GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
Breakthrough Energy
MODAL FORSCHUNGSCAMPUS
FORSCHUNGS CAMPUS
öffentlich-private Partnerschaft für Innovationen

A parallel MIP solver for HiGHS!

M. Turner (joint work with F. Wesselmann)

1st Jun. 2026

HiGHS Workshop 2026

ZIB
ZUSE INSTITUTE BERLIN

Serial HiGHS General Design

Algorithm 1: High-Level HiGHS Framework (v1.14)

Presolve

Process Root Node

while **there exists an open node** **do**
    Select best open node
    Solve node LP relaxation
    Add a round of cuts
    Run heuristics
    Dive
end

Goal: Parallelise the while loop, i.e., parallel branch-and-cut.

Limitation: The solve is largely serial until branching starts (future WIP)

Dream: 2.5x performance improvement from parallelisation

Parallel HiGHS General Design

Algorithm 2: High-Level HiGHS Framework (v1.15)

Function `ProcessNode()`**:**

- Solve node LP relaxation
- Add a round of cuts
- Run heuristics
- Dive

Presolve

Process Root Node

while there exists an open node do

- **if** `workers < openNodes && !maxWorkers` **then**
  - `CreateWorkers()`
- **end**
- Assign best open nodes to workers
- Run `ProcessNode()` in parallel
- Sync

end

What is a Worker?

A worker can independently separate, run heuristics, and dive, while caching newly derived globally valid information.

Contains copies of:

- LP + Search structure (large)
- Global Domain + Cut-pool + Conflict-pool + Pseudo-cost (medium)
- Sepa structure + Best primal solution (small)

Large design decisions:

- Only copy data structures once. Afterwards exclusively repair local changes and sync global changes.
- Create more workers than threads.

Pros vs Cons of Design

Pros:

- Easy to understand
- Only a single sync point
- Work-stealing paradigm with more workers than threads averages out non-uniform worker load

Cons:

- No nodes being processed during sync
- All threads might still have to wait on one long dive
- Uses substantially more memory than serial solve

Gripes + Complaints + Rant

Frustrating hurdles:

- Global references and pointers were accessed and updated from many places
- Debugging non-determinism is difficult
- Minor changes to the main solve loop affect many instances
- Syncing information is easy to get incorrect but still “valid”
- Some atomics were unavoidable

An Example Log

```
Solving MIP model with:
   6594 rows
   8232 cols (7432 binary, 0 integer, 800 implied int., 0 continuous, 0 domain fixed)
   37863 nonzeros

Src: B => Branching; C => Central rounding; F => Feasibility pump; H => Heuristic;
     I => Shifting; J => Feasibility jump; L => Sub-MIP; P => Empty MIP; R => Randomized rounding;
     S => Solve LP; T => Evaluate node; U => Unbounded; X => User solution; Y => HiGHS solution;
     Z => ZI Round; l => Trivial lower; p => Trivial point; u => Trivial upper; z => Trivial zero

        Nodes      |    B&B Tree     |            Objective Bounds              | Dynamic Constraints |       Work
Src  Proc. InQueue |  Leaves   Expl. | BestBound       BestSol              Gap |  Cuts   InLp Confl. | LpIters     Time

 J       0       0         0   0.00%   -inf            396140             Large        0      0      0         0     0.9s
 R       0       0         0   0.00%   318             81373            99.61%         0      0      0     15254     5.3s
         0       0         0   0.00%   318             81373            99.61%        24      6      0     23697    10.5s
 C       0       0         0   0.00%   318             45460            99.30%        38     10      0     25868    12.4s
         0       0         0   0.00%   318             45460            99.30%       115     27      0     32769    18.0s
         0       0         0   0.00%   318             45460            99.30%       183     48      0     39438    23.2s
         0       0         0   0.00%   318             45460            99.30%       266     59      0     46521    28.7s
 L       0       0         0   0.00%   318             329               3.34%       402     68      0     50961    42.5s

Symmetry detection completed in 0.0s
Found 3 generator(s) and 11 full orbitope(s) acting on 3466 columns

         7       8         7   0.00%   318             329               3.34%       409     53      0     78040    48.8s
        31      32        31   0.00%   318             329               3.34%       436     60      0    151831    58.2s
 T    3454       0        96 100.00%   318             318               0.00%       436     60      0     7179k   574.1s
      3454       0        96 100.00%   318             318               0.00%       540     60    353     7179k   574.2s

Solving report
  Model             neos-933966
  Status            Optimal
  Primal bound      318
  Dual bound        318
```

Highs MIP Changes in the Last Year

Technique	% Improve. Range	% Affected	Overhead
Degree 1 test	$[0, 1]$	< 5	$\times$
Dual subst. + bound strength.	$[2, 3]$	< 50	✓
CG strength.	$[0, 1]$	< 50	$\times$
Predictive bound analysis	$[1, 2]$	< 30	$\times$
Singleton column stuffing	$[1, 2]$	< 10	$\times$
Solution enumeration	$[2, 3]$	< 50	✓
Extract additional VBs	$[1, 2]$	< 20	$\times$
MIR strength. for VBs	$[0, 1]$	< 10	$\times$
Automatically apply implications	$[1, 3]$	< 50	$\times$
Tuning + code improvements	$[1, 4]$	< 90	$\times$
Big fixes	$[-3, 0]$	< 15	$\times$
Parallel search	$[?, ?]$	< 85	✓

Future HiGHS MILP Goals

Short-term:

- Bug fixes (please always makes issues if you encounter a problem)
- Single-machine scheduling separator
- Find infeasible assignments and strongly connected components via the clique table
- Solve disconnected instances separately

Medium-term:

- Add Local-MIP type heuristic
- Add strongCG + flow cover cuts
- The year of presolve!

Long-term:

- Add sequence dependent lifted binary + GUB cover cuts
- Introduce unified framework for indicator / SOS constraints
- Lazy constraints
- Unit commitment specific techniques