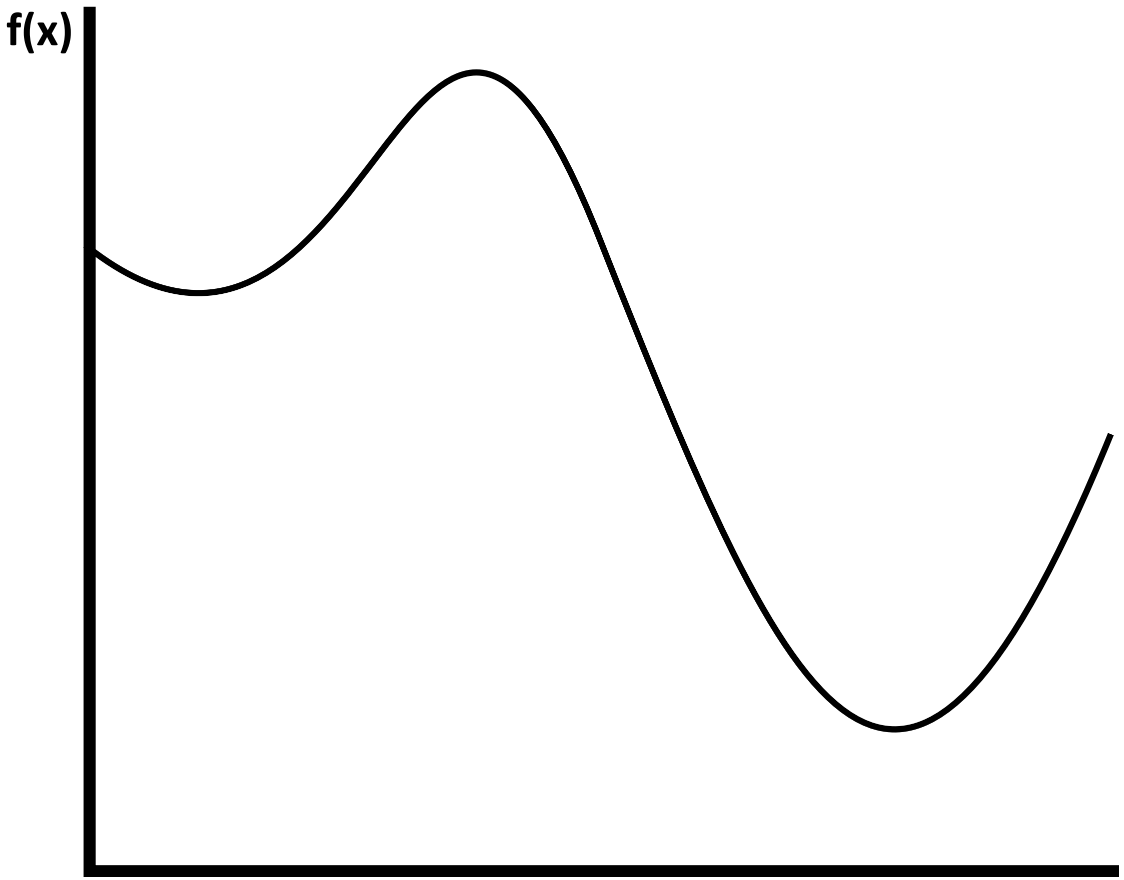




5. Wilhelm, M.E. and Stuber, M.D. **EAGO.jl: easy advanced global optimization in Julia.** *Optimization Methods and Software*, 37(2):425–450, aug 2022. doi:10.1080/10556788.2020.1786566.

GPU Parallelization Targets

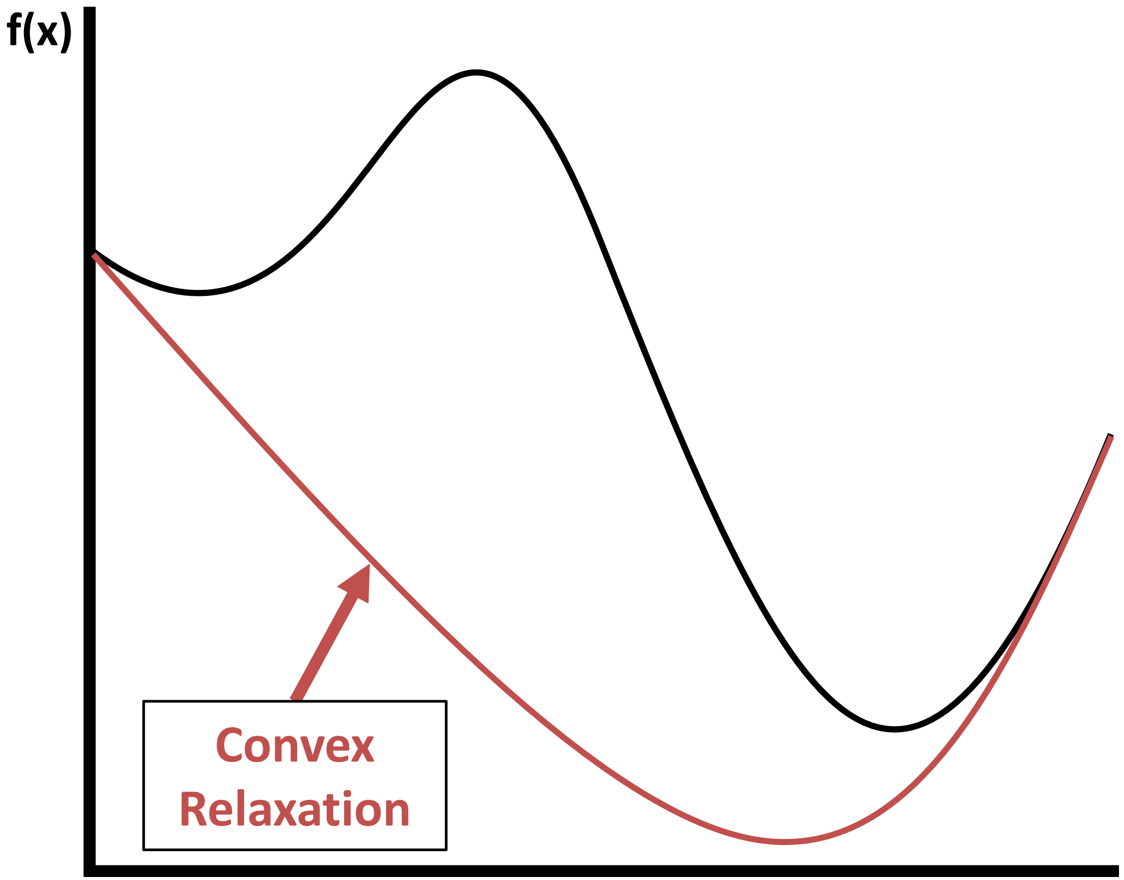
<u>3 Main Parallelization Targets:</u>



GPU Parallelization Targets

3 Main Parallelization Targets:

1. Relaxation Generation

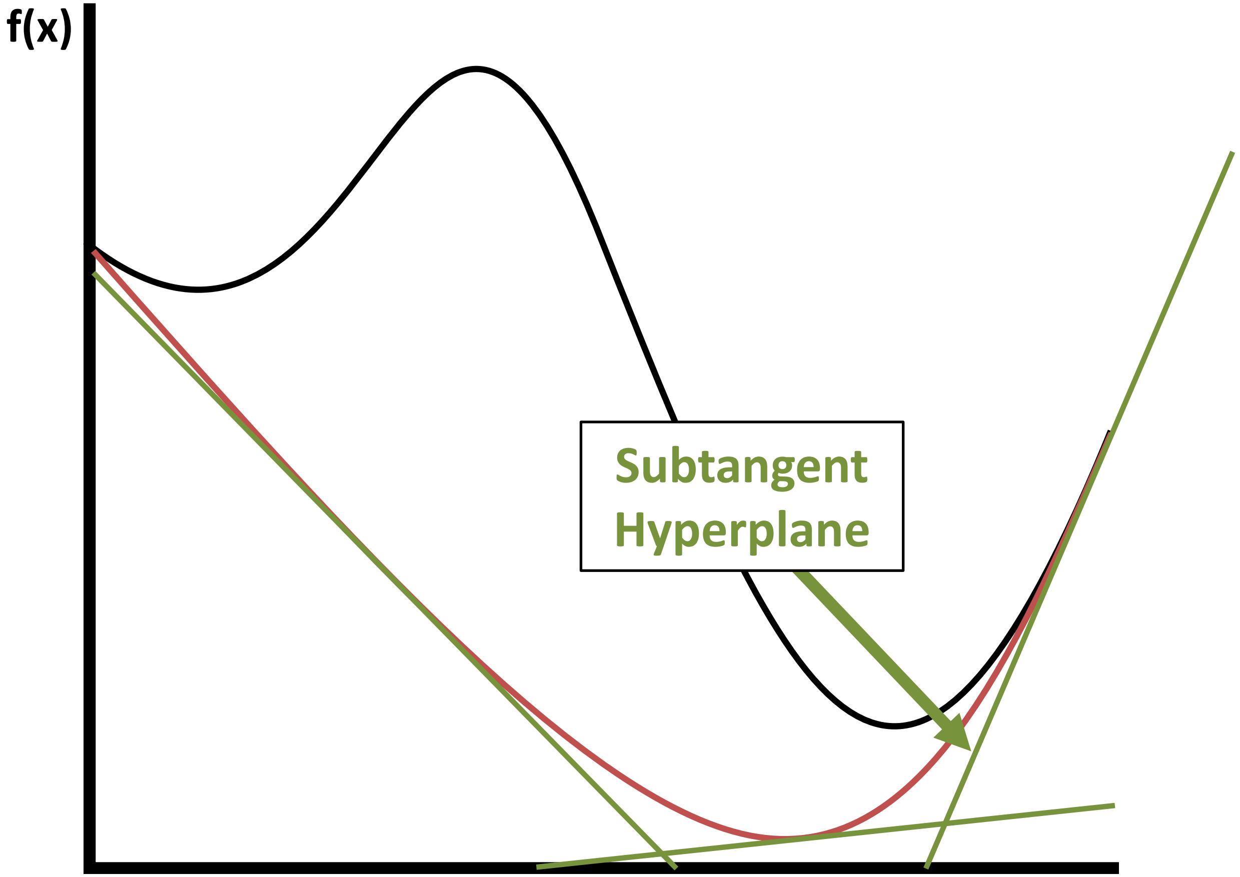


GPU Parallelization Targets

3 Main Parallelization Targets:

1. Relaxation Generation

2. Linear Program (LP) Creation



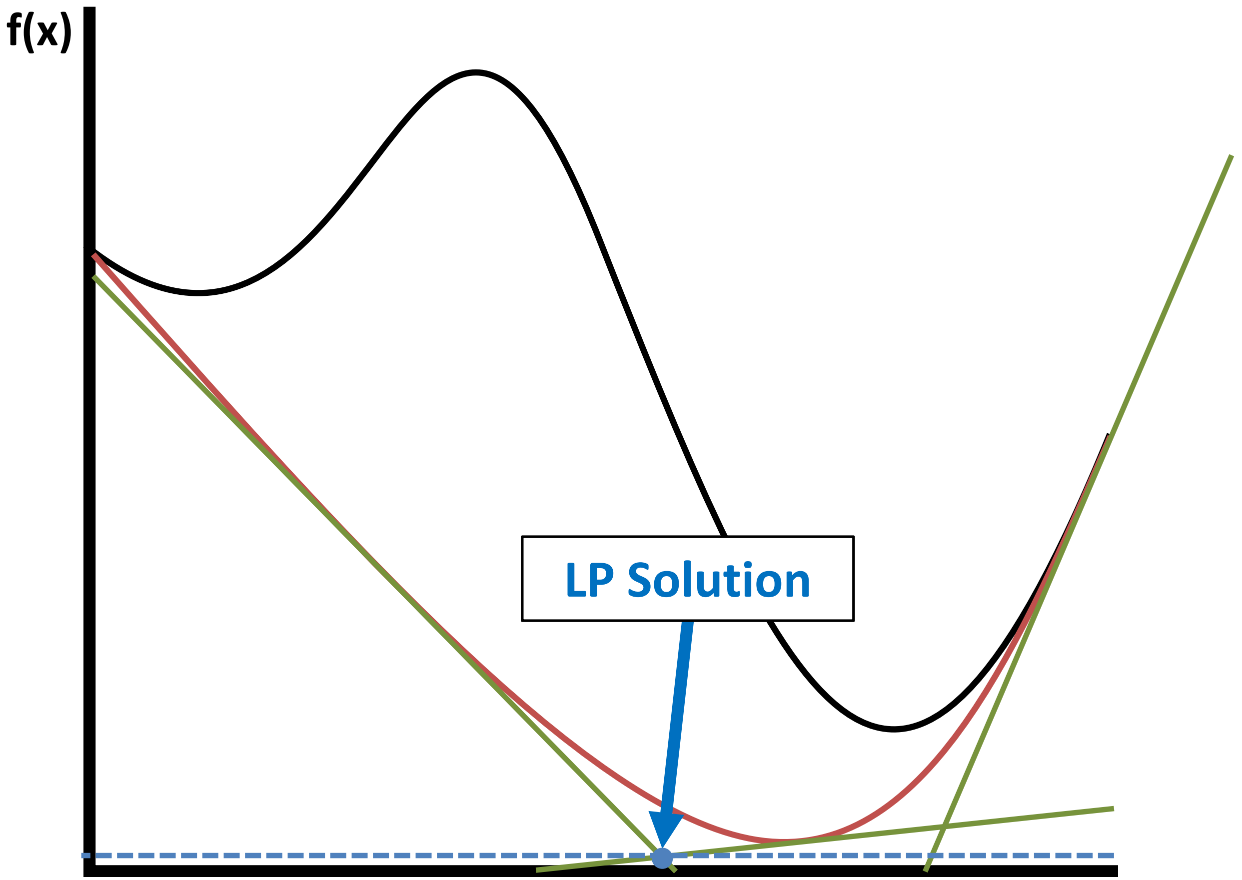
GPU Parallelization Targets

<u>3 Main Parallelization Targets:</u>

1. Relaxation Generation

2. Linear Program (LP) Creation

3. LP Solution

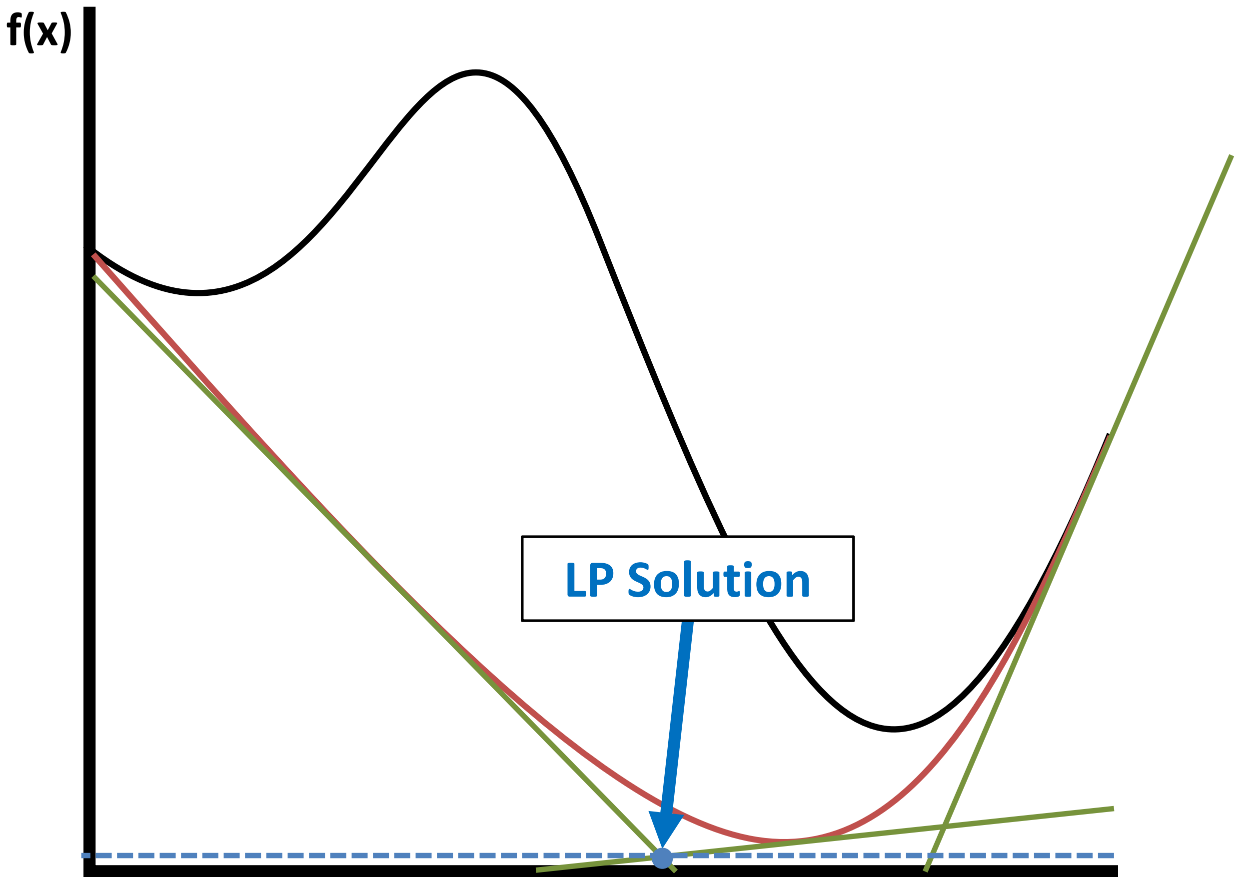


GPU Parallelization Targets

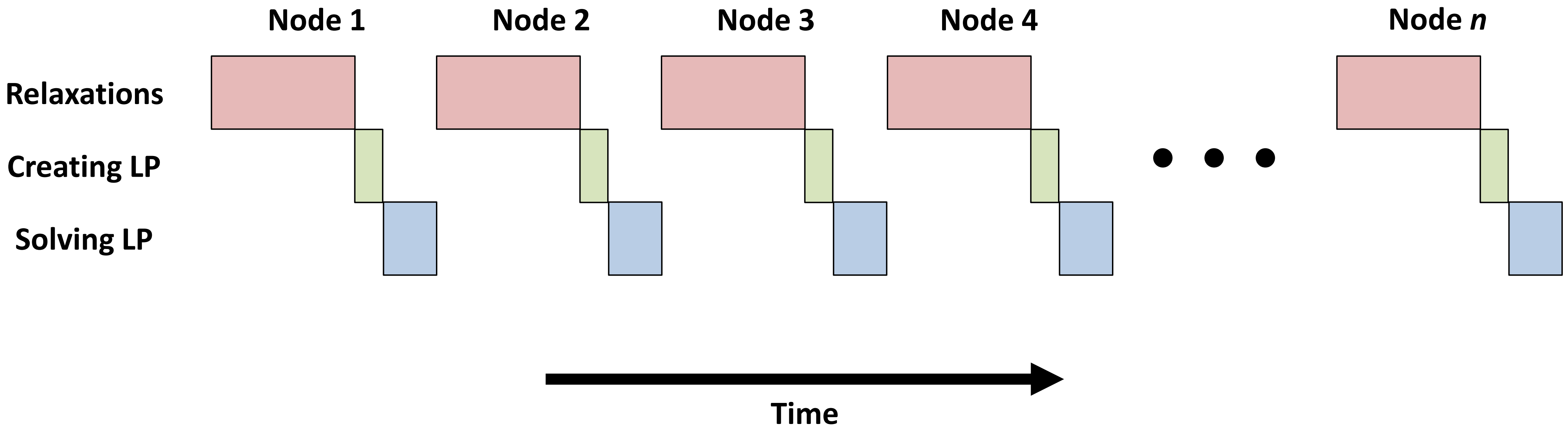
3 Main Parallelization Targets:

1. Relaxation Generation
2. Linear Program (LP) Creation
3. LP Solution

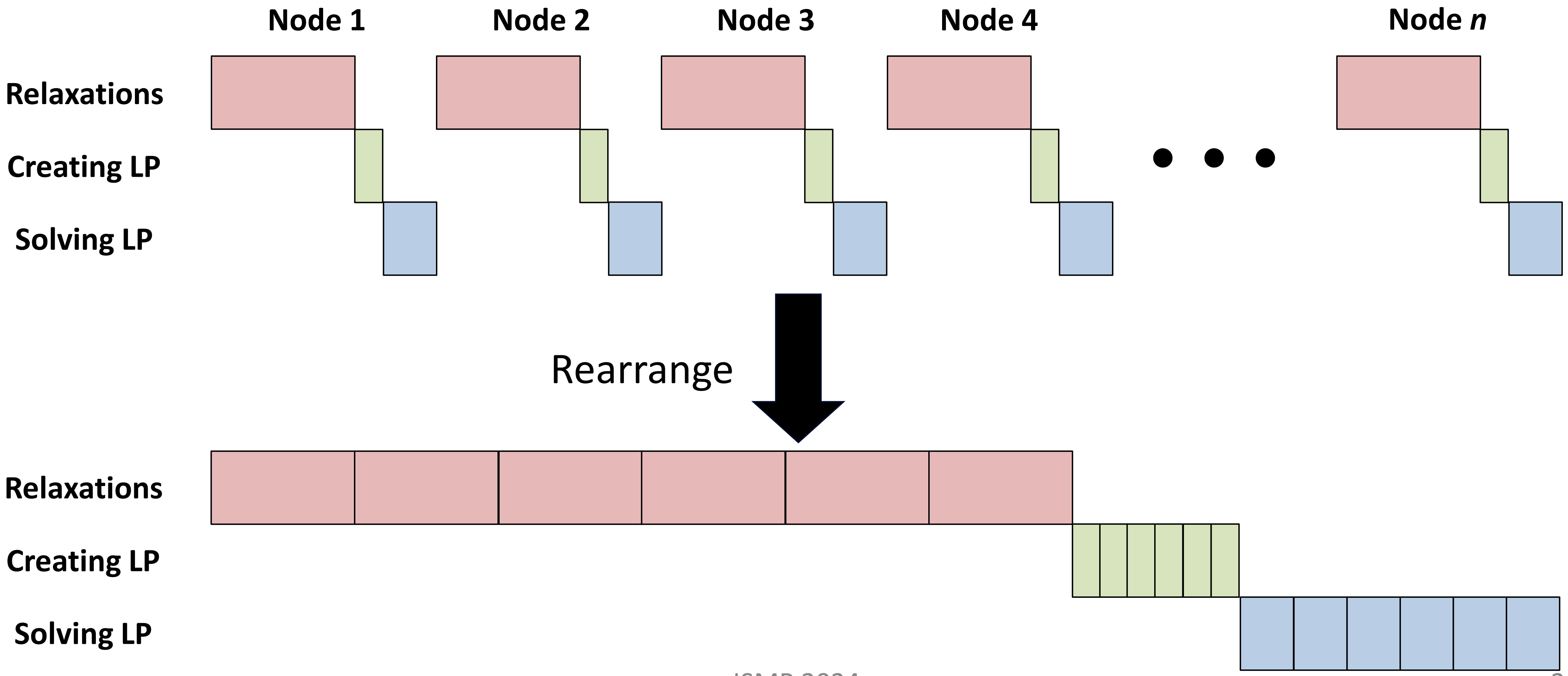
If these tasks can be efficiently parallelized, we can run B&B on powerful GPU hardware



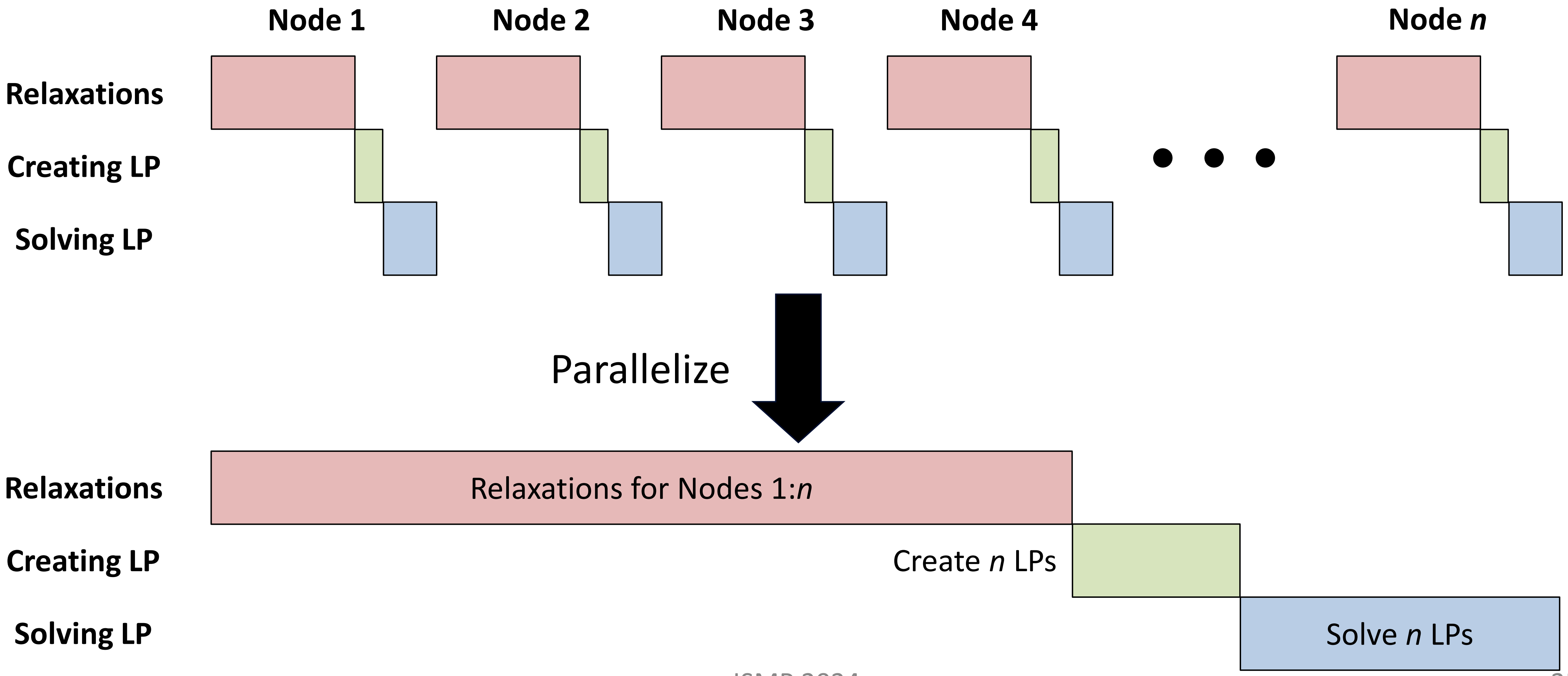
Parallelization Strategy



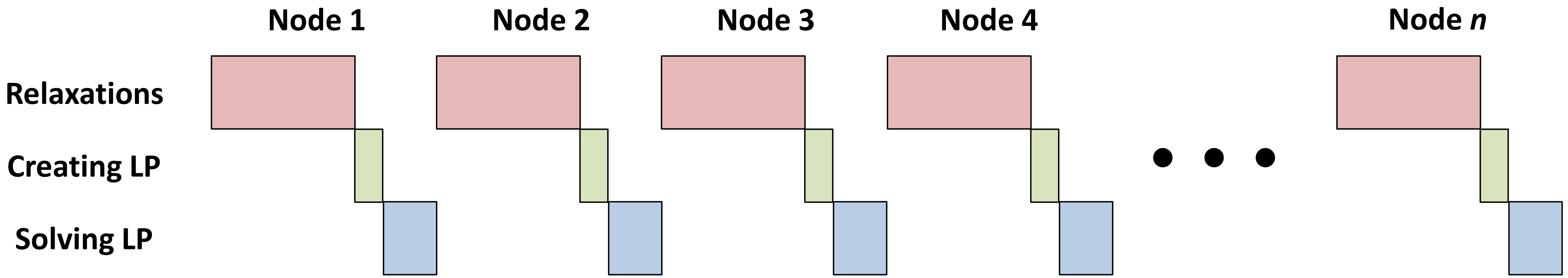
Parallelization Strategy



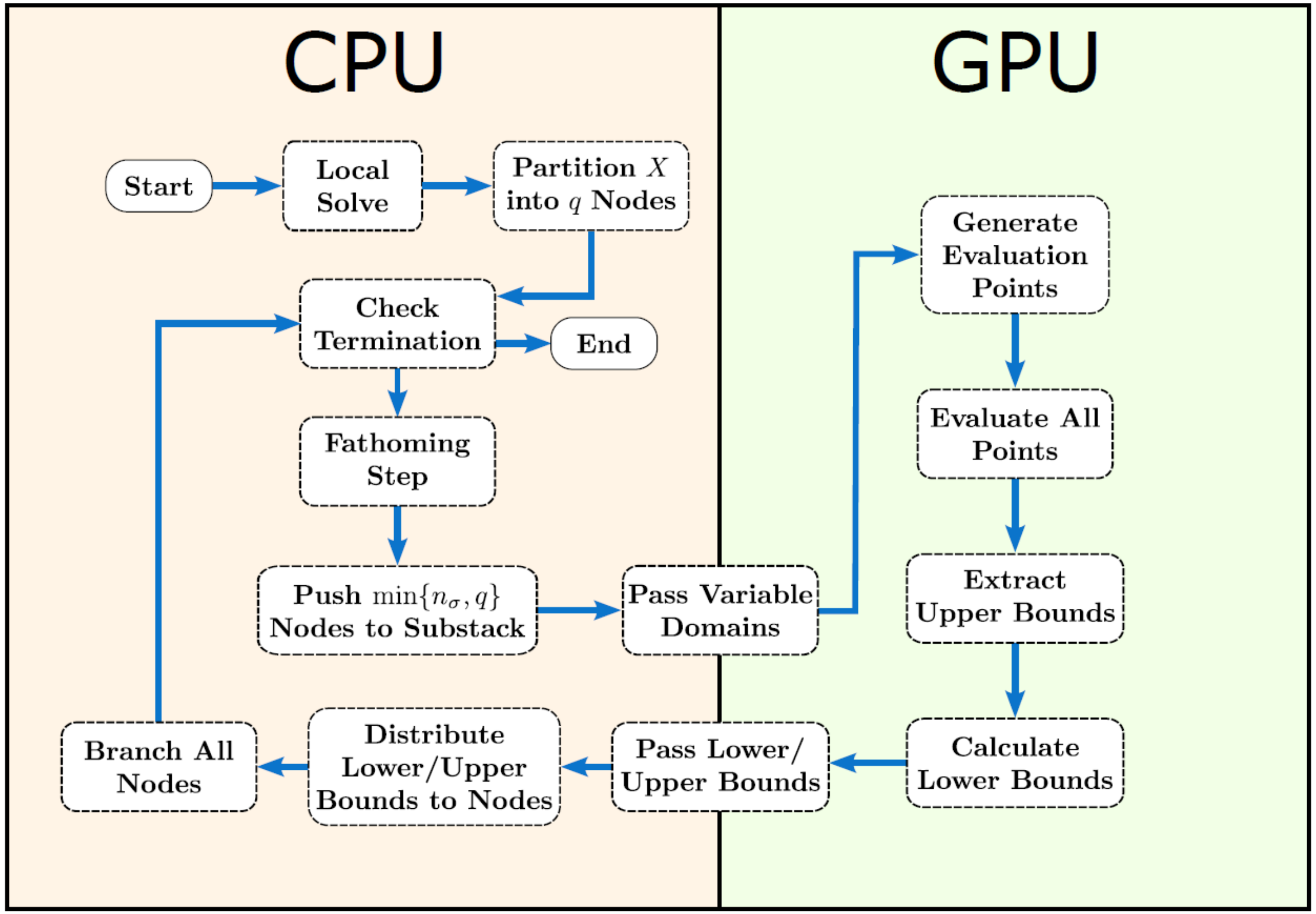
Parallelization Strategy



Parallelization Strategy



ParBB Algorithm



6. Gottlieb, R.X., et al., **Automatic source code generation for deterministic global optimization with parallel architectures**, *Under Review*.

3 Key Parallelization Targets

Relaxations

LP Generation

LP Solves

GPU-Accelerated Relaxations

`SourceCodeMcCormick.jl`

Enables **GPU-compatible** McCormick relaxations through:

- Primal trace generation
- Creation of subfunctions for:
  - Interval extensions
  - Relaxations
  - Subgradients of relaxations
- Connecting subfunctions using generalized McCormick theory
- Constructing **evaluator functions** for original expressions

```
using SourceCodeMcCormick, Symbolics
Symbolics.@variables x, y
expr = (4 + (-2.1+x^2/4)*x^2)*x^2 + x*y + (-4+4*y^2)*y^2
func = fgen(expr, [:cv, :lo, :cvgrad])
```

GPU-Accelerated Relaxations

`SourceCodeMcCormick.jl`

Enables **GPU-compatible** McCormick relaxations through:

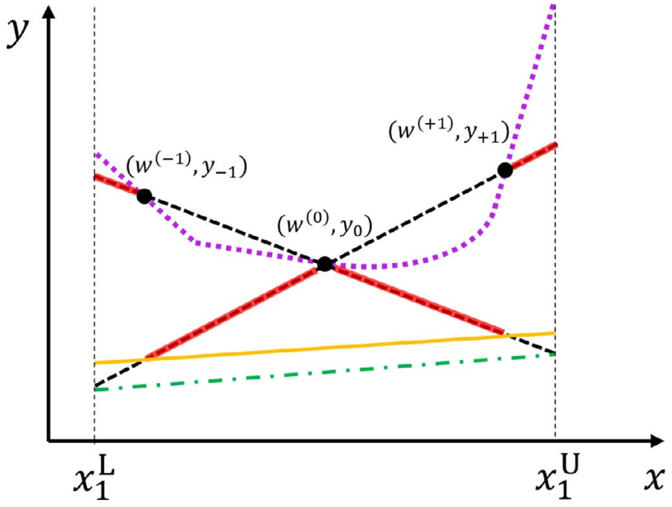
- Primal trace generation
- Creation of subfunctions for:
  - Interval extensions
  - Relaxations
  - **Subgradients of relaxations**
- Connecting subfunctions using generalized McCormick theory
- Constructing **evaluator functions** for original expressions

```
using SourceCodeMcCormick, Symbolics
Symbolics.@variables x, y
expr = (4 + (-2.1+x^2/4)*x^2)*x^2 + x*y + (-4+4*y^2)*y^2
func = fgen(expr, [:cv, :lo, :cvgrad])
```

Subgradient Utility

<u>Before Adding **Subgradients**</u>

- Black-box sampling technique[7] for lower bounds
- No nontrivial constraints



<u>After Adding **Subgradients**</u>

- Tighter, subgradient-based lower-bounding method
- Nontrivial constraints via LP generation

Same method, but on **GPU**

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

3 Key Parallelization Targets

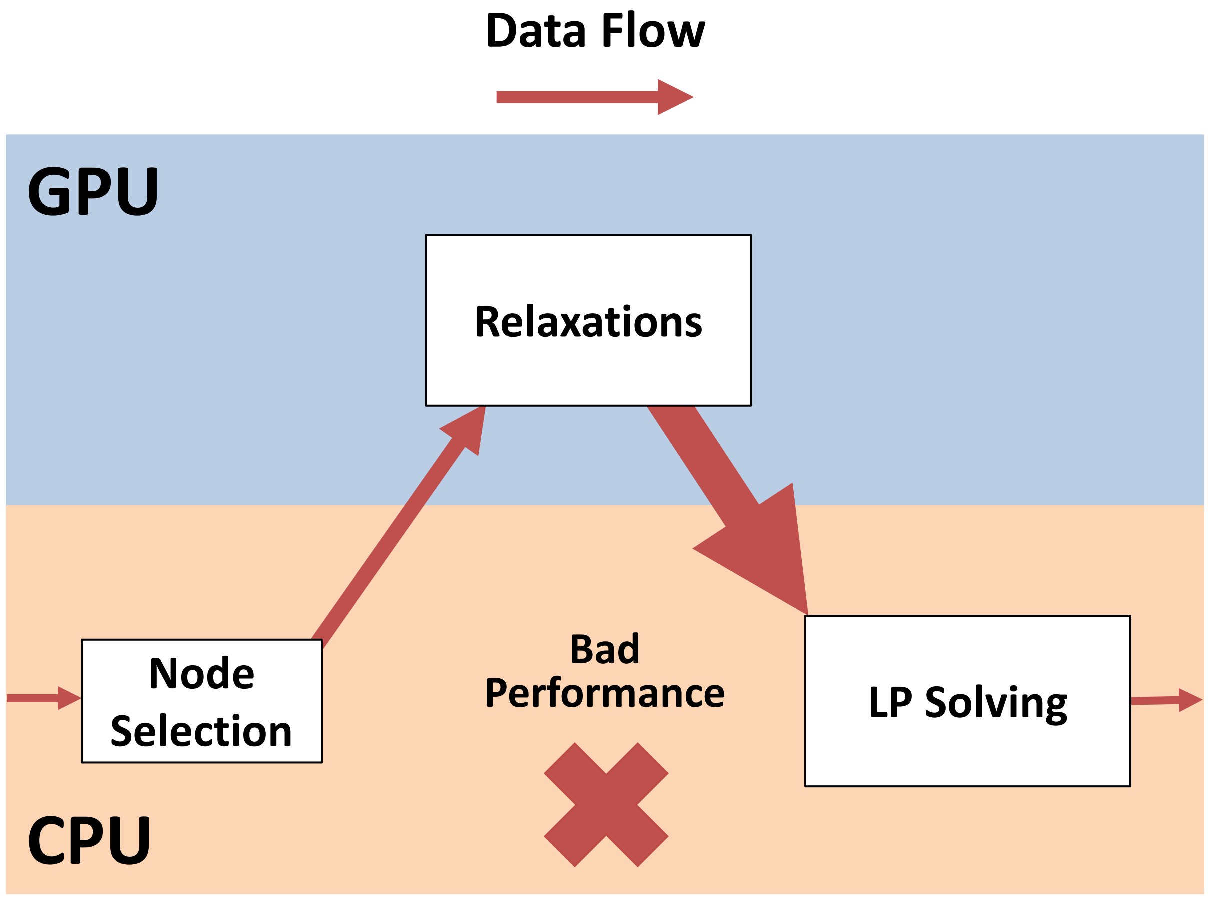
Relaxations ✓

LP Generation

LP Solves

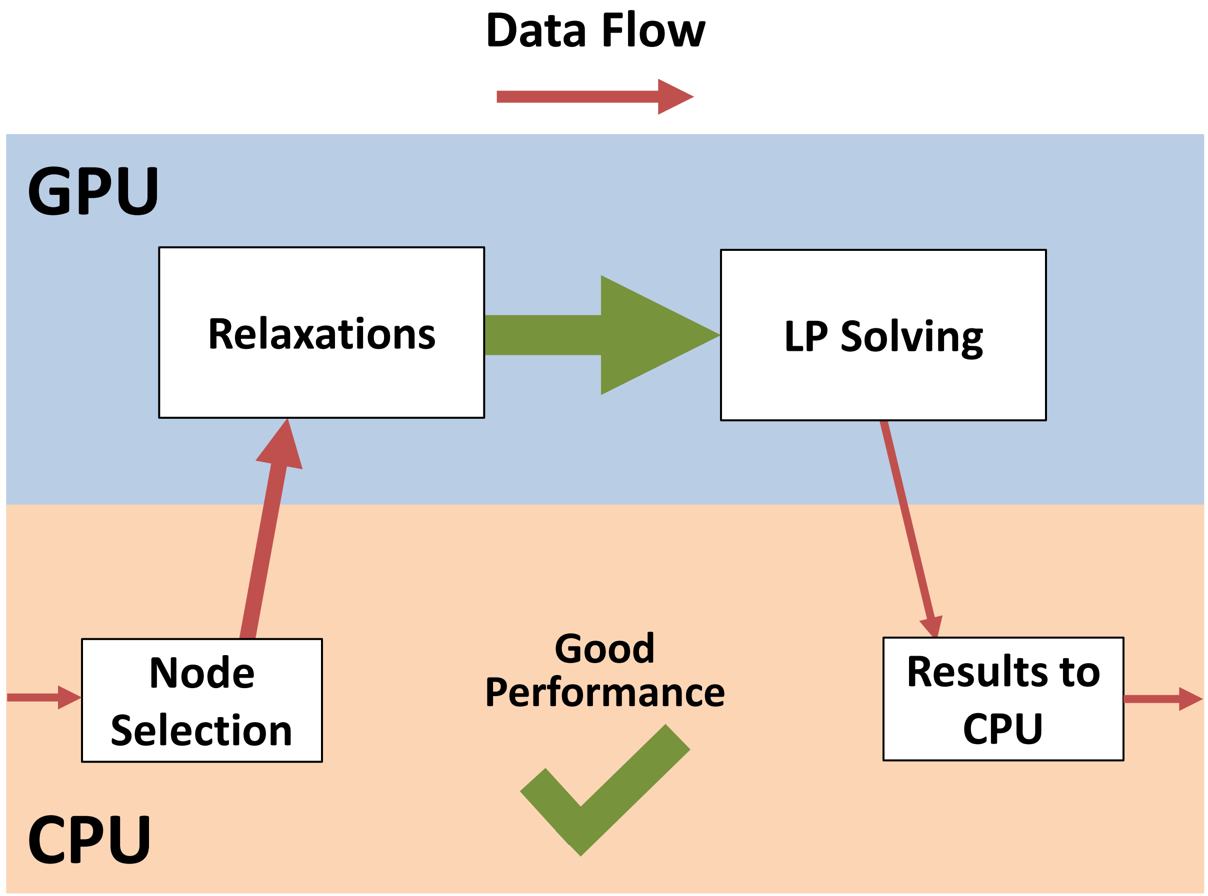
We Need GPU LPs!

- Calculating relaxations on GPUs is fast, but creates **lots of data**
  - Relaxations; intervals; subgradients
- **Memory transfer overhead** could negate benefits of GPU relaxations



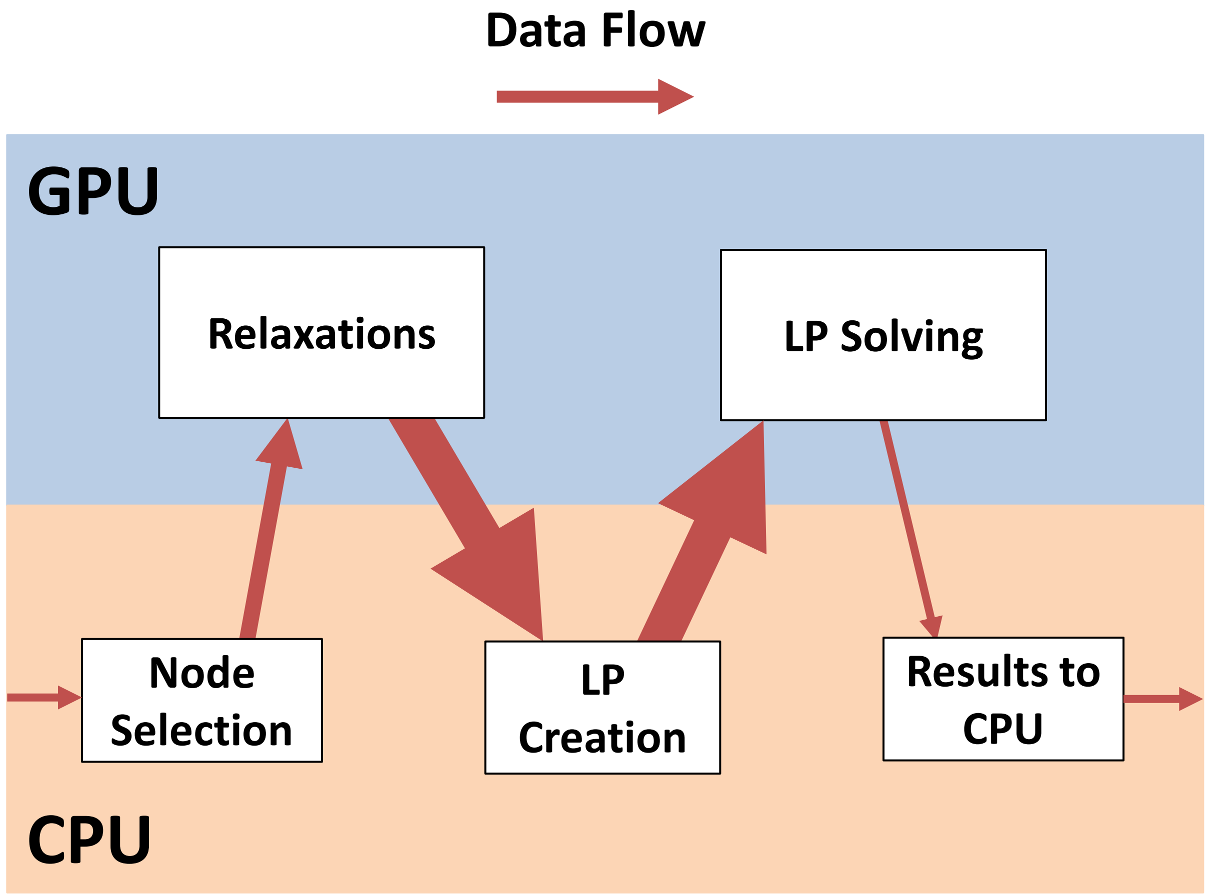
We Need GPU LPs!

- Calculating relaxations on GPUs is fast, but creates **lots of data**
  - Relaxations; intervals; subgradients
- **Memory transfer overhead** could negate benefits of GPU relaxations
- **GPU-based LP solver is needed**



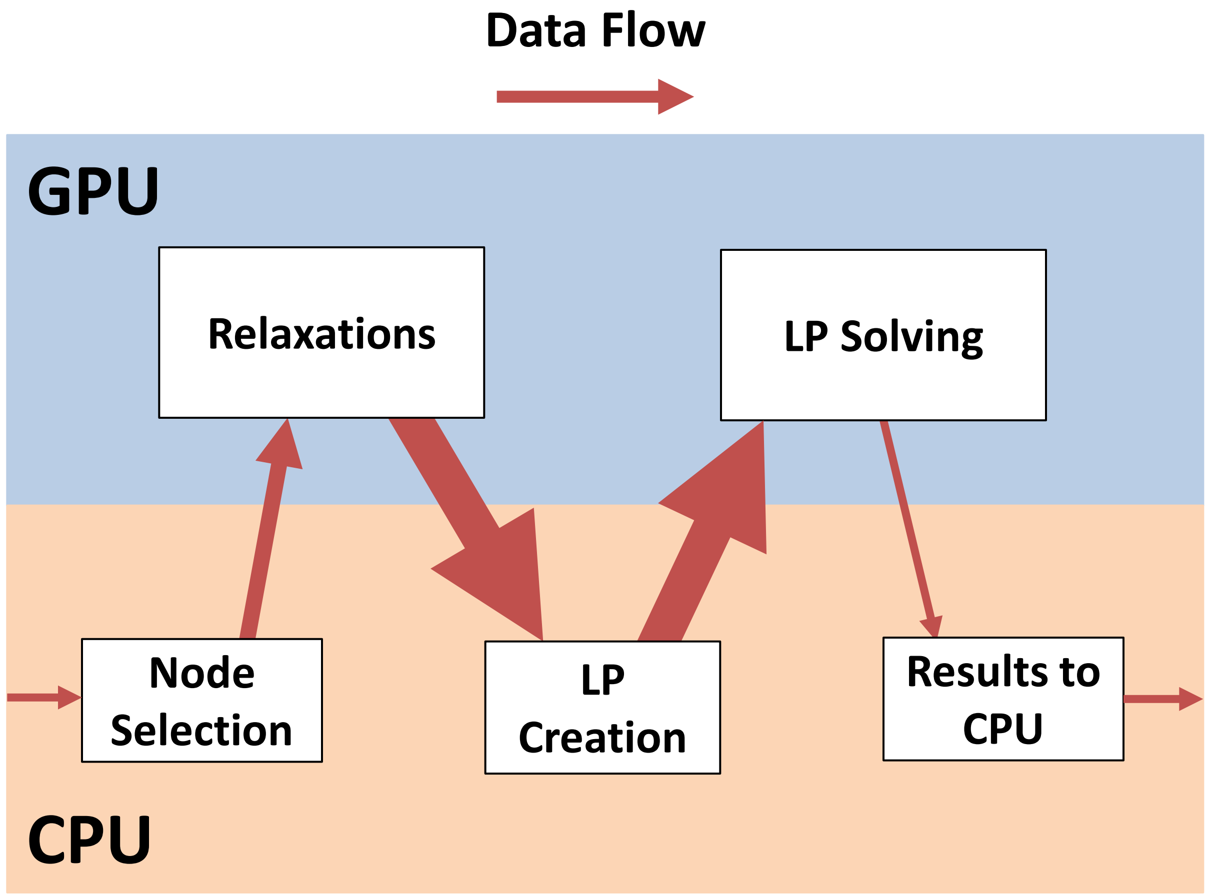
We Need GPU LPs!

- Calculating relaxations on GPUs is fast, but creates **lots of data**
  - Relaxations; intervals; subgradients
- **Memory transfer overhead** could negate benefits of GPU relaxations
- **GPU-based LP solver is needed**
  - LP creation must also be on GPU



We Need GPU LPs!

- Calculating relaxations on GPUs is fast, but creates **lots of data**
  - Relaxations; intervals; subgradients
- **Memory transfer overhead** could negate benefits of GPU relaxations
- **GPU-based LP solver is needed**
  - LP creation must also be on GPU
    - Need a **custom LP solver**



LPs on GPUs?

Does Gurobi support GPUs?

Greg Glockner

10 months ago · Updated

Follow

The Gurobi development team is watching GPUs (Graphics Processing Units) closely, but up to this point, all of the evidence indicates that they aren't well suited to the needs of an LP/MIP/QP solver. Specifically:

- GPUs don't work well for sparse linear algebra, which dominates much of linear programming. GPUs rely on keeping hundreds or even thousands of independent processors busy at a time. The extremely sparse matrices that are typical in linear programming don't admit nearly that level of parallelism.
- GPUs are built around SIMD computations, where all processors perform the same instruction in each cycle (but on different data). Parallel MIP explores different sections of the search tree on different processors. The computations required at different nodes in the search tree are quite different, so SIMD computation is not well suited to the needs of parallel MIP.

Note that CPUs and GPUs are both improving parallelism as a means to increase performance. The Gurobi Optimizer is designed to effectively exploit multiple cores in a CPU, so you'll definitely see a benefit from more parallelism in the future.

GUROBI OPTIMIZATION

Gurobi.com

LPs on GPUs?

Does Gurobi support GPUs?

Greg Glockner
10 months ago · Updated

Follow

all of the evidence indicates that they aren't well suited to the needs of an LP/MIP/QP solver.

all of the evidence indicates that they aren't well suited to the needs of an LP/MIP/QP solver. Specifically:

- GPUs don't work well for sparse linear algebra, which dominates much of linear programming. GPUs rely on keeping hundreds or even thousands of independent processors busy at a time. The extremely sparse matrices that are typical in linear programming don't admit nearly that level of parallelism.
- GPUs are built around SIMD computations, where all processors perform the same instruction in each cycle (but on different data). Parallel MIP explores different sections of the search tree on different processors. The computations required at different nodes in the search tree are quite different, so SIMD computation is not well suited to the needs of parallel MIP.

Note that CPUs and GPUs are both improving parallelism as a means to increase performance. The Gurobi Optimizer is designed to effectively exploit multiple cores in a CPU, so you'll definitely see a benefit from more parallelism in the future.

GUROBI OPTIMIZATION

Gurobi.com

LPs on GPUs?

Does Gurobi support GPUs?

Greg Glockner
10 months ago · Updated

Follow

all of the evidence indicates that they aren't well suited to the needs of an LP/MIP/QP solver.

GPUs don't work well for sparse linear algebra,

GUROBI OPTIMIZATION

Gurobi.com

LPs on GPUs?

Does Gurobi support GPUs?

Greg Glockner
10 months ago · Updated

Follow

all of the evidence indicates that they aren't well suited to the needs of an LP/MIP/QP solver.

GPUs don't work well for sparse linear algebra,

GUROBI
OPTIMIZATION

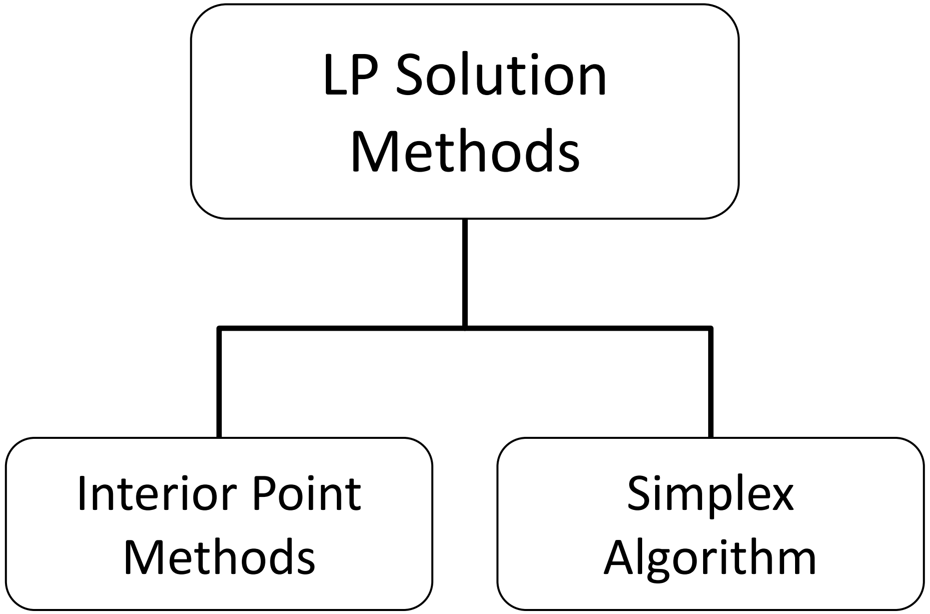
Gurobi.com

Linear subproblems in deterministic global optimization are small and dense

Custom LP Solver

Goal is to parallelize LPs that are all:

- Small
- Dense
- The same size
- Of similar complexity

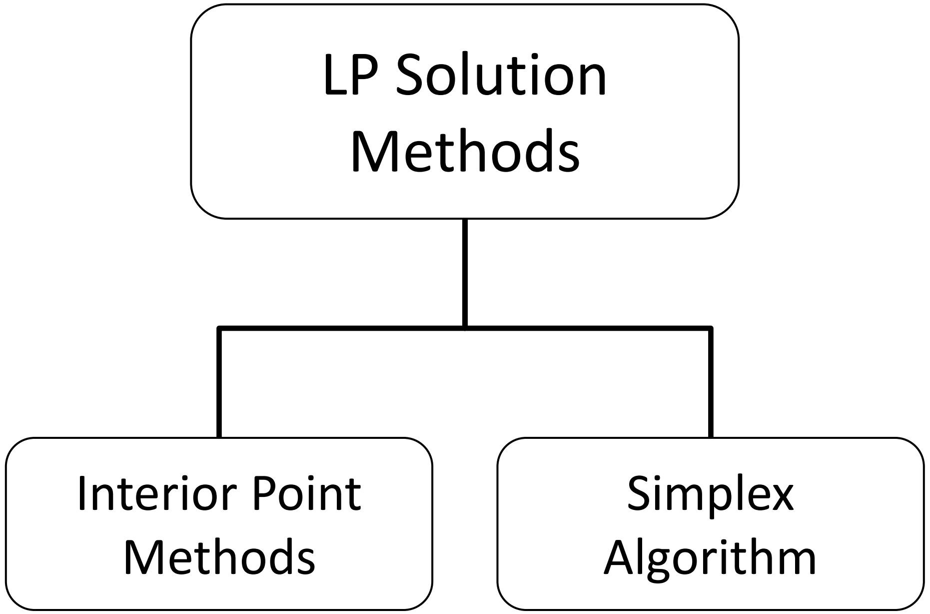


Custom LP Solver

Goal is to parallelize LPs that are all:

- Small
- Dense
- The same size
- Of similar complexity

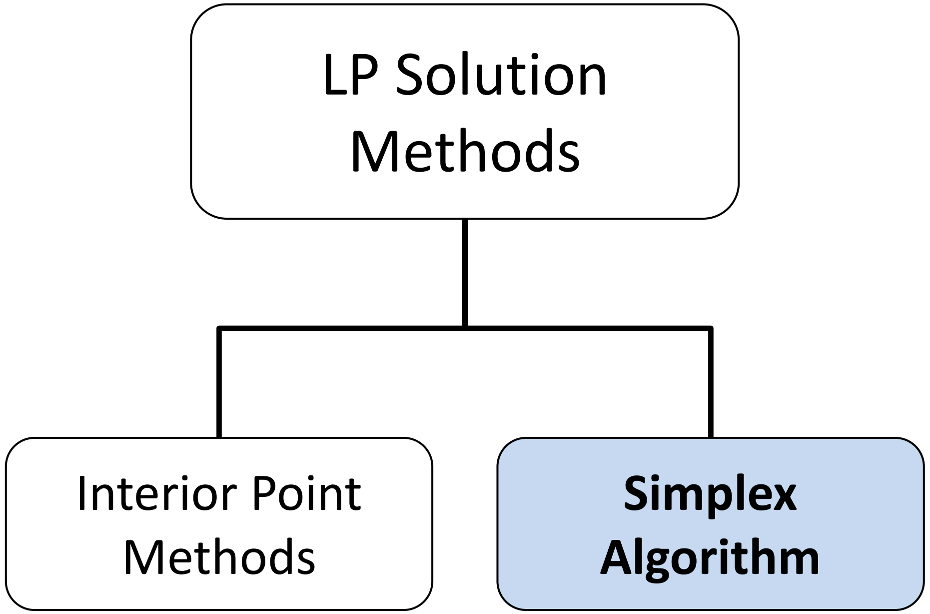
Typical heuristics for GPU-based LP methods may not apply



Custom LP Solver

Implementing **two-phase Simplex method**

- Intuitive
- Simple to set up tableau(s)
- Straightforward to find BFS(s)
- No matrix inversion needed



Custom LP Solver

Implementing **two-phase Simplex method**

- Intuitive
- Simple to set up tableau(s)
- Straightforward to find BFS(s)
- No matrix inversion needed

- Not the only solution! Other methods may work as well (or better!)

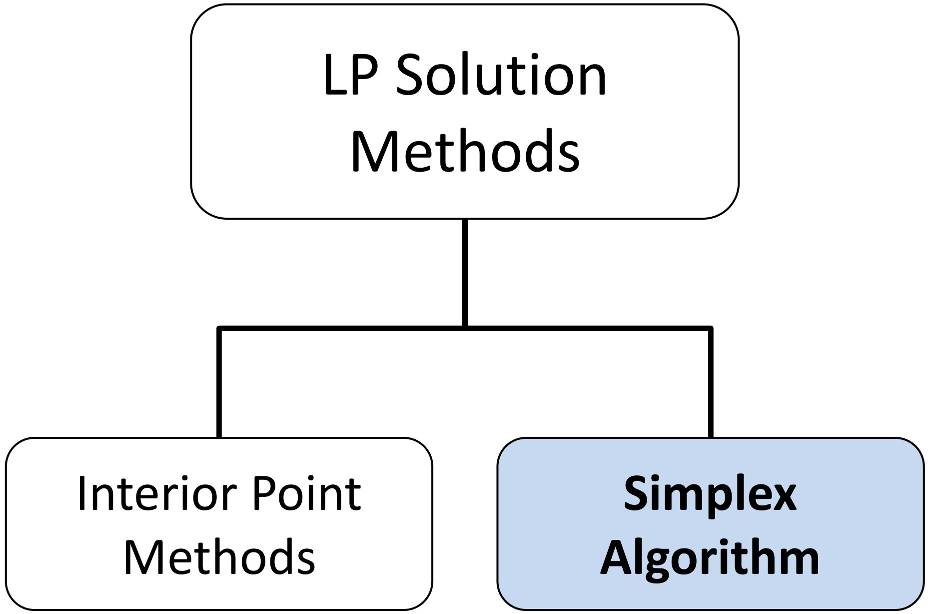


Tableau Generation

$$\min f_{cv}(x)$$

$$\begin{aligned}
\text{s.t.}\ \ 0.8x + 12 &\le f_{cv}(x)\\
1.6x + \ \ 4 &\le f_{cv}(x)\\
\text{-}2.8x + 24 &\le f_{cv}(x)\\
x_L \le x &\le x_U
\end{aligned}$$

- Shifting/scaling domain
- Epigraph reformulation
- Restructuring

$$\min \xi^+ - \xi^-$$

$$\begin{aligned}
\text{s.t.} -\xi^+ + \xi^- \qquad\qquad &\ge f_L\\
-\xi^+ + \xi^- + 0.2x &\le -1\\
-\xi^+ + \xi^- + 0.4x &\le 1\\
-\xi^+ + \xi^- - 0.7x &\le -4\\
0 \le x &\le 1\\
0 \le \xi^+ &\le \text{Inf}\\
0 \le \xi^- &\le \text{Inf}
\end{aligned}$$

Tableau Generation

n LPs

$$\begin{aligned} &\min f_{cv}(x) \\ \text{s.t.}\ & 0.8x + 12 \le f_{cv}(x) \\ & 1.6x + 4 \le f_{cv}(x) \\ & \text{-}2.8x + 24 \le f_{cv}(x) \\ & x_L \le x \le x_U \end{aligned}$$

• • •

$$\begin{aligned} &\min f_{cv}(x) \\ \text{s.t.}\ & -1.6x + 8 \le f_{cv}(x) \\ & -0.4x + 12 \le f_{cv}(x) \\ & \text{-}4.4x + 16 \le f_{cv}(x) \\ & x_L \le x \le x_U \end{aligned}$$

- ➢ Shifting/scaling domain
- ➢ Epigraph reformulation
- ➢ Restructuring

$$\begin{aligned} &\min \xi^+ - \xi^- \\ \text{s.t.}\ & -\xi^+ + \xi^- \ge f_L \\ & -\xi^+ + \xi^- + 0.2x \le -1 \\ & -\xi^+ + \xi^- + 0.4x \le 1 \\ & -\xi^+ + \xi^- - 0.7x \le -4 \\ & 0 \le x \le 1 \\ & 0 \le \xi^+ \le \text{Inf} \\ & 0 \le \xi^- \le \text{Inf} \end{aligned}$$

• • •

$$\begin{aligned} &\min \xi^+ - \xi^- \\ \text{s.t.}\ & -\xi^+ + \xi^- \ge f_L \\ & -\xi^+ + \xi^- - 0.4x \le -2 \\ & -\xi^+ + \xi^- - 0.1x \le 3 \\ & -\xi^+ + \xi^- - 1.1x \le 2 \\ & 0 \le x \le 1 \\ & 0 \le \xi^+ \le \text{Inf} \\ & 0 \le \xi^- \le \text{Inf} \end{aligned}$$

n LPs

Tableau Generation

$$\begin{aligned}
&\min\ \xi^+ - \xi^- \\
&\text{s.t.} -\xi^+ + \xi^- \geq f_L \\
&\quad -\xi^+ + \xi^- + 0.2x \leq -1 \\
&\quad -\xi^+ + \xi^- + 0.4x \leq 1 \\
&\quad -\xi^+ + \xi^- - 0.7x \leq -4 \\
&\quad 0 \leq x \leq 1 \\
&\quad 0 \leq \xi^+ \leq \text{Inf} \\
&\quad 0 \leq \xi^- \leq \text{Inf}
\end{aligned}$$

• • •

$$\begin{aligned}
&\min\ \xi^+ - \xi^- \\
&\text{s.t.} -\xi^+ + \xi^- \geq f_L \\
&\quad -\xi^+ + \xi^- - 0.4x \leq -2 \\
&\quad -\xi^+ + \xi^- - 0.1x \leq 3 \\
&\quad -\xi^+ + \xi^- - 1.1x \leq 2 \\
&\quad 0 \leq x \leq 1 \\
&\quad 0 \leq \xi^+ \leq \text{Inf} \\
&\quad 0 \leq \xi^- \leq \text{Inf}
\end{aligned}$$

	ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
LP #1	0	0	1	1	0	0	0	0	0	0	0	0	0	1
	-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
	1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
	-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
	1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
	-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
• • •	…	…	…	…	…	…	…	…	…	…	…	…	…	…
LP #n	0	0	1	1	0	0	0	0	0	0	0	0	0	1
	-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
	1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
	-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
	-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
	-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

Tableau Generation

$$
\begin{aligned}
\min\ & \xi^+ - \xi^- \\
\text{s.t.}\ & -\xi^+ + \xi^- \geq f_L \\
& -\xi^+ + \xi^- + 0.2x \leq -1 \\
& -\xi^+ + \xi^- + 0.4x \leq 1 \\
& -\xi^+ + \xi^- - 0.7x \leq -4 \\
& 0 \leq x \leq 1 \\
& 0 \leq \xi^+ \leq \text{Inf} \\
& 0 \leq \xi^- \leq \text{Inf}
\end{aligned}
$$

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
…	…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

Tableau Generation

*n**[height of one LP]

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
…	…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

3 Key Parallelization Targets

- **Relaxations**
- **LP Generation**
- **LP Solves**

GPU Simplex

<u>Parallelization approach depends on step:</u>

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- Access column information
- Find pivot column

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	[illegible]	[illegible]	[illegible]	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	[illegible]	[illegible]	[illegible]	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	[illegible]	[illegible]	[illegible]	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	[illegible]	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	[illegible]	[illegible]	[illegible]	1	0	0	0	0	0	0	0	-2

Thread #1

Thread #2

Thread #3

• • •

Thread #*n*

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
[illegible]	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	[illegible]	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	[illegible]	[illegible]	[illegible]	[illegible]	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
[illegible]	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	[illegible]	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
[illegible]	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	[illegible]	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	[illegible]	[illegible]	[illegible]	[illegible]	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
[illegible]	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	[illegible]	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0	0	0	0	-2

Thread #1

Thread #2

Thread #3

• • •

Thread #*n*

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	[illegible]	[illegible]	[illegible]	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	1.1	0	0	0	0	1	0	0	0	0	0	2
1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	[illegible]	[illegible]	[illegible]	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	[illegible]	[illegible]	[illegible]	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	[illegible]	0	1	0	0	0	0	0	0	3
-1	1	1.1	0	0	0	0	1	0	0	0	0	0	2
1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	[illegible]	[illegible]	[illegible]	0	0	0	0	0	0	-2

Thread #1

Thread #2

Thread #3

• • •

Thread #*n*

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	[illegible]	0	0	0	0	1	0	0	0	0	0	2
1	-1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	[illegible]	0	0	0	0	1	0	0	0	0	0	2
1	-1	[illegible]	[illegible]	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0	0	-2

Thread #1

Thread #2

Thread #3

• • •

Thread #*n*

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

B) Parallel reduction steps

- ➢ Find minimum ratio (pivot row)

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

B) Parallel reduction steps

- ➢ Find minimum ratio (pivot row)

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	[illegible]	-1	[illegible]	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	[illegible]	[illegible]	[illegible]	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	[illegible]	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	[illegible]	[illegible]	[illegible]	[illegible]	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

Block #2
Threads #1-*h*

Block #1
Threads #1-*h*

GPU Simplex

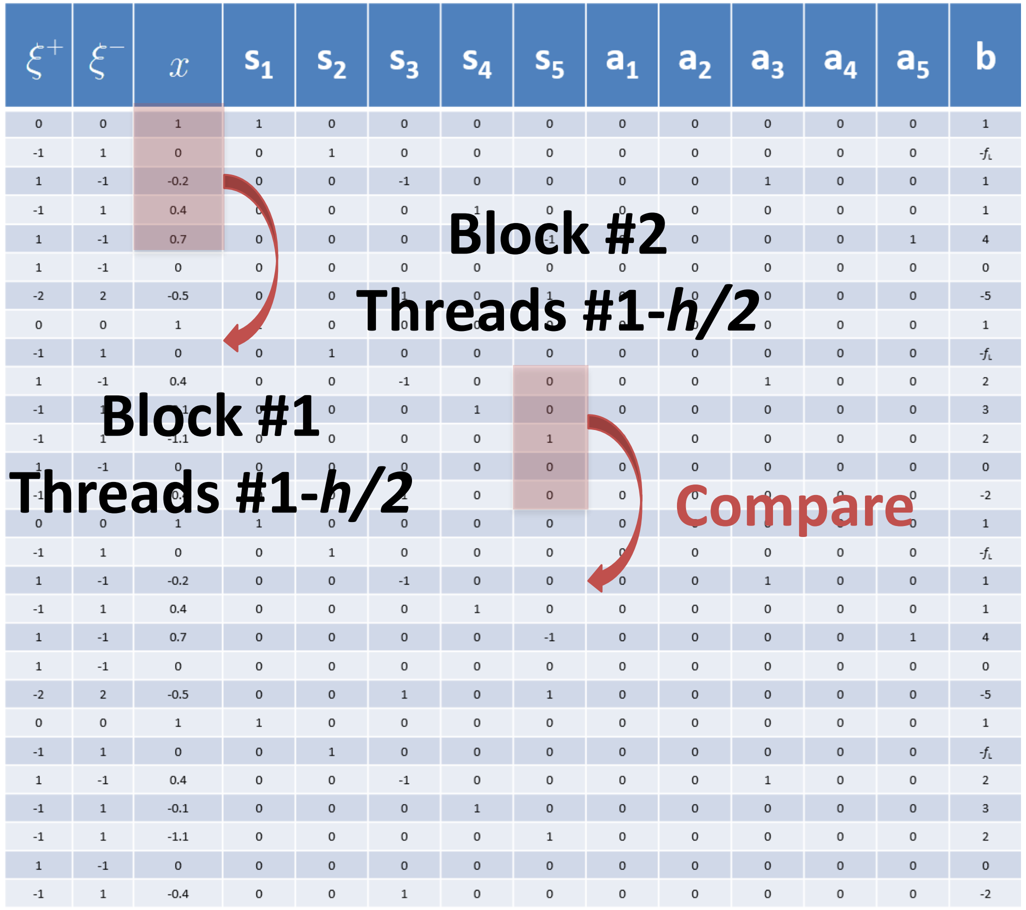
<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

B) Parallel reduction steps

- ➢ Find minimum ratio (pivot row)



GPU Simplex

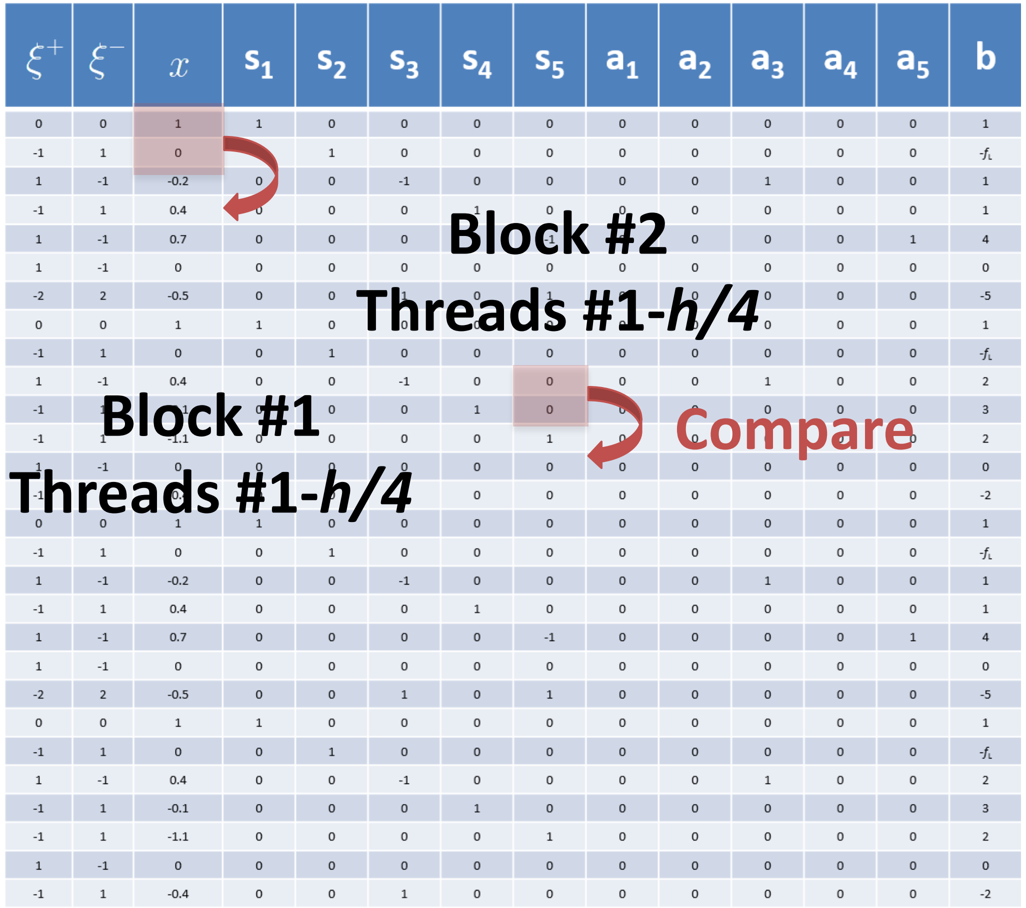
<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

B) Parallel reduction steps

- ➢ Find minimum ratio (pivot row)



GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)

- ➢ Access column information
- ➢ Find pivot column

B) Parallel reduction steps

- ➢ Find minimum ratio (pivot row)

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	[illegible]	0	0	0	0	0	0	0	0	0	1
-1	1	0	[illegible]	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	[illegible]	-1	[illegible]	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	[illegible]	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	[illegible]	0	1	0	0	2
-1	1	[illegible]	[illegible]	0	0	1	0	[illegible]	0	[illegible]	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	[illegible]	[illegible]	[illegible]	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

Block #2
Thread #1

Block #1
Thread #1

Compare

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)
- ➢ Access column information
- ➢ Find pivot column

B) Parallel reduction steps
- ➢ Find minimum ratio (pivot row)

C) "Batch" steps (apply to each LP)
- ➢ Pivoting

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

GPU Simplex

<u>Parallelization approach depends on step:</u>

A) "Vectorized" steps (apply to each row)
  - ➢ Access column information
  - ➢ Find pivot column

B) Parallel reduction steps
  - ➢ Find minimum ratio (pivot row)

C) "Batch" steps (apply to each LP)
  - ➢ Pivoting

ξ^+	ξ^-	x	s_1	s_2	s_3	s_4	s_5	a_1	a_2	a_3	a_4	a_5	b
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	0	1	0	0	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	[illegible]	-1	0	[illegible]	0	0	1	0	0	2
[illegible]	[illegible]	[illegible]	0	[illegible]	[illegible]	1	[illegible]	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	[illegible]	-0.4	0	0	1	0	0	0	0	0	0	0	-2
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	[illegible]	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	-0.2	0	0	-1	0	0	0	0	1	0	0	1
-1	1	0.4	0	0	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0	0	1
1	-1	0.7	0	0	0	0	-1	0	0	0	0	1	4
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-2	2	-0.5	0	0	1	0	1	0	0	0	0	0	-5
0	0	1	1	0	0	0	0	0	0	0	0	0	1
-1	1	0	0	1	0	0	0	0	0	0	0	0	$-f_L$
1	-1	0.4	0	0	-1	0	0	0	0	1	0	0	2
-1	1	-0.1	0	0	0	1	0	0	0	0	0	0	3
-1	1	-1.1	0	0	0	0	1	0	0	0	0	0	2
1	-1	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	-0.4	0	0	1	0	0	0	0	0	0	0	-2

Block #3, Thread #2

Block #3

3 Key Parallelization Targets

Relaxations

LP Generation

LP Solves

Vapor Pressure Parameter Estimation

$$\mathbf{p}^* \in \arg\min_{\mathbf{p}\in P\subset\mathbb{R}^{n_p}} f(\mathbf{p}) = \sum_{i=1}^{N}\left[\frac{\pi^{calc}(\mathbf{x}_i,\mathbf{p}) - \pi_i^{\exp}}{\pi_i^{\exp}}\right]^2$$

$$\mathbf{p} = (a_0, a_1, a_2, b_0, b_1, b_2)$$

$$\mathbf{x}_i = (w_i, T_i)$$

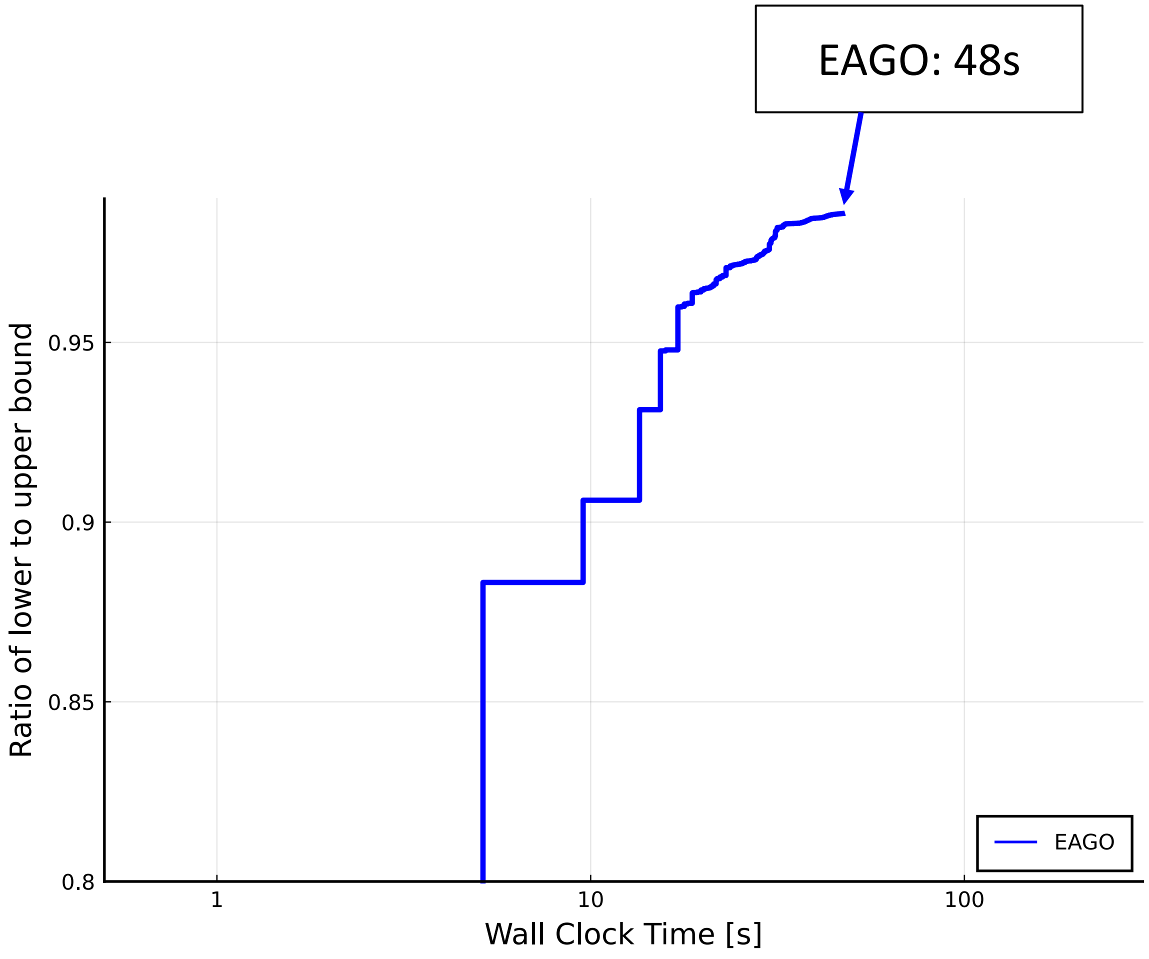
$$\log(\pi^{calc}) = \sum_{i=0}^{2} a_i w^i + \frac{\sum_{i=0}^{2} b_i w^i}{T}$$

	Davidson and Erickson's working pair[2]
	$LiNO_3 + KNO_3 + NaNO_3$ (53:28:19)
a_0/kPa	8.7369
a_1/kPa	27.0375
a_2/kPa	−21.4172
b_0/kPa·K	−2432.1378
b_1/kPa·K	−6955.3785
b_2/kPa·K	4525.9568

Davidson and Erickson[2] working pair			
$LiNO_3 + KNO_3 + NaNO_3$ (53:28:19)			
p/kPa	w	p/kPa	w
T = 333.15 K		T = 353.15 K	
12.38	0.4992	28.33	0.4994
10.40	0.5997	24.17	0.6000
6.31	0.6997	18.00	0.7000
		12.38	0.7493
T = 373.15 K		T = 393.15 K	
60.13	0.4997	118.57	0.5004
48.49	0.6006	92.90	0.6015
36.46	0.7003	70.48	0.7008
26.54	0.7495	53.49	0.7500
20.32	0.8000	40.23	0.8004
13.61	0.8503	27.84	0.8506
T = 413.15 K		T = 433.15 K	
215.95	0.5014	377.13	0.5030
179.25	0.6033	292.92	0.6054
126.59	0.7017	220.38	0.7031
100.39	0.7508	175.32	0.7520
78.16	0.8011	140.91	0.8022
52.82	0.8511	93.25	0.8519
33.67	0.8998	59.32	0.9003
		38.36	0.9501
T = 453.15 K		T = 473.15 K	
596.57	0.5050	920.02	0.5079
496.87	0.6092	789.60	0.6145
361.58	0.7051	564.39	0.7078
288.22	0.7537	451.19	0.7560
229.18	0.8036	357.82	0.8056
159.01	0.8531	249.55	0.8547
101.20	0.9011	162.51	0.9022
63.43	0.9506	99.24	0.9512

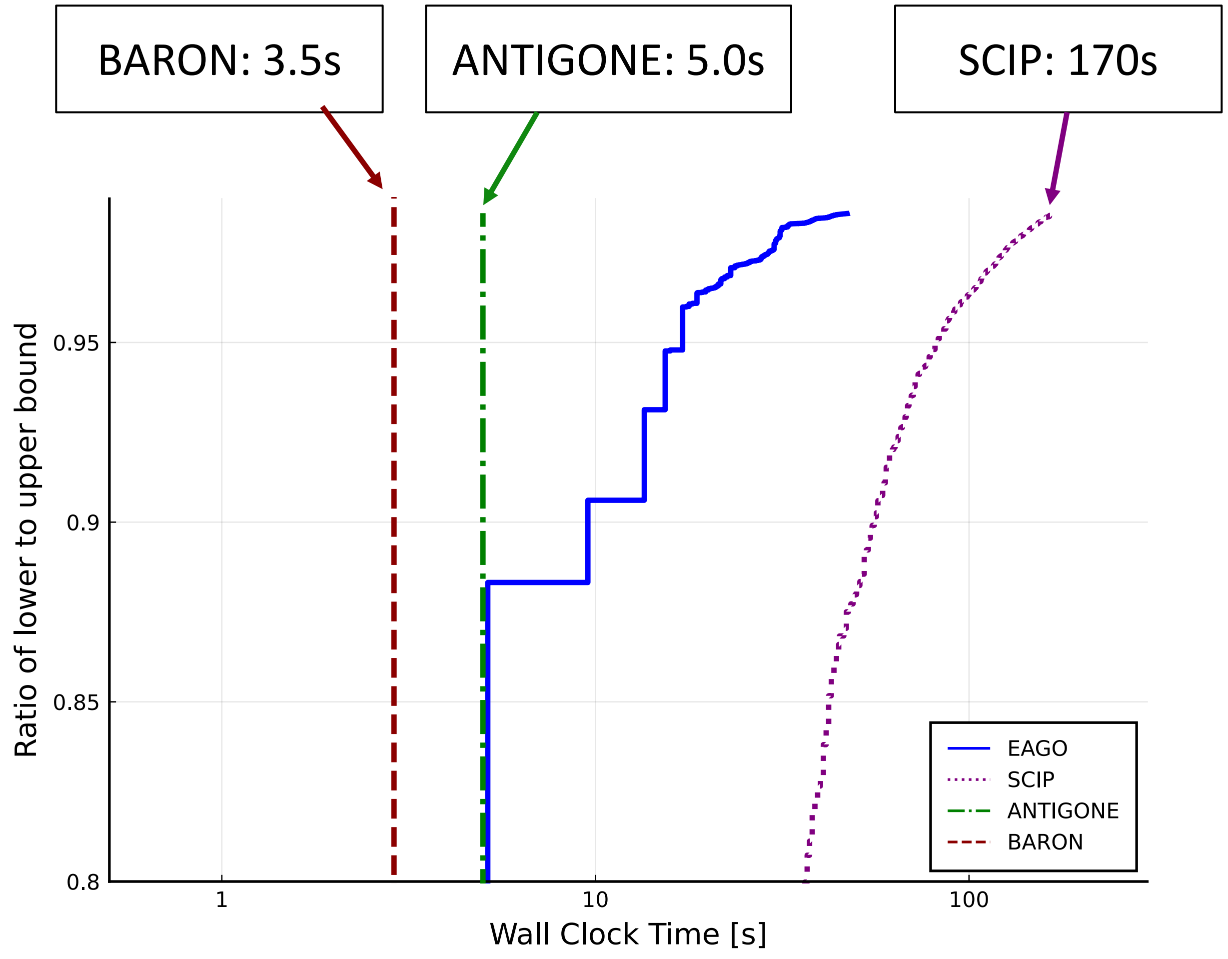
8. Álvarez, M.E., et al., **Vapor-liquid equilibrium of aqueous alkaline nitrate and nitrite solutions for absorption refrigeration cycles with high-temperature driving heat**, *Journal of Chemical & Engineering Data* 56 (2011), pp. 491–496.

Convergence Plot



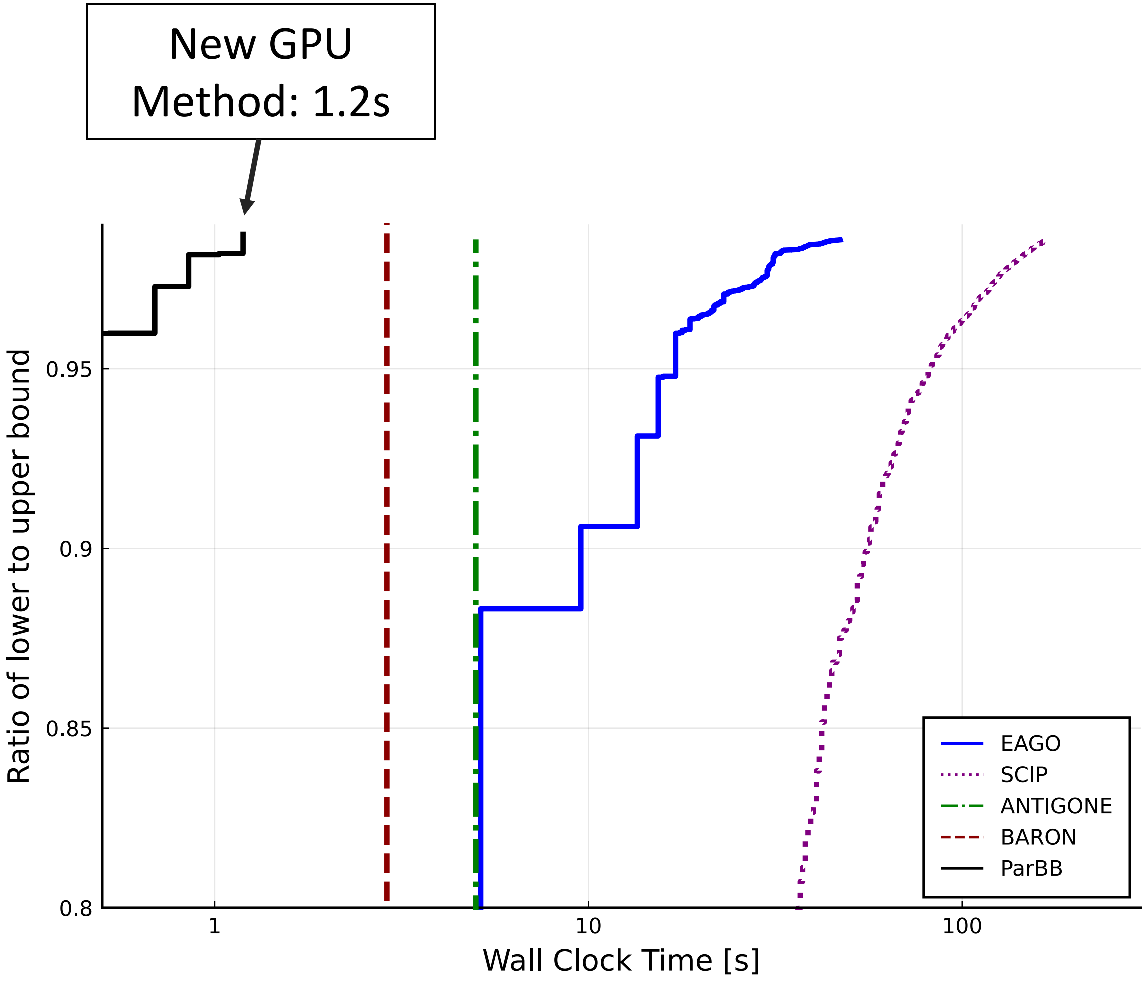
*All examples run on an Intel Xeon W-2195 2.30/4.0 GHz (base/turbo) processor, with an NVIDIA Quadro GV100 GPU

Convergence Plot



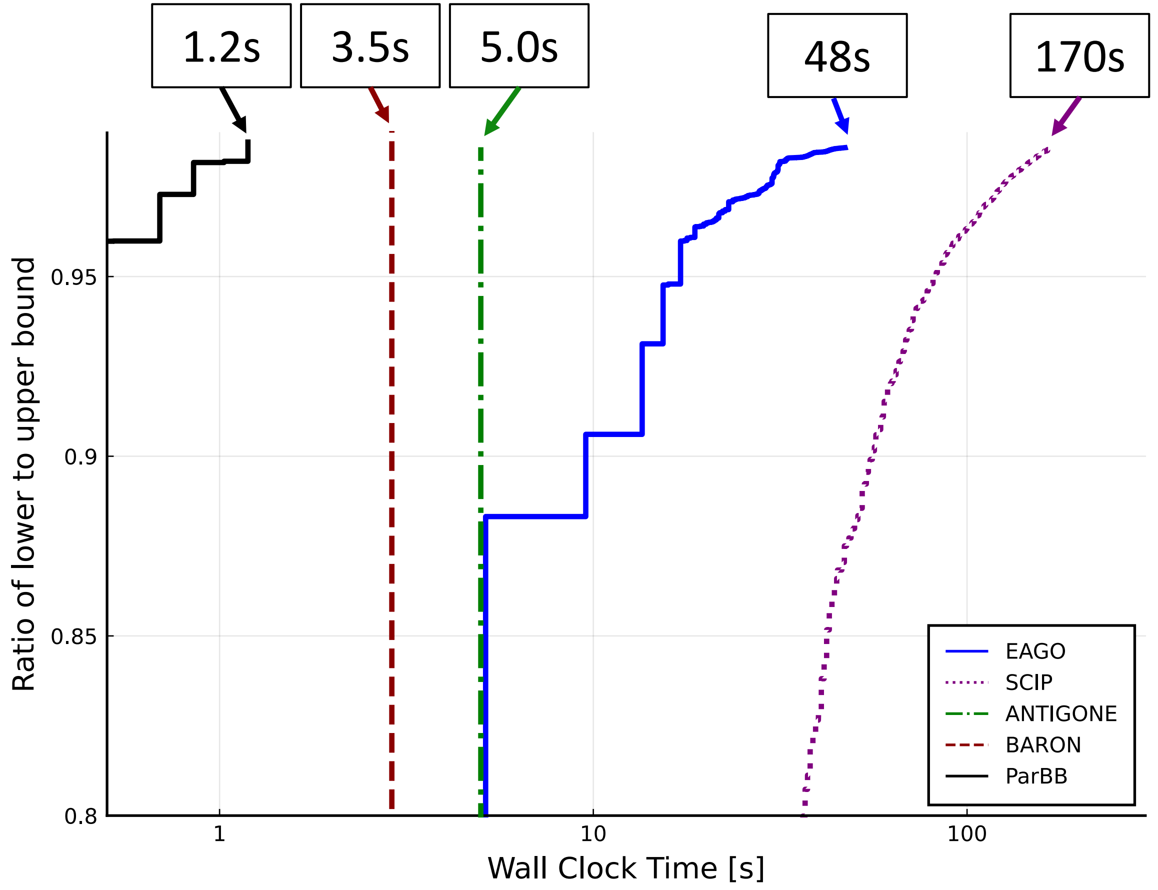
*All examples run on an Intel Xeon W-2195 2.30/4.0 GHz (base/turbo) processor, with an NVIDIA Quadro GV100 GPU

Convergence Plot



*All examples run on an Intel Xeon W-2195 2.30/4.0 GHz (base/turbo) processor, with an NVIDIA Quadro GV100 GPU

Understanding Comparisons



BARON/ANTIGONE

- Problem solved during preprocessing

SCIP

- Evaluated 28161 B&B nodes

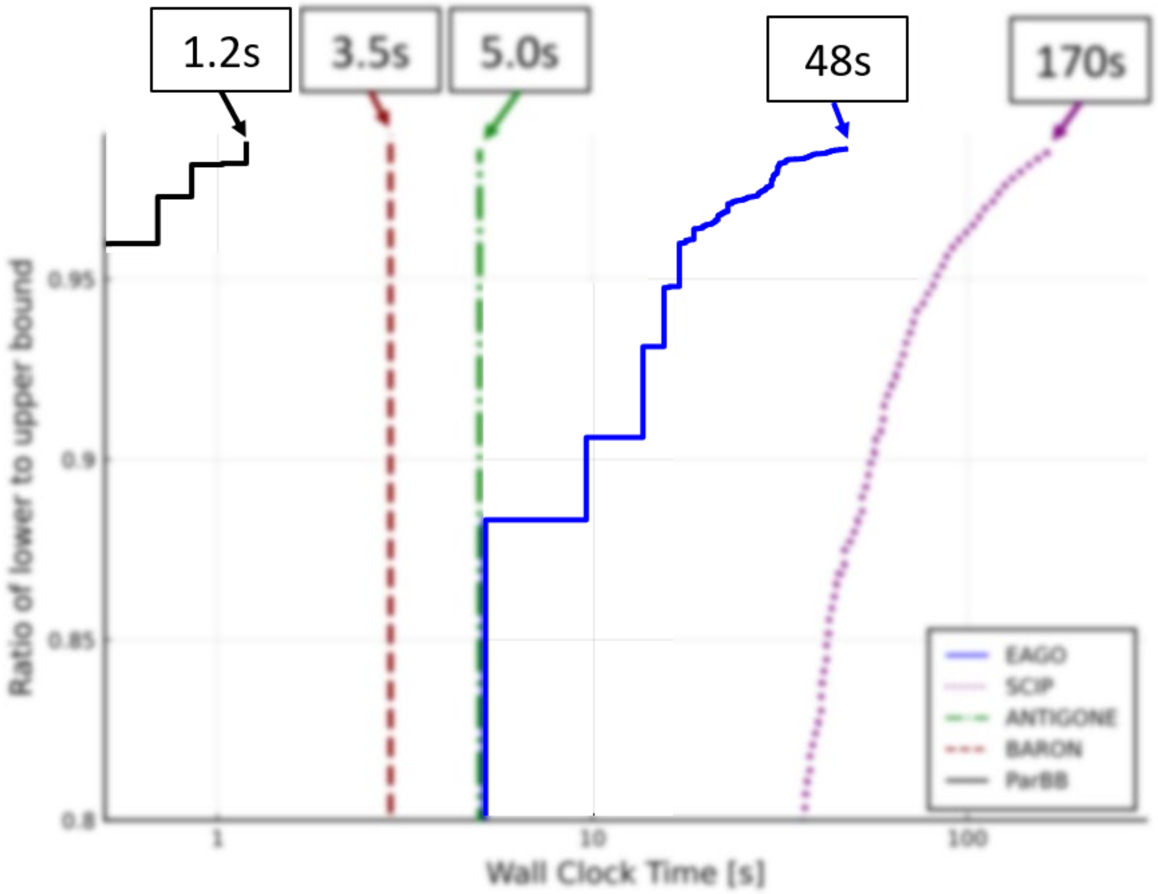
EAGO

- Evaluated 23005 B&B nodes

ParBB

- Evaluated 26542 B&B nodes

Understanding Comparisons



<u>BARON/ANTIGONE</u>

- Problem solved <u>during preprocessing</u>

<u>SCIP</u>

- Evaluated 28161 B&B nodes

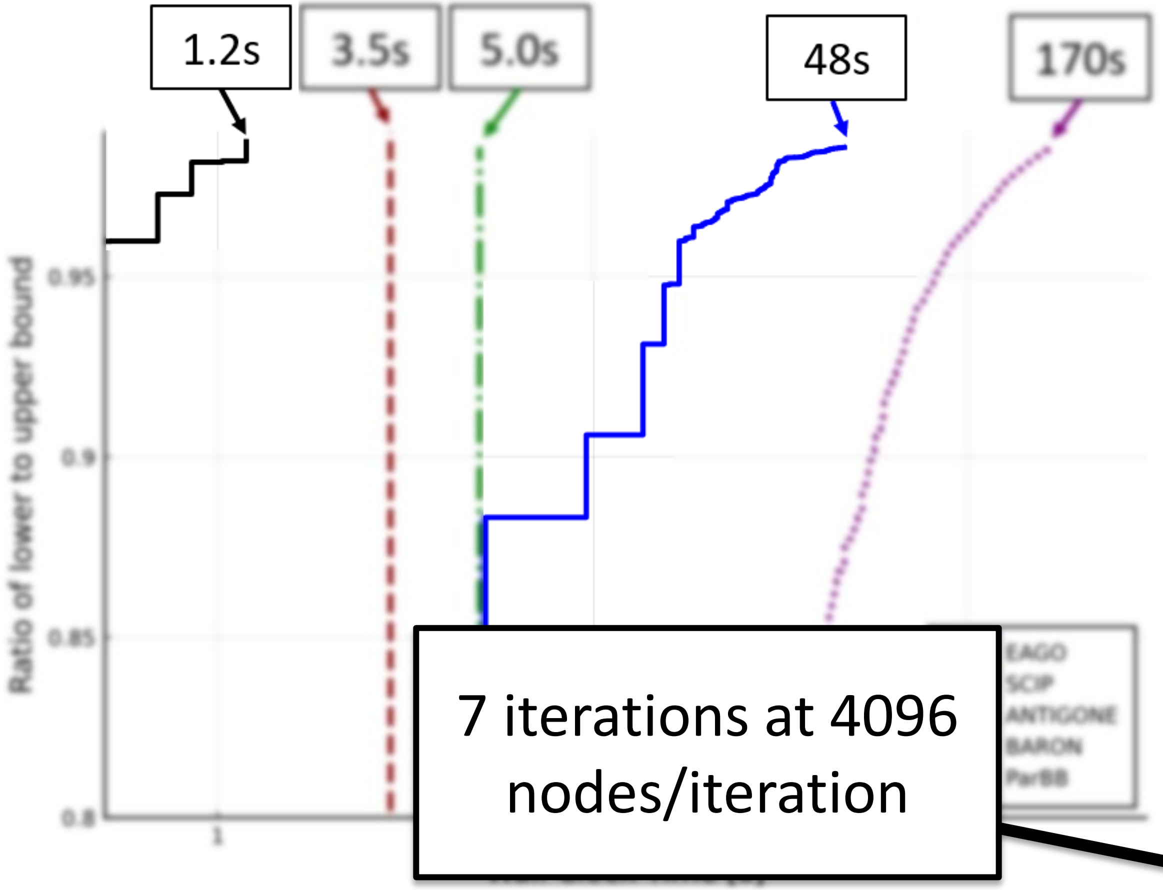
<u>EAGO</u>

- **Evaluated 23005 B&B nodes**

<u>ParBB</u>

- **Evaluated 26542 B&B nodes**

Understanding Comparisons



BARON/ANTIGONE

- Problem solved during preprocessing

SCIP

- Evaluated 28161 B&B nodes

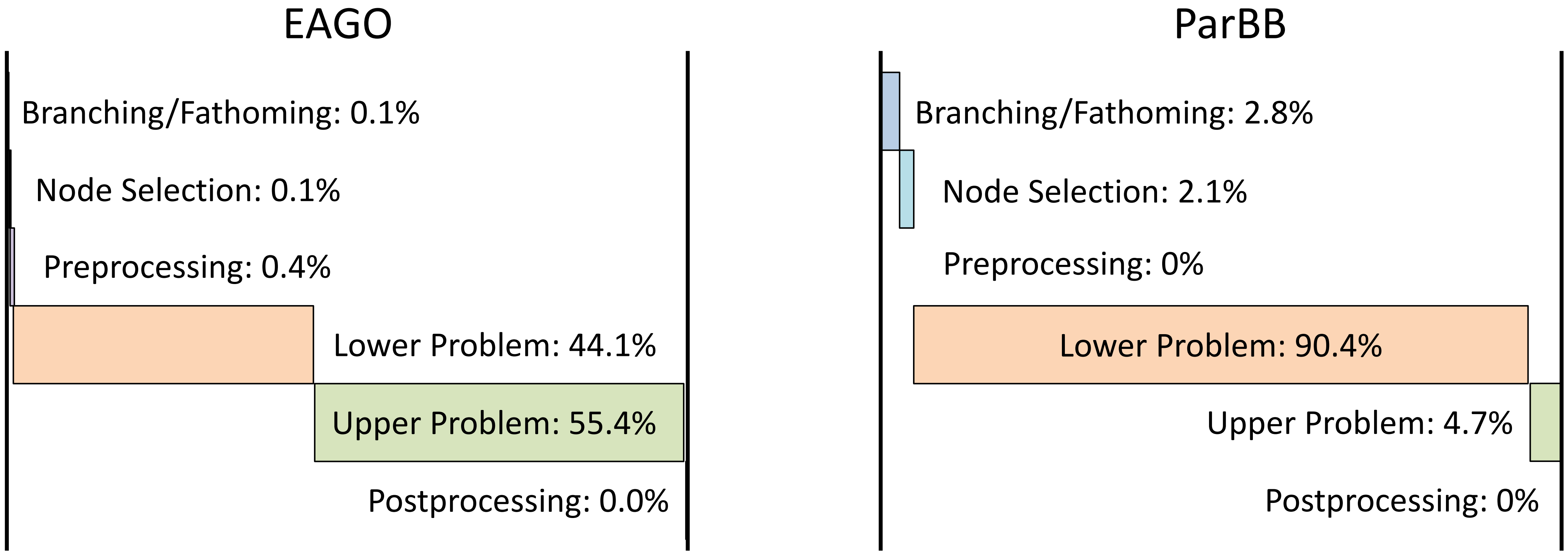
EAGO

- **Evaluated 23005 B&B nodes**

ParBB

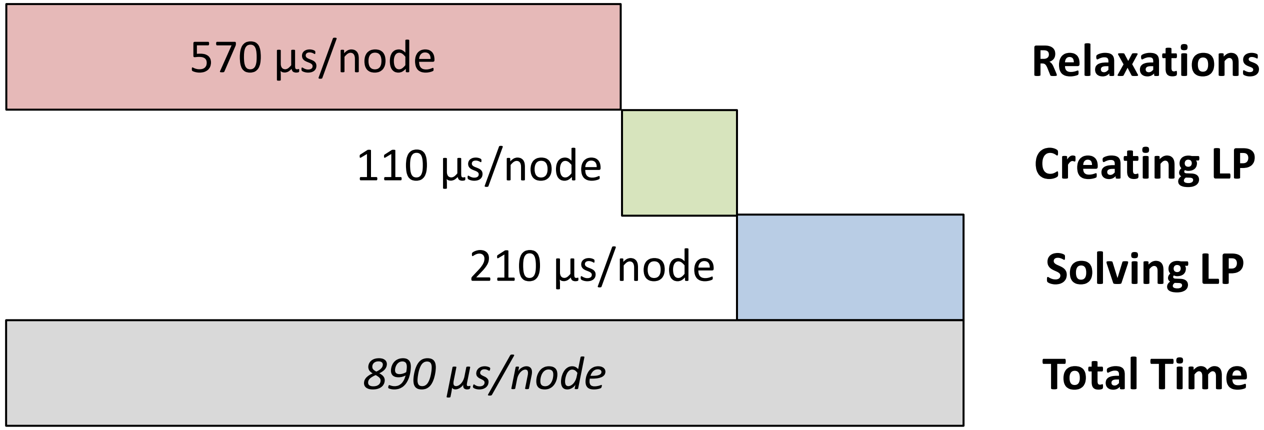
- **Evaluated 26542 B&B nodes**

EAGO vs. ParBB Comparison

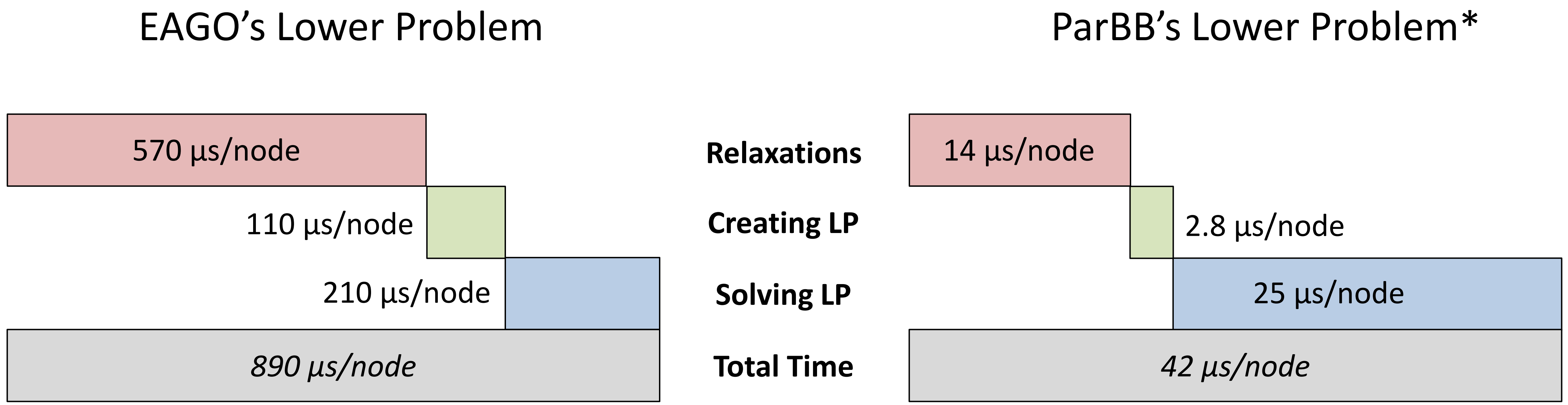


EAGO vs. ParBB Comparison

EAGO's Lower Problem

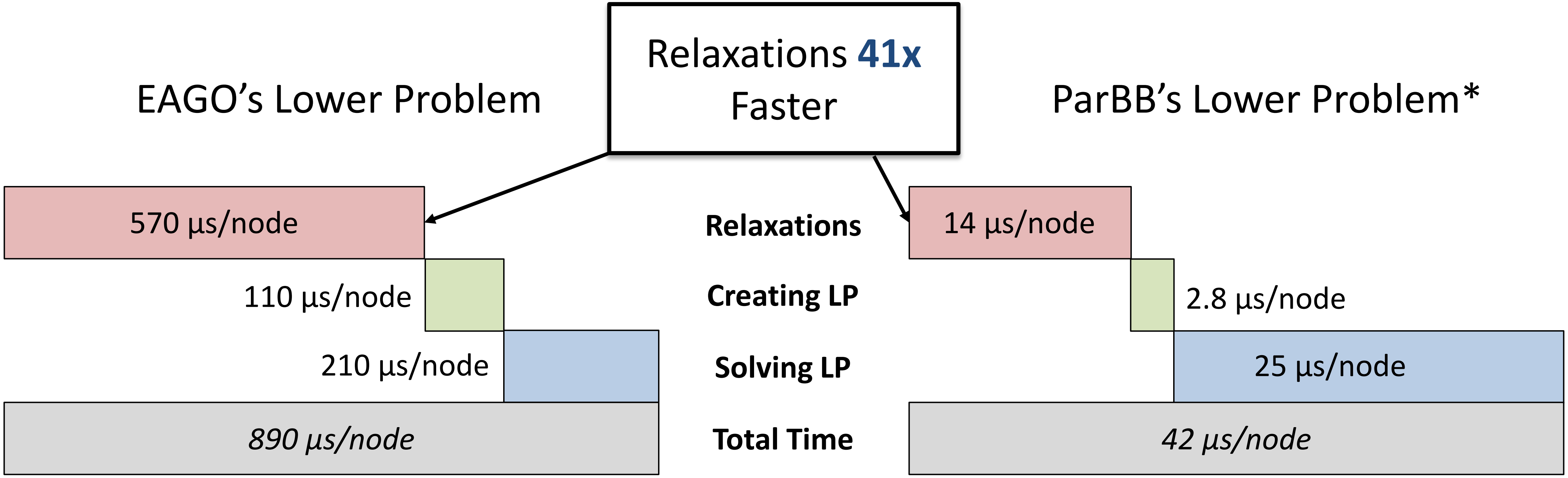


EAGO vs. ParBB Comparison



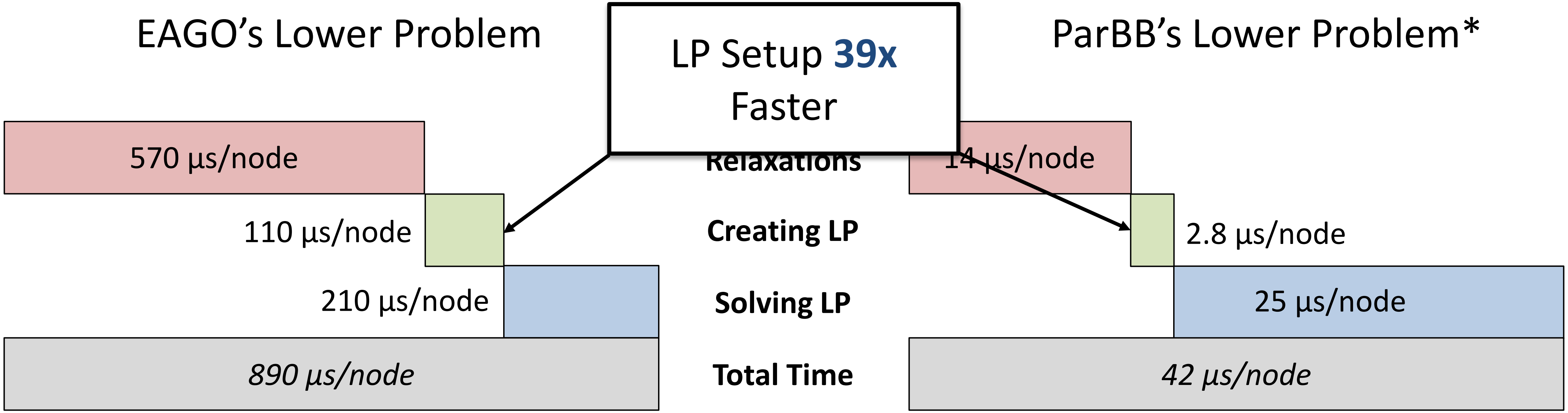
*Time averaged over *n* nodes

EAGO vs. ParBB Comparison



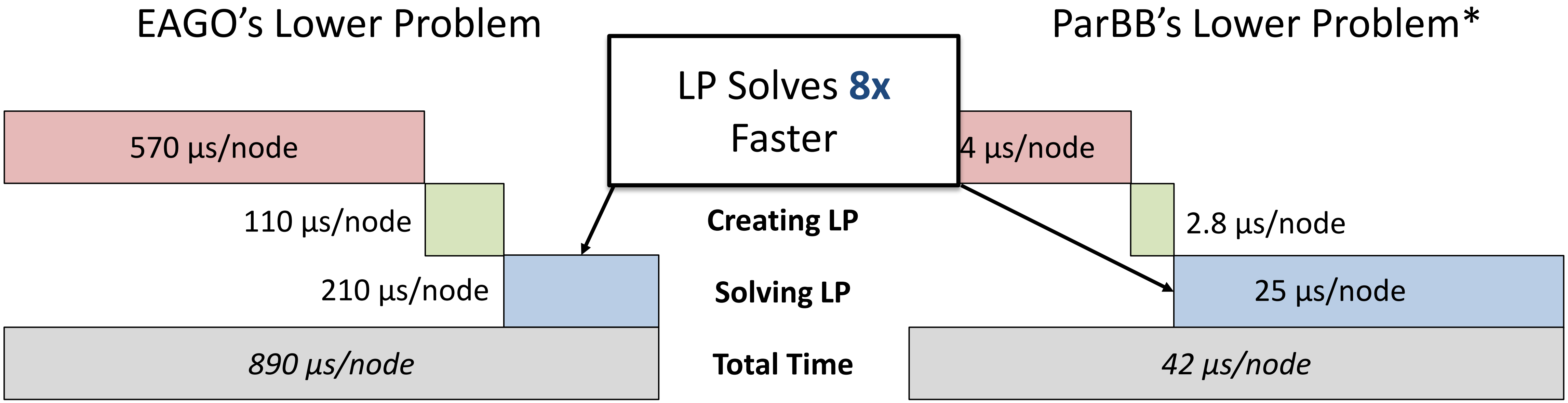
*Time averaged over *n* nodes

EAGO vs. ParBB Comparison



*Time averaged over *n* nodes

EAGO vs. ParBB Comparison

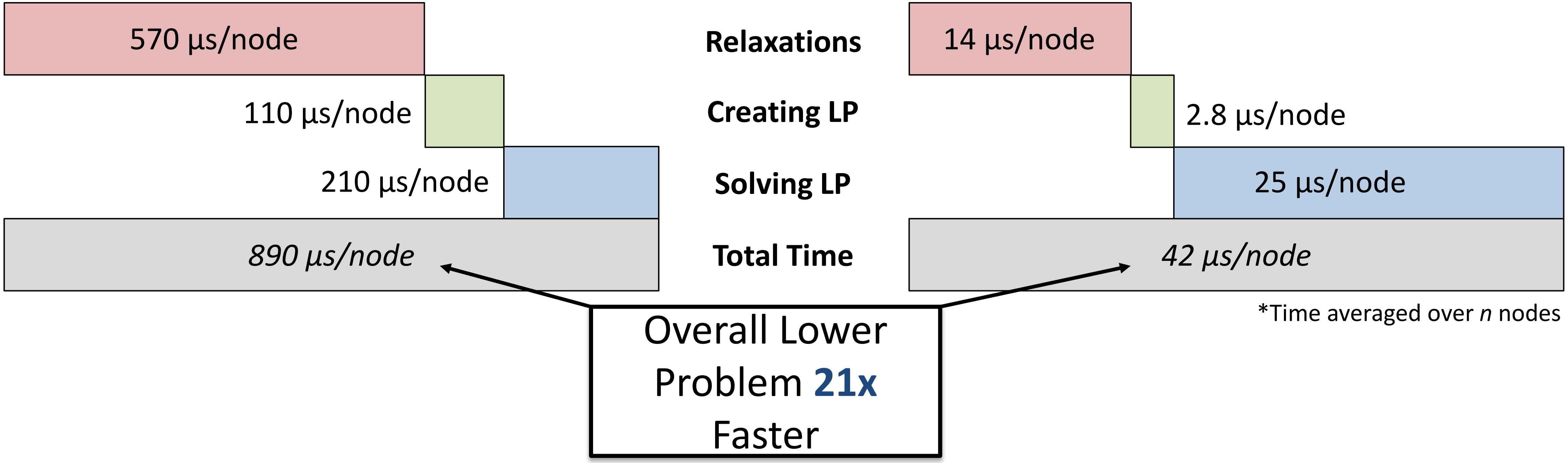


*Time averaged over *n* nodes

EAGO vs. ParBB Comparison

EAGO's Lower Problem

ParBB's Lower Problem*



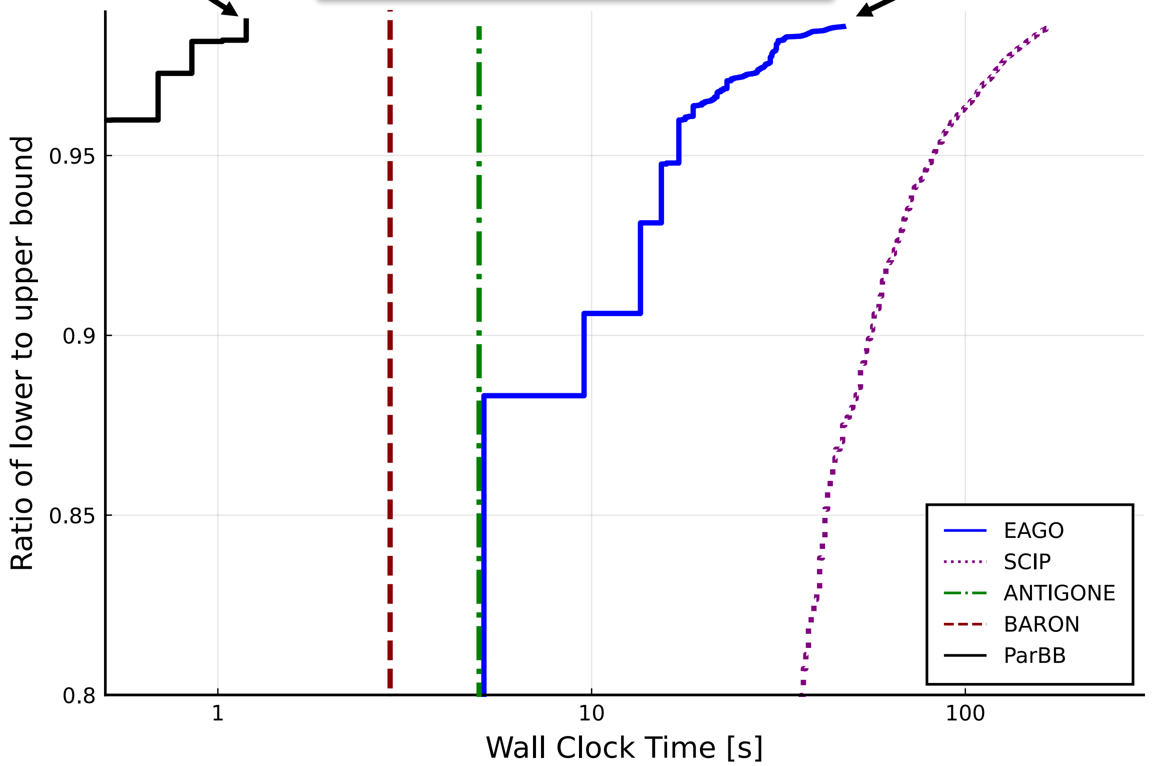
*Time averaged over *n* nodes

Convergence Plot

New GPU Method: 1.2s

Solution **40x** Faster

EAGO: 48s



Future Steps

Many avenues of future research

GPU Relaxations

- Automatic subexpression detection and replacement
- Algorithmic CUDA kernel generation

GPU Simplex

- Hot/warm starting after adding cuts
- Adding cycling detection to use faster heuristic than Bland's rule

GPU B&B

- Support for nontrivial constraints*
- MINLP handling
- Parallelized preprocessing (OBBT, FBBT)
- (Long term) Integration with EAGO/JuMP

*Almost there! Coded up, but more testing needed

Acknowledgements

Members of the Process Systems and Operations Research Laboratory at the University of Connecticut (https://psor.uconn.edu/)

Process Systems and Operations Research Laboratory

UCONN
UNIVERSITY OF CONNECTICUT

{ ISMP 2024 }

Funding:

National Science Foundation, Award No.: **1932723**

National Science Foundation, Award No.: **2330054**

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 or the United States Government.

Questions?

Process Systems and Operations Research Laboratory

https://www.psor.uconn.edu

EAGO

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