ISCAS 中国科学院软件研究所
Institute of Software Chinese Academy of Sciences

约束求解研究室
Research Laboratory of
Constrained Solving

中国科学院大学
University of Chinese Academy of Sciences

Parallel MIP Solving with Dynamic Task Decomposition

Peng Lin, Shaowei Cai *, Mengchuan Zou, Shengqi Chen

Institute of Software, Chinese Academy of Sciences

2025.08.12

Presenter

* Corresponding Author

Outline

Background

- Mixed Integer Programming
- Parallel MIP Solving

PartiMIP

- Process Flow of The Framework
- Dynamic Task Decomposition
- Acceleration Components

Experiments

- Comparison to Parallel Divide-and-Conquer Strategies
- Comparison to Sequential Solving
- Ablation Study
- New Best Known Solutions to Open Instances

Future Work

Background

Mixed Integer Programming

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to:} \quad & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u} \\ & \boldsymbol{x} \in \mathbb{R}^n,\ x_j \in \mathbb{Z}\ for\ all\ j \in I \end{aligned}$$

Objective Function

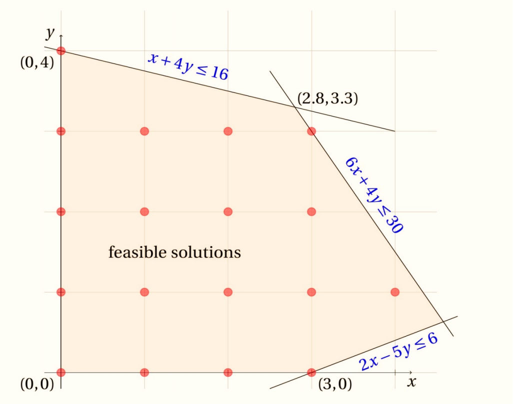
General Linear Constraints

Global Bounds

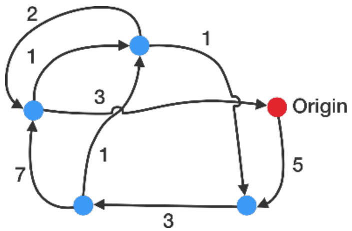
Integrality Constraints

Solving MIP

is NP-Hard

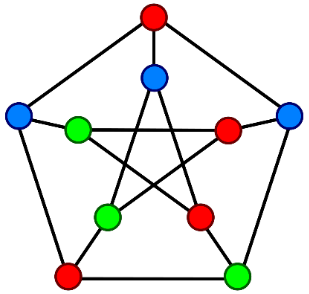


Powerful Expressive Ability



TSP

Bin Packing



Graph Problems

Extensive Practical Applications

Resource Allocation

Crew Scheduling

Production Planning

Parallel MIP Solving

Nearly all SoTA MIP solvers support parallelism.

Commercial Solvers

GUROBI OPTIMIZATION

Gurobi
https://www.gurobi.com/

IBM CPLEX

CPLEX
https://www.ibm.com/products/ilog-cplex-optimization-studio

Academic / Open-source Solvers

SCIP Solving Constraint Integer Programs

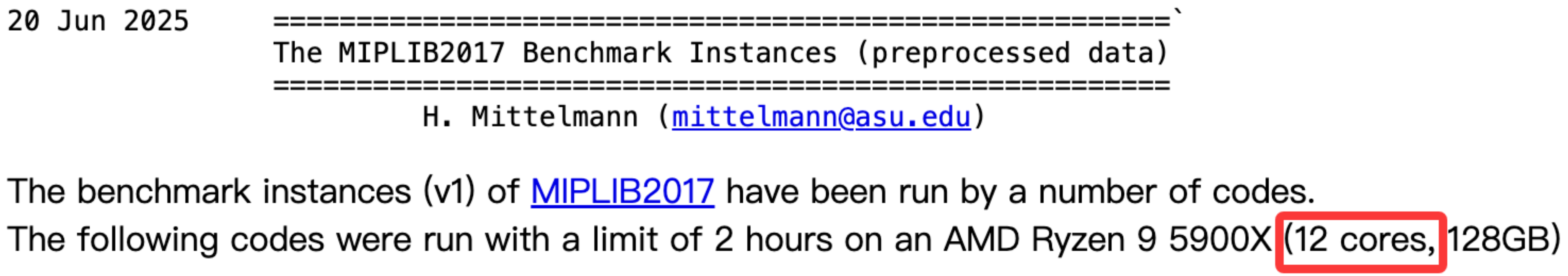
UG Ubiquity Generator framework

SCIP/FiberSCIP
[Achterberg, 2009;
Shinano, IJOC'18]

HiGHS

HiGHS
[Huangfu and Hall, MPC'18]

The widely recognized H. Mittelmann benchmark ranks MIP solvers based on parallel performance.



```
20 Jun 2025      =======================================================`
                 The MIPLIB2017 Benchmark Instances (preprocessed data)
                 =======================================================
                         H. Mittelmann (mittelmann@asu.edu)
```

The benchmark instances (v1) of MIPLIB2017 have been run by a number of codes.
The following codes were run with a limit of 2 hours on an AMD Ryzen 9 5900X (12 cores, 128GB)

Source: https://plato.asu.edu/ftp/milp.html

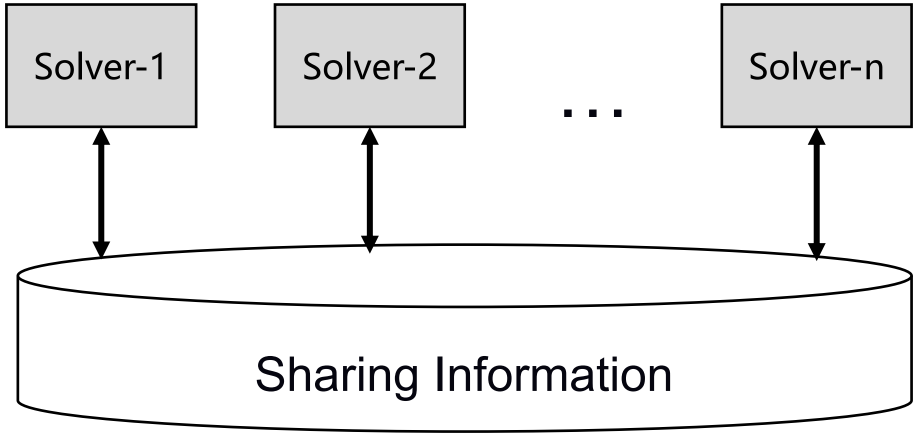
Challenges for Building Parallel MIP Solvers

- **From Scratch**
  - A massive effort is needed to build a general and effective parallel MIP solver.

- **Based on Existing Solvers**
  - Limited access to top solvers
    - The best sequential solvers are commercial and closed-source
    - Academic access is restricted to black-box usage, limiting parallel integration
  - Sequential dependence
    - Node processing order of B&B solvers is crucial for performance.
    - Replicating the order in parallel introduces costly overhead

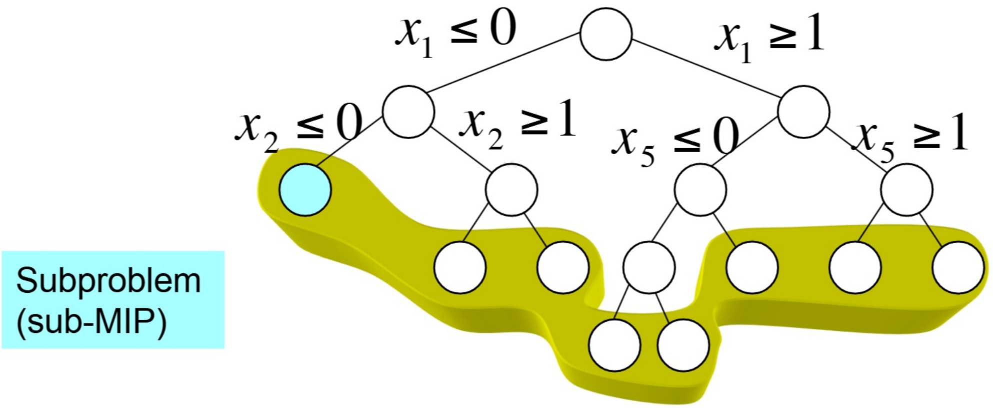
Approaches of Parallel MIP Solving

- **Portfolio Methods**
  - Run multiple complementary solvers/configurations on identical/perturbed instances.
    - Parallel Local search: ParalLP [Lin et al., IJCAI'24]
      - Different initial solutions.
    - Racing ramp-up: FiberSCIP [Shinano et al., IJOC'18]
      - Different parameters, branching rules, etc.
  - Limitation: Performance is inherently constrained by the best sequential execution.



Approaches of Parallel MIP Solving

- **Divide-and-Conquer Methods**
  - Accelerate solving by parallelizing key algorithmic components.
    - Parallel branch-and-bound: FiberSCIP [Shinano et al., IJOC'18]
    - Parallel dual simplex: HiGHS [Huangfu and Hall, MPC'18]
  - Potential: Can outperform the best sequential methods.

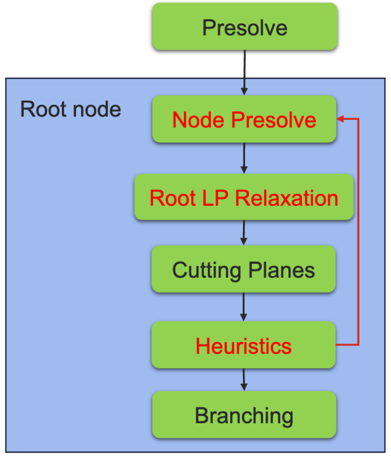


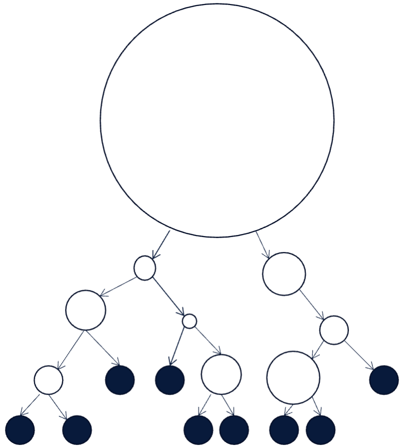
Subproblems of B&B can be processed independently.

Source: https://www.scipopt.org/workshop2014/parascip_libraries.pdf

Challenges in Current Divide-and-Conquer

- **Tightly coupled with underlying sequential solvers.**
  - Heavily affected by the search strategies of sequential solvers.
  - For example, in parallel branch-and-bound
    - Sequential solvers generate parallel processing nodes
    - Determined by the branching and node selection strategies of sequential solvers
- **Parallel B&B can only be parallelized after the root node processing.**
  - Parallel node solving happens after branching.
  - Root node solving is vital but usually requires a significant amount of time.





Source: https://www.gurobi.com/wp-content/uploads/How-to-Exploit-Parallelism-in-Linear-and-Mixed-Integer-Programming.pdf

PartiMIP

Goals of PartiMIP

- **Focus on divide-and-conquer**
  - Potential scalability
  - Easy to integrate with portfolio strategies in the future
- **Flexible parallel strategies**
  - Enable search strategies independent of the base solver's internal logic.
- **Quick parallelization**
  - Enabling parallel solving before the root node processing.
- **Friendly Interface**
  - Base solvers only require standard I/O
  - Not limited to B&B solvers

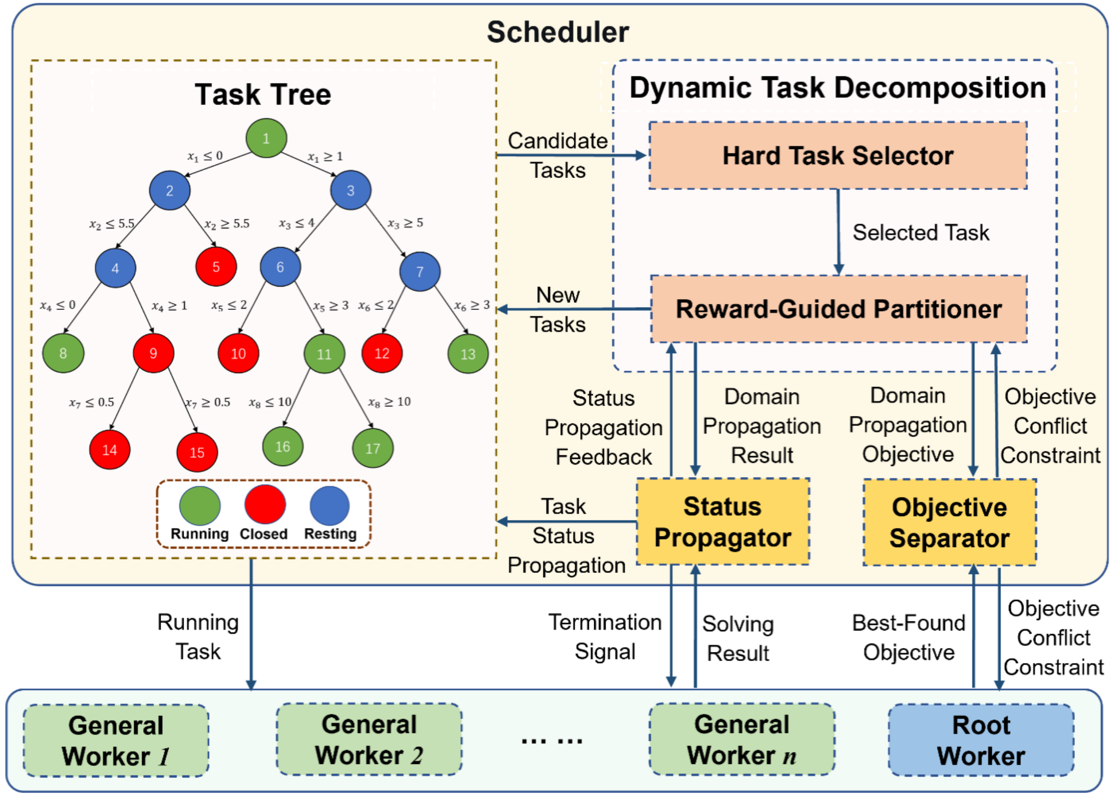
Roles in PartiMIP

Scheduler

- Maintains a dynamic task tree
  - expands the tree via task decomposition
- Deduce task states
  - status propagator
- Prunes search space
  - objective separator

Workers

- Invoke a base MIP solver on assigned tasks
- Loosely coupled
  - interact via standard I/O interfaces

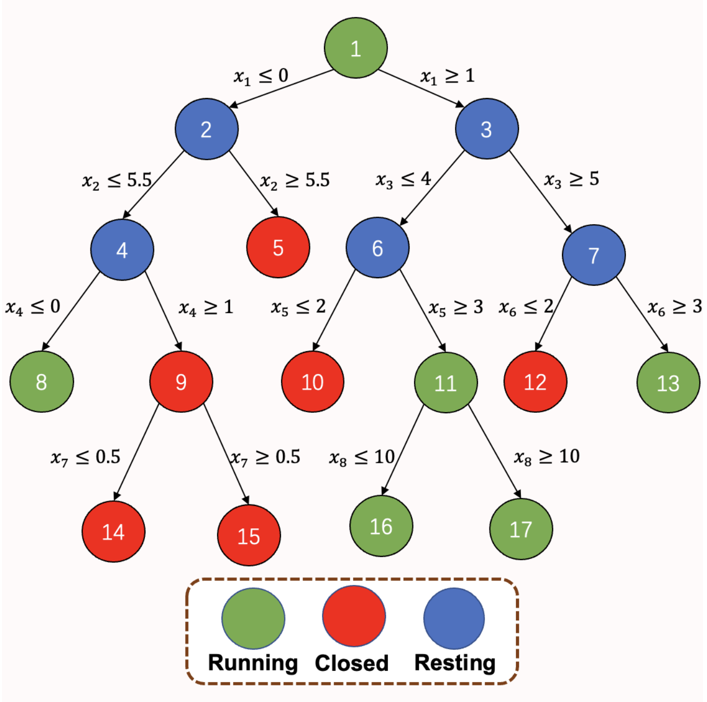


Task Tree

- Nodes are the solving tasks for the original problem or subproblems of a given MIP instance.

Task Status

- **Running**: currently being solved by workers
- **Closed**: finished (either optimal or infeasible)
  - The entire solve ends when root task is closed
- **Resting**: decomposed but unassigned
  - Results are inferred from subtasks



Leaf Task: each has a distinct search space

Process Flow of the Framework

Root First

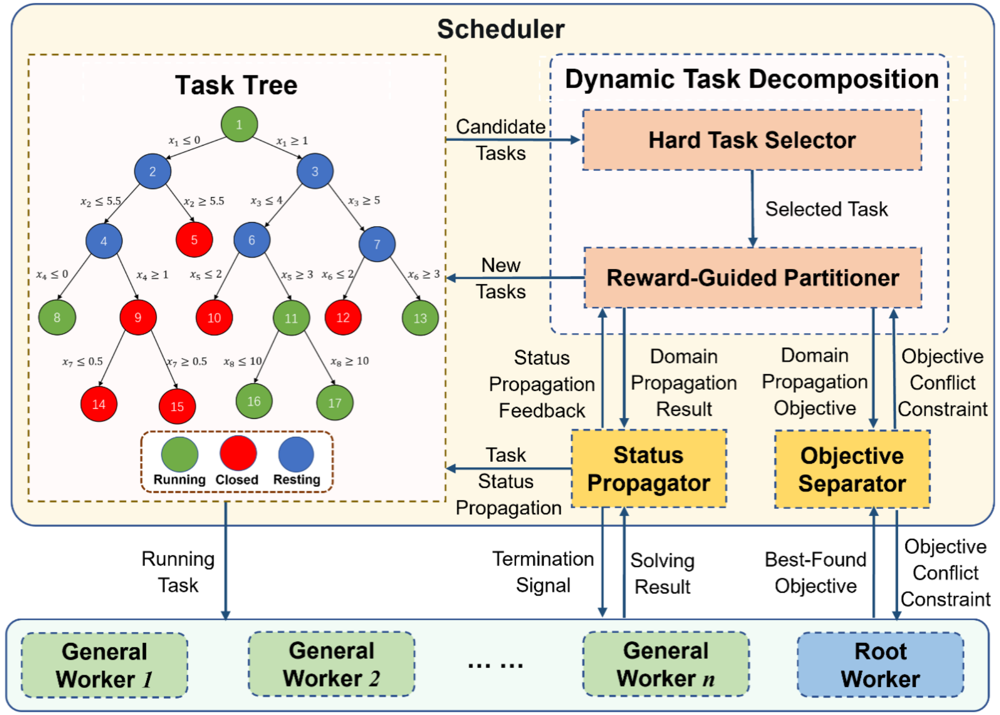
- Scheduler assigns root task to root worker
- Enables fast termination for easy instances

Initial Phase

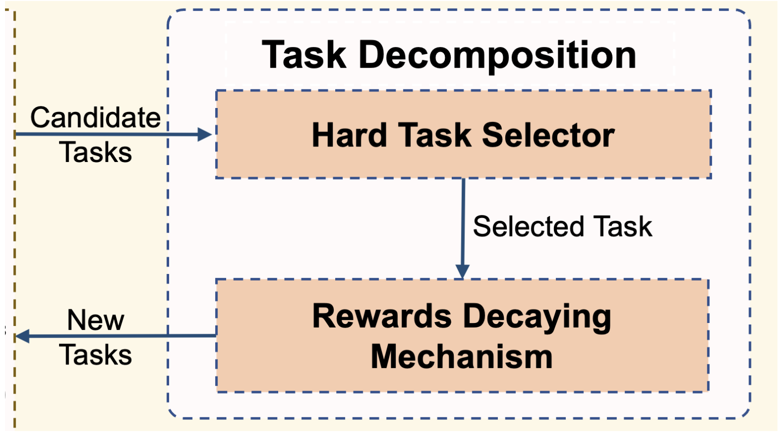
- Scheduler parallelly decomposes leaf tasks
- Continues until there are enough leaf tasks
- All general workers solve distinct spaces

Dynamic Phase

- When a worker finishes its task, it becomes idle
- Scheduler dynamically decomposes running tasks
- Newly created subtasks are assigned to idle workers



Task Decomposition



1. **Leaf Task Selection**
   Select a current leaf node from the task tree.
2. **Variable Choice**
   Choose a branching variable for the selected task.
3. **Domain Split**
   Divide the chosen variable’s domain into two subranges.
4. **Subtask Creation**
   Generate two new subtasks corresponding to subranges.
5. **Domain Propagation**
   Apply propagation to tighten each subtask’s search space.
6. **Tree Update**
   Insert the new subtasks as leaf nodes in the task tree.

Hard Task Selector

- Decompose challenging tasks first to guide resources to bottlenecks.
- Measure "Hardness"
  - Initial Decomposition

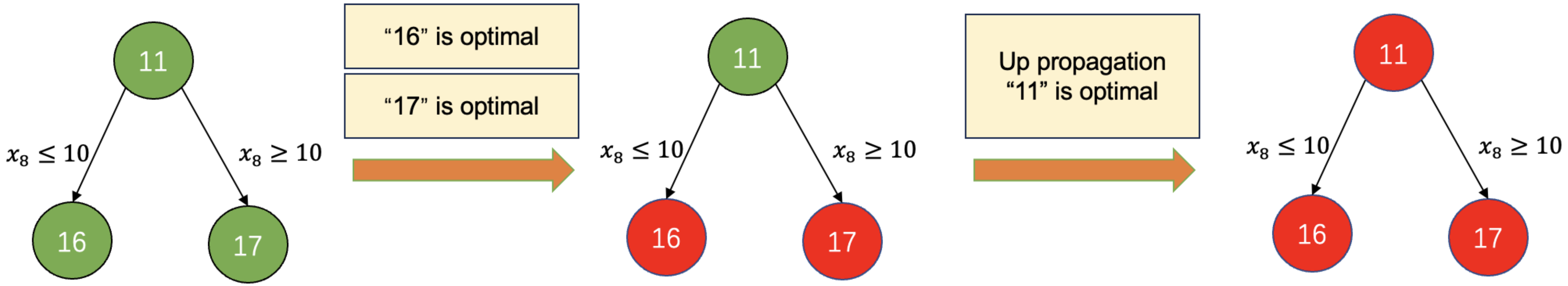
    Hardness is estimated by the number of non-zero elements (nnz) in the task‘s constraints.
  - Dynamic Decomposition

    Hardness is $nnz \times duration$ (how long it has been running).

$$\text{Hardness}(\mathcal{T}) = \begin{cases} nnz \text{ of } \mathcal{T}, & \text{initial decomposition phase,} \\ nnz \text{ of } \mathcal{T} \times duration \text{ of } \ \mathcal{T}, & \text{dynamic decomposition phase.} \end{cases}$$

Reward Decomposition-effective Variable

- Reinforce effective variable selections for future decompositions
- Decomposition-effective variable
  - one that leads to faster resolution of subtasks than the original task.



- Rewarding Rule:
  - $\text{Reward}(x_T) < \text{Reward}(x_T) + 1$, if task T is closed via upward propagation
    - x_T is variable used to decompose task T

Reward-Guided Variable Selection

Variable Selection

- Choose the variable with the highest reward and break the tie by constraint degree

Risk in reward-guided selection

- Positive feedback loop
    - High reward → more likely to be selected → reward increases again
- Premature convergence; other good variables are ignored

Decaying Strategy

- Think of the reward as a "global quota" consumed with each use.
- When a variable is selected for decomposition, its reward is reduced by 1

$$\mathrm{Reward}(x_{\mathcal{T}}) := \begin{cases} \mathrm{Reward}(x_{\mathcal{T}}) - 1, & \text{if } \mathrm{Reward}(x_{\mathcal{T}}) > 0 \\ \mathrm{Reward}(x_{\mathcal{T}}), & \text{otherwise} \end{cases}$$

Acceleration Components

Task Status Propagation

- Subtasks' search spaces together exactly equal their parent's.
- **Status Signals**
  - Domain propagation
  - Worker results
- **Upward Propagation**
  - All children infeasible → parent infeasible
  - Any child optimal → parent optimal
- **Downward Propagation**
  - When parent closes, children inherit the same status

Objective Conflict Constraint

- Ensure workers only explore solutions better than the current global best–found solution.
- **Mechanism**:
  - Track the real-time best objective value, O^*.
  - For each new task, add the constraint:

$$\text{Objective Conflict Constraint: } \mathbf{c}_{\mathcal{T}}\mathbf{x}_{\mathcal{T}} < \mathcal{O}^* - \text{offset}_{\mathcal{T}}$$

- **Benefit**
  - Prunes search space and guides workers toward improving solutions

Experiments

Experiment Settings

Benchmark & Solvers

- Evaluated on the complete MIPLIB 2017 benchmark (240 instances).
- Integrated with state-of-the-art open-source solvers: SCIP (v9.2.0) and HiGHS (v1.9.0).

Testing Environment

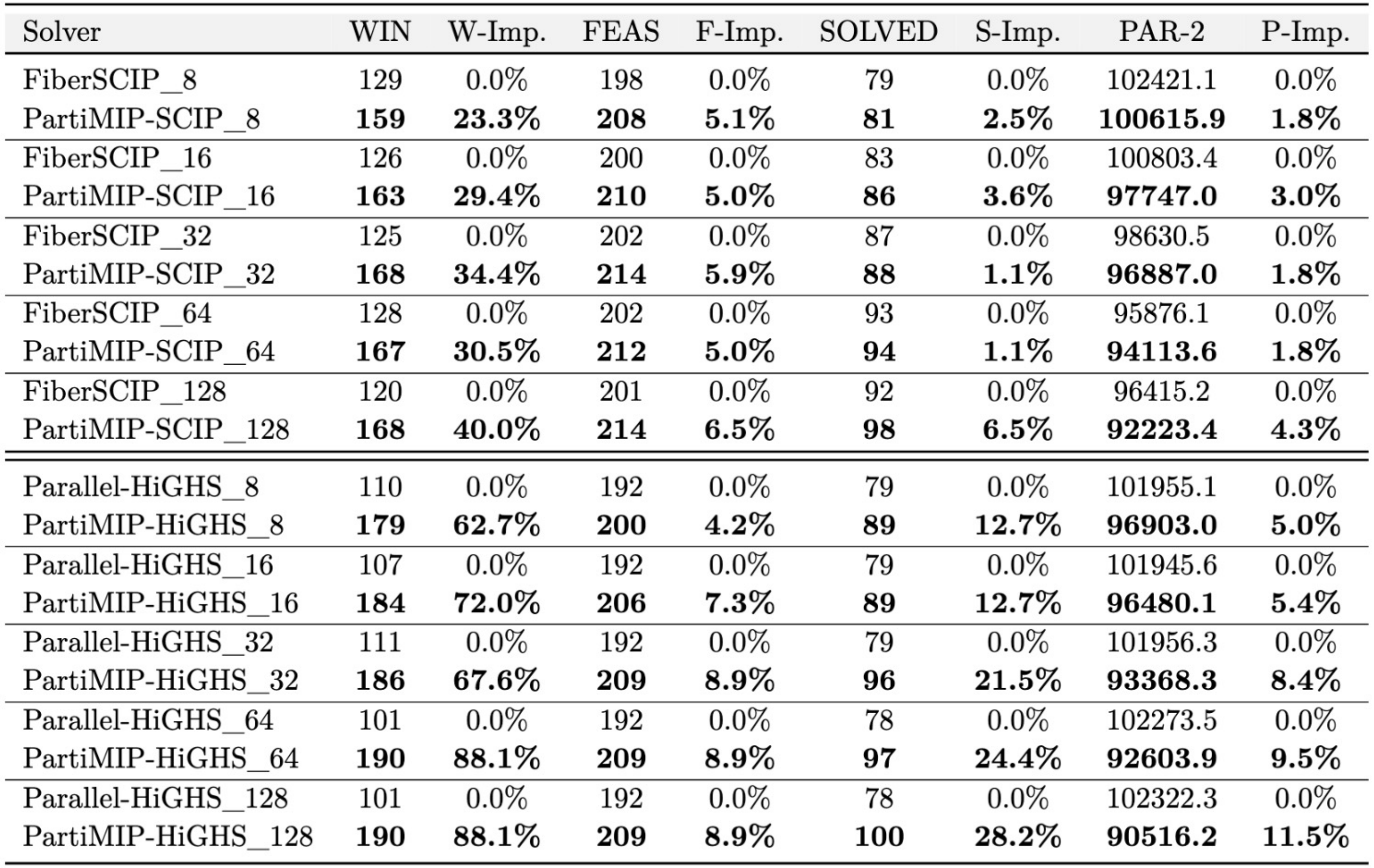
- Scale: Tested on 8, 16, 32, 64, and 128 cores — the largest scale reported for entire MIPLIB.
- Time Limit: 300 seconds per instance, with over 2.3 CPU years of total compute time.

Key Performance Metrics

- Instances Solved (#SOLVED): Total problems solved to optimality / infeasible.
- Efficiency (PAR-2 Score): A combined score of runtime and completion rate.
- Solution Quality (#FEAS / #WIN): Ability to find feasible and best solutions.

Comparison to Parallel D&C Strategies

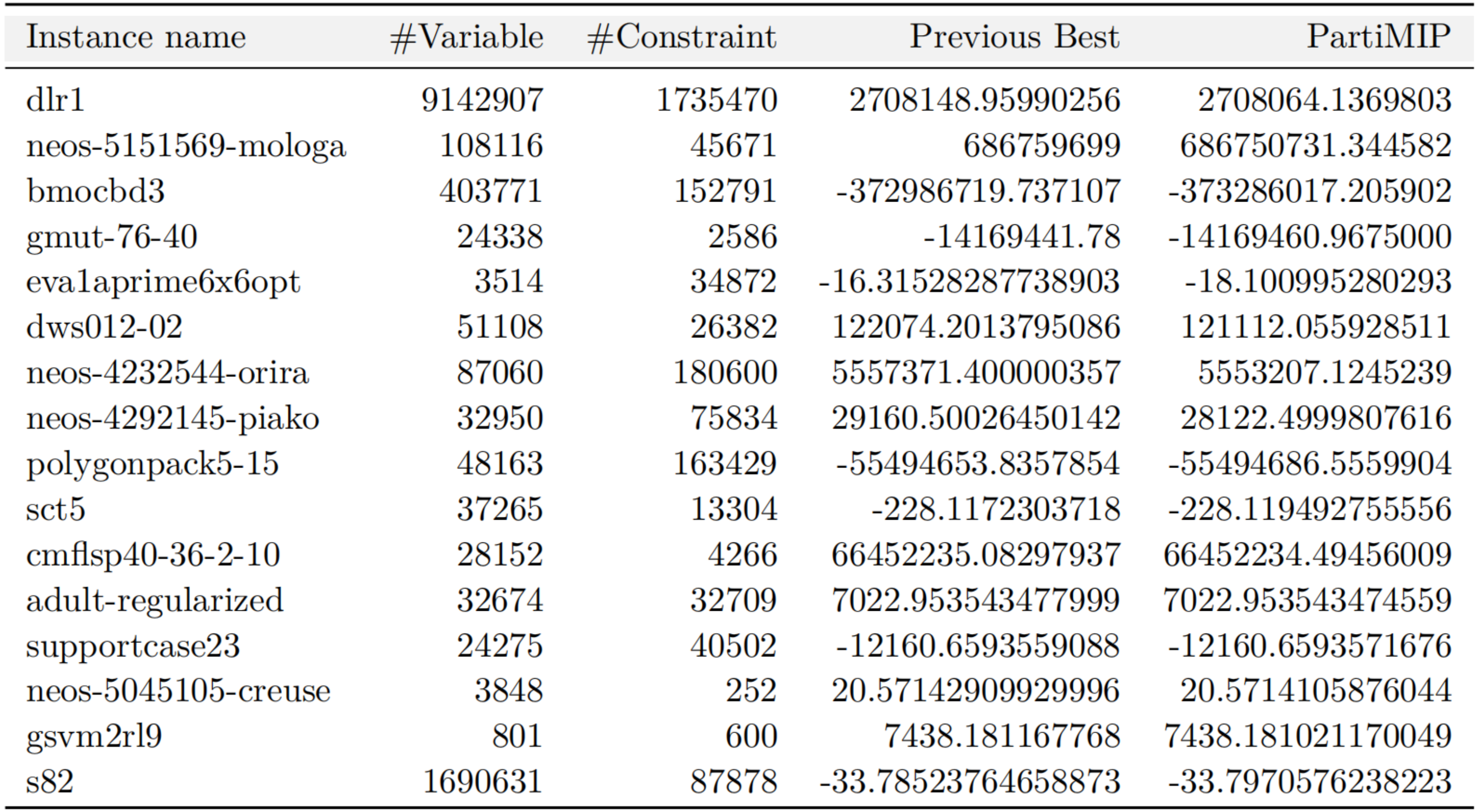
PartiMIP consistently outperforms the default parallel divide-and-conquer approaches of SCIP and HiGHS.



Solver	WIN	W-Imp.	FEAS	F-Imp.	SOLVED	S-Imp.	PAR-2	P-Imp.
FiberSCIP_8	129	0.0%	198	0.0%	79	0.0%	102421.1	0.0%
PartiMIP-SCIP_8	**159**	**23.3%**	**208**	**5.1%**	**81**	**2.5%**	**100615.9**	**1.8%**
FiberSCIP_16	126	0.0%	200	0.0%	83	0.0%	100803.4	0.0%
PartiMIP-SCIP_16	**163**	**29.4%**	**210**	**5.0%**	**86**	**3.6%**	**97747.0**	**3.0%**
FiberSCIP_32	125	0.0%	202	0.0%	87	0.0%	98630.5	0.0%
PartiMIP-SCIP_32	**168**	**34.4%**	**214**	**5.9%**	**88**	**1.1%**	**96887.0**	**1.8%**
FiberSCIP_64	128	0.0%	202	0.0%	93	0.0%	95876.1	0.0%
PartiMIP-SCIP_64	**167**	**30.5%**	**212**	**5.0%**	**94**	**1.1%**	**94113.6**	**1.8%**
FiberSCIP_128	120	0.0%	201	0.0%	92	0.0%	96415.2	0.0%
PartiMIP-SCIP_128	**168**	**40.0%**	**214**	**6.5%**	**98**	**6.5%**	**92223.4**	**4.3%**
Parallel-HiGHS_8	110	0.0%	192	0.0%	79	0.0%	101955.1	0.0%
PartiMIP-HiGHS_8	**179**	**62.7%**	**200**	**4.2%**	**89**	**12.7%**	**96903.0**	**5.0%**
Parallel-HiGHS_16	107	0.0%	192	0.0%	79	0.0%	101945.6	0.0%
PartiMIP-HiGHS_16	**184**	**72.0%**	**206**	**7.3%**	**89**	**12.7%**	**96480.1**	**5.4%**
Parallel-HiGHS_32	111	0.0%	192	0.0%	79	0.0%	101956.3	0.0%
PartiMIP-HiGHS_32	**186**	**67.6%**	**209**	**8.9%**	**96**	**21.5%**	**93368.3**	**8.4%**
Parallel-HiGHS_64	101	0.0%	192	0.0%	78	0.0%	102273.5	0.0%
PartiMIP-HiGHS_64	**190**	**88.1%**	**209**	**8.9%**	**97**	**24.4%**	**92603.9**	**9.5%**
Parallel-HiGHS_128	101	0.0%	192	0.0%	78	0.0%	102322.3	0.0%
PartiMIP-HiGHS_128	**190**	**88.1%**	**209**	**8.9%**	**100**	**28.2%**	**90516.2**	**11.5%**

New Best-Known Solutions

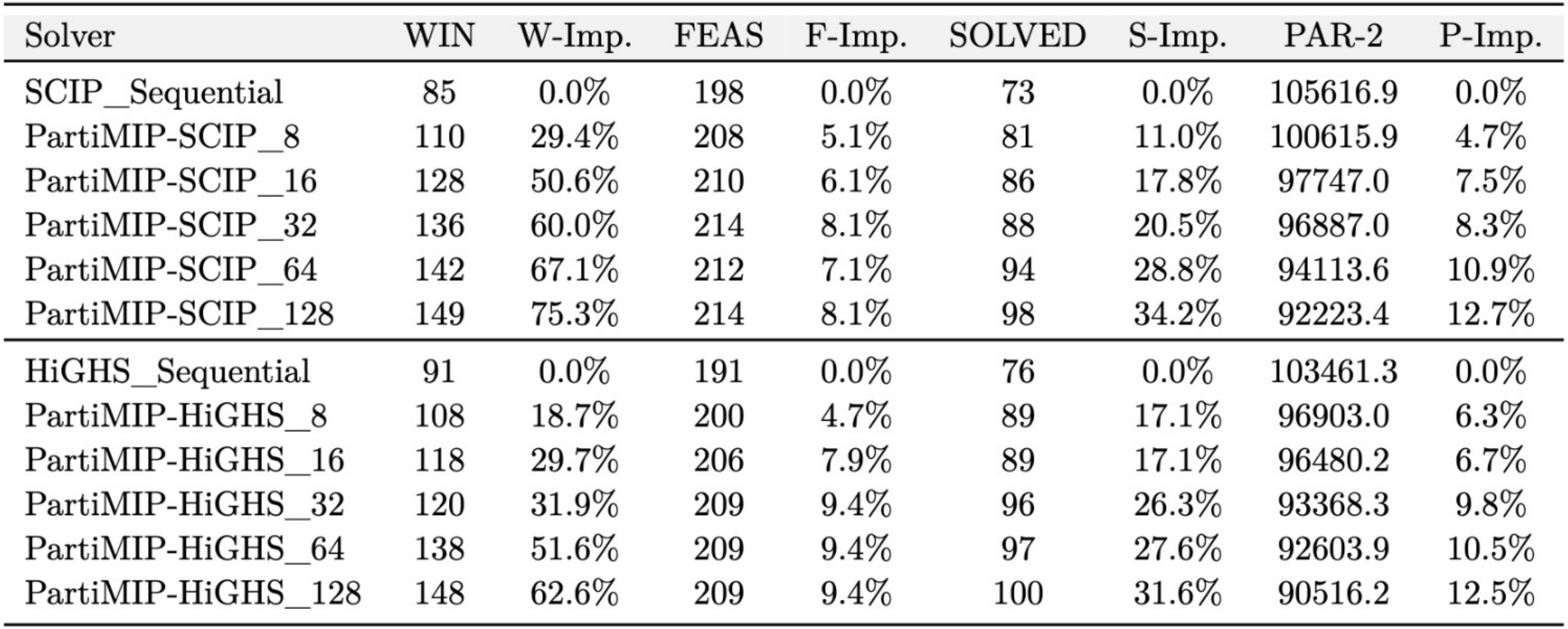
PartiMIP establishes 16 new best-known solutions for MIPLIB open instances.



Instance name	#Variable	#Constraint	Previous Best	PartiMIP
dlr1	9142907	1735470	2708148.95990256	2708064.1369803
neos-5151569-mologa	108116	45671	686759699	686750731.344582
bmocbd3	403771	152791	-372986719.737107	-373286017.205902
gmut-76-40	24338	2586	-14169441.78	-14169460.9675000
eva1aprime6x6opt	3514	34872	-16.31528287738903	-18.100995280293
dws012-02	51108	26382	122074.2013795086	121112.055928511
neos-4232544-orira	87060	180600	5557371.400000357	5553207.1245239
neos-4292145-piako	32950	75834	29160.50026450142	28122.4999807616
polygonpack5-15	48163	163429	-55494653.8357854	-55494686.5559904
sct5	37265	13304	-228.1172303718	-228.119492755556
cmflsp40-36-2-10	28152	4266	66452235.08297937	66452234.49456009
adult-regularized	32674	32709	7022.953543477999	7022.953543474559
supportcase23	24275	40502	-12160.6593559088	-12160.6593571676
neos-5045105-creuse	3848	252	20.57142909929996	20.5714105876044
gsvm2rl9	801	600	7438.181167768	7438.181021170049
s82	1690631	87878	-33.78523764658873	-33.7970576238223

Comparison to Sequential Solving

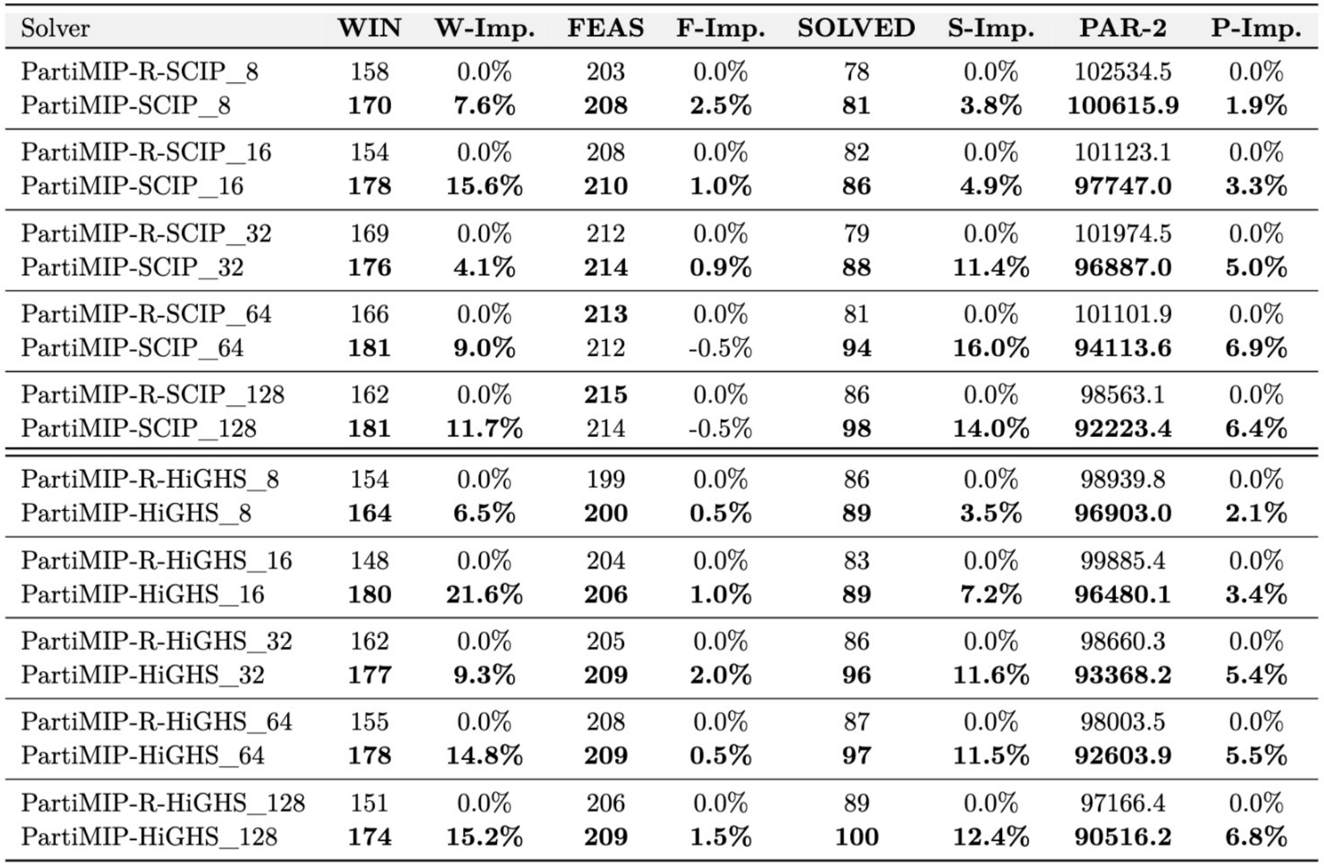
PartiMIP significantly enhance the performance of sequential MIP solvers



Solver	WIN	W-Imp.	FEAS	F-Imp.	SOLVED	S-Imp.	PAR-2	P-Imp.
SCIP_Sequential	85	0.0%	198	0.0%	73	0.0%	105616.9	0.0%
PartiMIP-SCIP_8	110	29.4%	208	5.1%	81	11.0%	100615.9	4.7%
PartiMIP-SCIP_16	128	50.6%	210	6.1%	86	17.8%	97747.0	7.5%
PartiMIP-SCIP_32	136	60.0%	214	8.1%	88	20.5%	96887.0	8.3%
PartiMIP-SCIP_64	142	67.1%	212	7.1%	94	28.8%	94113.6	10.9%
PartiMIP-SCIP_128	149	75.3%	214	8.1%	98	34.2%	92223.4	12.7%
HiGHS_Sequential	91	0.0%	191	0.0%	76	0.0%	103461.3	0.0%
PartiMIP-HiGHS_8	108	18.7%	200	4.7%	89	17.1%	96903.0	6.3%
PartiMIP-HiGHS_16	118	29.7%	206	7.9%	89	17.1%	96480.2	6.7%
PartiMIP-HiGHS_32	120	31.9%	209	9.4%	96	26.3%	93368.3	9.8%
PartiMIP-HiGHS_64	138	51.6%	209	9.4%	97	27.6%	92603.9	10.5%
PartiMIP-HiGHS_128	148	62.6%	209	9.4%	100	31.6%	90516.2	12.5%

Ablation Study

- We compared PartiMIP against a modified version
  - that uses random variable selection (PartiMIP-R).
- Our reward-guided method shows consistent and significant outperformance.



Solver	WIN	W-Imp.	FEAS	F-Imp.	SOLVED	S-Imp.	PAR-2	P-Imp.
PartiMIP-R-SCIP_8	158	0.0%	203	0.0%	78	0.0%	102534.5	0.0%
PartiMIP-SCIP_8	**170**	**7.6%**	**208**	**2.5%**	**81**	**3.8%**	**100615.9**	**1.9%**
PartiMIP-R-SCIP_16	154	0.0%	208	0.0%	82	0.0%	101123.1	0.0%
PartiMIP-SCIP_16	**178**	**15.6%**	**210**	**1.0%**	**86**	**4.9%**	**97747.0**	**3.3%**
PartiMIP-R-SCIP_32	169	0.0%	212	0.0%	79	0.0%	101974.5	0.0%
PartiMIP-SCIP_32	**176**	**4.1%**	**214**	**0.9%**	**88**	**11.4%**	**96887.0**	**5.0%**
PartiMIP-R-SCIP_64	166	0.0%	**213**	0.0%	81	0.0%	101101.9	0.0%
PartiMIP-SCIP_64	**181**	**9.0%**	212	-0.5%	**94**	**16.0%**	**94113.6**	**6.9%**
PartiMIP-R-SCIP_128	162	0.0%	**215**	0.0%	86	0.0%	98563.1	0.0%
PartiMIP-SCIP_128	**181**	**11.7%**	214	-0.5%	**98**	**14.0%**	**92223.4**	**6.4%**
PartiMIP-R-HiGHS_8	154	0.0%	199	0.0%	86	0.0%	98939.8	0.0%
PartiMIP-HiGHS_8	**164**	**6.5%**	**200**	**0.5%**	**89**	**3.5%**	**96903.0**	**2.1%**
PartiMIP-R-HiGHS_16	148	0.0%	204	0.0%	83	0.0%	99885.4	0.0%
PartiMIP-HiGHS_16	**180**	**21.6%**	**206**	**1.0%**	**89**	**7.2%**	**96480.1**	**3.4%**
PartiMIP-R-HiGHS_32	162	0.0%	205	0.0%	86	0.0%	98660.3	0.0%
PartiMIP-HiGHS_32	**177**	**9.3%**	**209**	**2.0%**	**96**	**11.6%**	**93368.2**	**5.4%**
PartiMIP-R-HiGHS_64	155	0.0%	208	0.0%	87	0.0%	98003.5	0.0%
PartiMIP-HiGHS_64	**178**	**14.8%**	**209**	**0.5%**	**97**	**11.5%**	**92603.9**	**5.5%**
PartiMIP-R-HiGHS_128	151	0.0%	206	0.0%	89	0.0%	97166.4	0.0%
PartiMIP-HiGHS_128	**174**	**15.2%**	**209**	**1.5%**	**100**	**12.4%**	**90516.2**	**6.8%**

Future Works

- Extend the experiment time limits
- More sophisticated selection and branching strategies
- Integration with commercial solvers
- Leverage more base solvers' internal information
    - e.g, node number, global cuts

Thank You!
Q&A