An Efficient Local Search Solver for Mixed Integer Programming

Peng Lin, Mengchuan Zou, and Shaowei Cai*

Institute of Software, Chinese Academy of Sciences, Beijing, China

School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China

{linpeng, zoumc, caisw}@ios.ac.cn

2024.09.04

Presenter

* Corresponding author

Outline

Background

- Mixed Integer Programming
- Local Search

Local-MIP

- Tailored Operators
- Weighting Scheme
- Two-level Scoring Function Structure
- Search Process of Local-MIP

Experiments

- Comparison with SOTA Solvers
- New Records to Open Instances

Mixed Integer Programming

$$\min \quad \mathbf{c}^\top \boldsymbol{x}$$

Objective Function

$$\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$$

General Linear Constraints

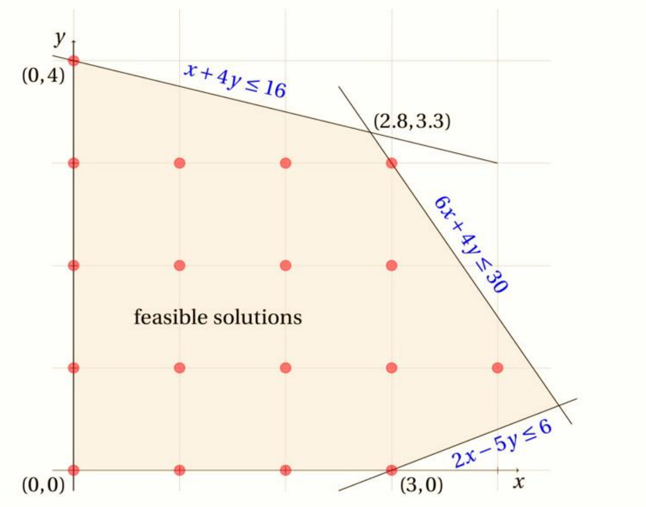
$$\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}$$

Global Bounds

$$\boldsymbol{x} \in \mathbb{R}^n, \;\; x_j \in \mathbb{Z} \;\; for \; all \;\; j \in I$$

Integrality Constraints

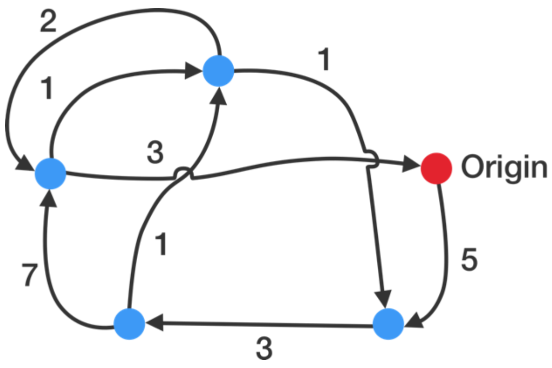
NP-Hard



- Solution $\boldsymbol{s}$: values assigned for each variable
  - s_j : the value of x_j
- Feasible solution ⇔ satisfies all constraints
- Lower objective value indicates higher quality

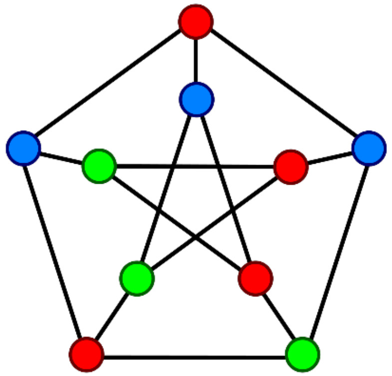
Mixed Integer Programming

Powerful Expressive Ability



Traveling Salesman Problem

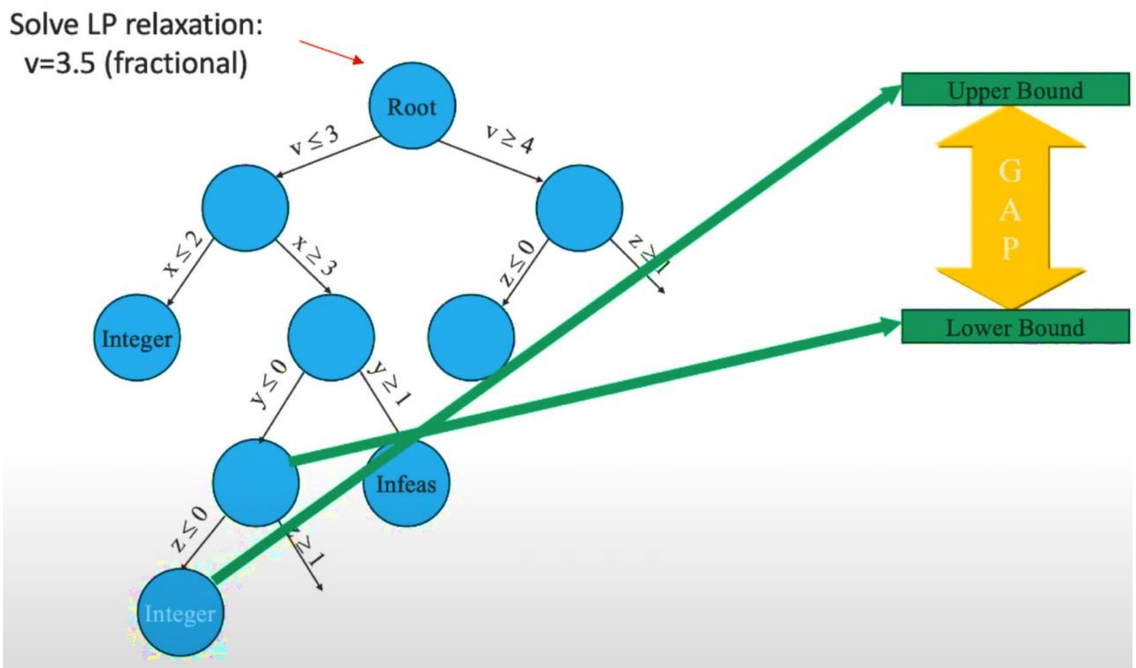
Bin Packing



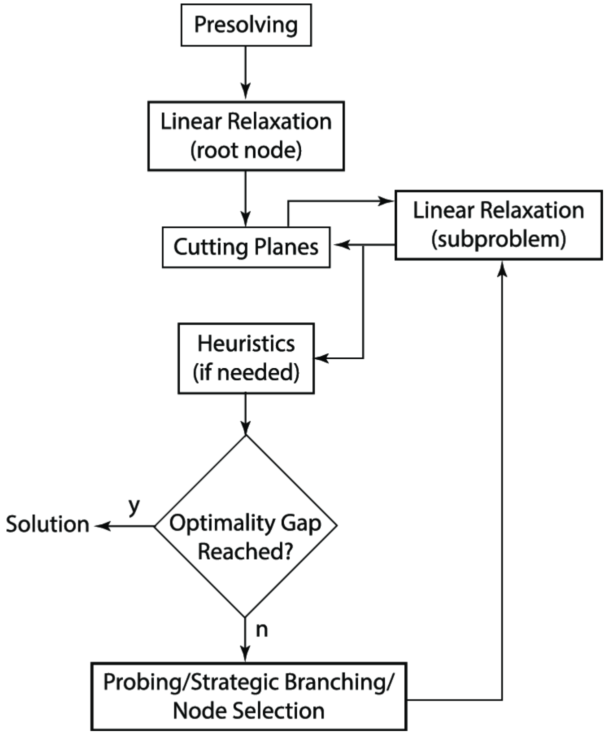
Many Graph Problems

Extensive Practical Applications

Branch-and-Bound



Branch-and-Cut



Exact Algorithms

- **Compute the optimal solution or prove infeasibility**
- **Large-scale practical instances are difficult to handle**

State-of-the-art MIP Solvers

Academic/Open-source Solvers

SCIP
[Achterberg, 2009]

HiGHS
[Huangfu and Hall, 2018]

OR-Tools
https://developers.google.com

CBC
https://www.coin-or.org/

Commercial Solvers

Gurobi
https://www.gurobi.com/

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

Xpress
https://www.fico.com/en/products/fico-xpress-optimization

COPT
https://www.shanshu.ai/copt/

- Almost all state-of-the-art MIP solvers are based on the branch-and-cut framework
- Note
  - **SCIP** and **HiGHS** are the top two academic solvers
  - Overall, **Gurobi** is the most powerful in most cases.

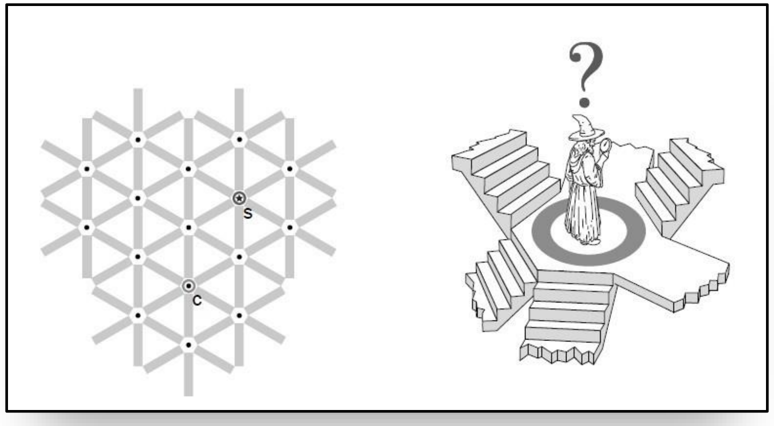
Local search is an efficient heuristic method for solving NP hard problems

Local Search
- **Find high-quality solutions quickly**
- **High-quality solutions are good usability in practice**

- **Search space $\boldsymbol{S}$:** define the **neighborhood relation $\boldsymbol{N} \subseteq \boldsymbol{S} \times \boldsymbol{S}$**
- **Operator :** define how to modify variables to generate candidate solutions, characterizing **neighborhood**
- **Scoring functions:** evaluate different candidate solutions to update the current solution

Local Search Process

- Starting from a initial solution
- Iteratively performing neighbor operation until time limit



Local search for **special cases** of MIP have been proposed

- Pseudo-Boolean optimization (0-1 programming) [Beresnev et al., 2012; Chu et al., 2023]
- Pure integer programming [Prestwich et al., 2008; Lin et al., 2023]
- MIP without the objective function [Luteberget et al., 2023]

Main challenges to developing an efficient local search solver for MIP

- Enhancing adaptability
  - Solve general formulation rather than specific forms
- Balance the optimization and the satisfaction
  - These two factors are sometimes conflicting

Main techniques in our solver Local-MIP

Breakthrough move: operator for optimization

Mixed tight move: operator for satisfaction

Weighting scheme: balance the priority of search

Two-level scoring function structure

- First level: **progress score**
- Second level: **bonus score**

Operator for Optimization: Breakthrough Move

- **Improve the objective function aiming to break through the best-found solution**

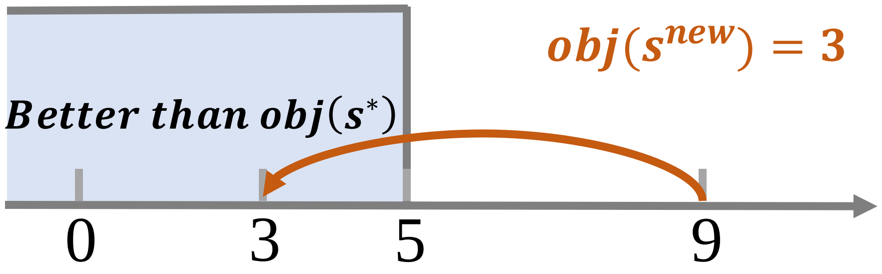
Given a variable x_j that appears in the objective function (i.e., $c_j \neq 0$), and a solution s that $obj(s) \geq obj(s^*)$, the breakthrough move operator, $\boldsymbol{bm(x_j, s)}$, assigns a variable x_j to the threshold value making the objective value better than $obj(s^*)$ as possible and keeping x_j's bounds satisfied.

EXAMPLE

Given an integer variable $\boldsymbol{x_1}$ with $\boldsymbol{1 \leq x_1 \leq 5}$; real variable $\boldsymbol{x_2}$ with $\boldsymbol{1 \leq x_2 \leq 3}$

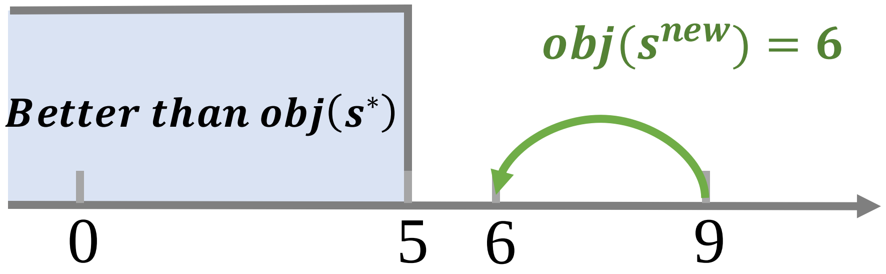
Given $\boldsymbol{obj = 3x_1 - 2x_2}$; $\boldsymbol{obj(s^*) = 5}$; $\boldsymbol{s^{cur} = \{s_1 = 4, s_2 = 1.5\}}$; $\boldsymbol{obj(s^{cur}) = 9}$

$\boldsymbol{bm(x_1, s^{cur})\text{: } assign\ \ x_1\ to\ 2}$



$\boldsymbol{Optimize\ obj\ to\ break\ through\ obj(s^*)}$

$\boldsymbol{bm(x_2, s^{cur})\text{: } assign\ \ x_2\ to\ 3}$



$\boldsymbol{Optimize\ obj\ and\ keep\ x_2\text{'s } bounds\ satisfied}$

Operator for Satisfaction: Mixed Tight Move

- **Satisfy a constraint as tight as possible**

Given a variable x_j , a constraint con_i containing x_j (i.e., $A_{ij} \neq 0$), and a solution s, the mixed tight move operator, $\boldsymbol{mtm(x_j, con_i, s)}$, assigns x_j to the threshold value making the constraint con_i satisfied and tight as possible while keeping x_j's bounds satisfied.

EXAMPLE

Integer variable $\boldsymbol{x_1}$ with $\mathbf{1} \leq \boldsymbol{x_1} \leq \mathbf{5}$; a real variable $\boldsymbol{x_2}$ with $\mathbf{1} \leq \boldsymbol{x_2} \leq \mathbf{3}$; $\boldsymbol{s^{cur}} = \{\boldsymbol{s_1} = \mathbf{4}, \boldsymbol{s_2} = \mathbf{1.5}\}$;

Given $\boldsymbol{con_1}$: $\boldsymbol{x_1} + \boldsymbol{x_2} \leq \mathbf{4.5}$

$\boldsymbol{mtm(x_1, con_1, s^{cur})}$: $\boldsymbol{assign\ \ x_1\ to\ 3}$

$\boldsymbol{mtm(x_2, con_1, s^{cur})}$: $\boldsymbol{assign\ \ x_2\ to\ 1}$

$\boldsymbol{satisfy\ con_1\ and}$
$\boldsymbol{have\ the\ least\ impact\ on\ other\ constraints}$

Given $\boldsymbol{con_2}$: $-\boldsymbol{x_1} + \boldsymbol{x_2} \leq -\mathbf{1}$

$\boldsymbol{mtm(x_1, con_2, s^{cur})}$: $\boldsymbol{assign\ \ x_1\ to\ 3}$

$\boldsymbol{mtm(x_2, con_2, s^{cur})}$: $\boldsymbol{assign\ \ x_2\ to\ 3}$

$\boldsymbol{takes\ the\ maximal\ change\ of\ variable\ and}$
$\boldsymbol{remains\ con_2\ satisfied}$

Balance the Priority of Search: Weighting Scheme

Weighting schemes : adjust the priority of constraints by diversified weights in search process

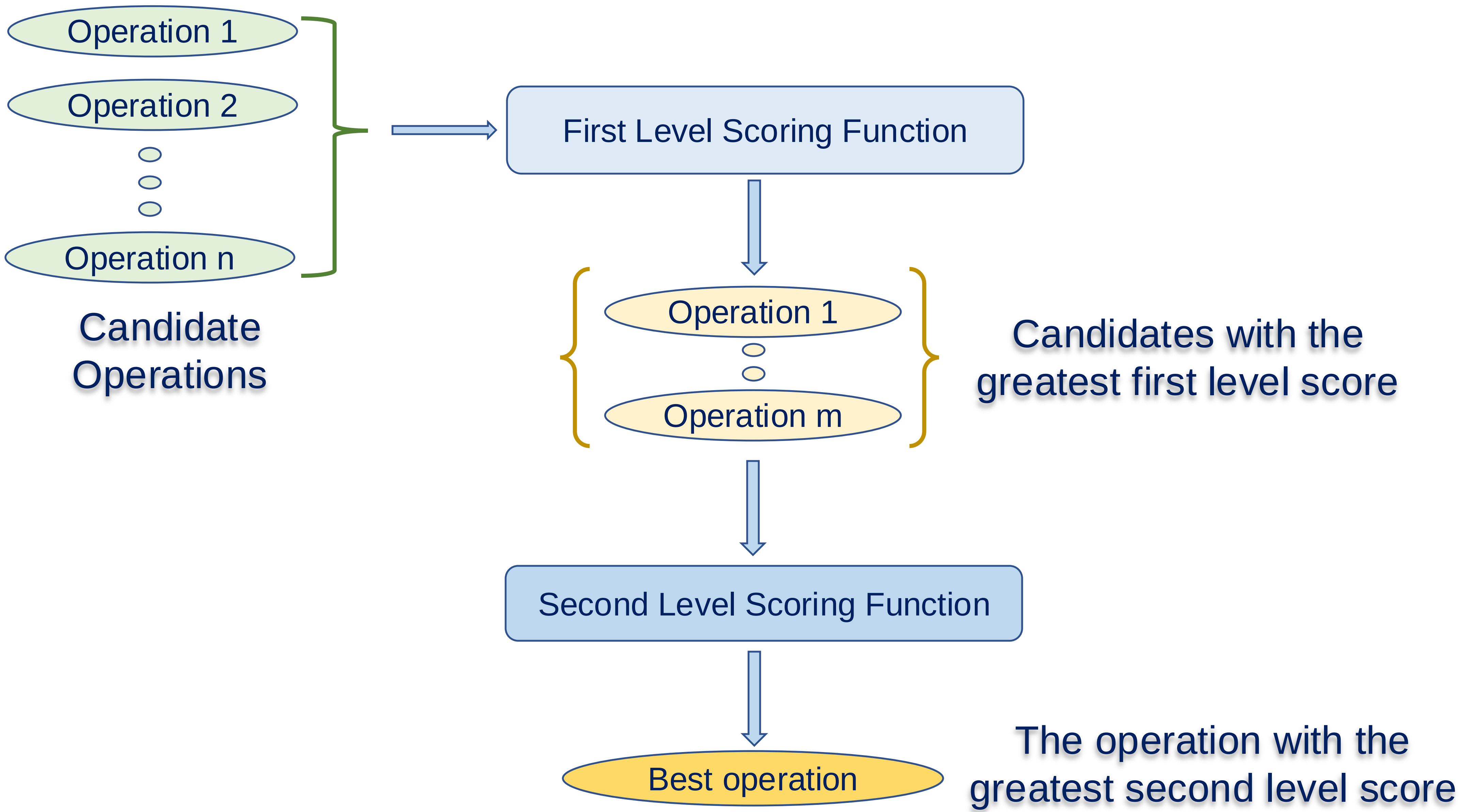
Dynamically balance the weights of the objective function and each constraint

When the search process is trapped in a local optimum

The current weights are unable to effectively guide the search

$$\begin{cases} w(obj) = w(obj) + 1, \text{ if the current solution is feasible} \\ w(con_i) = w(con_i) + 1 \text{ for each violated constraint } con_i, \text{ otherwise} \end{cases}$$

Two-level Scoring Function Structure



Two-level Scoring Function Structure

First Level Scoring Function: **Progress Score**	
Better objective value	Less unsatisfied constraints

↓

Second Level Scoring Function: **Bonus Score**	
Better best-found solution	Robust satisfaction

First Level: Progress Score

Progress Score for the Objective Function

▶ **Definition 4.** *Given an operation op, and the current solution $\boldsymbol{s}^{cur}$. Let $\boldsymbol{s}^{op}$ be the new candidate solution generated by performing op on $\boldsymbol{s}^{cur}$. The progress score of op for improving the quality of the objective value, denoted as $score^{obj}_{progress}(op)$,*

$$score^{obj}_{progress}(op) = \begin{cases} w(obj), & if\ obj(\boldsymbol{s}^{op}) < obj(\boldsymbol{s}^{cur}), \\ -w(obj), & if\ obj(\boldsymbol{s}^{op}) > obj(\boldsymbol{s}^{cur}), \\ 0, & else. \end{cases} \tag{6}$$

Progress Score for Each Constraint

▶ **Definition 5.** *Given an operation op, a constraint con_i, and the current solution $\boldsymbol{s}^{cur}$. Let $\boldsymbol{s}^{op}$ be the new candidate solution generated by performing op on $\boldsymbol{s}^{cur}$. The progress score of op for improving the satisfaction of the constraint con_i, denoted as $score^{con_i}_{progress}(op)$,*

$$score^{con_i}_{progress}(op) = \begin{cases} w(con_i), & if\ \boldsymbol{A_i} \cdot \boldsymbol{s}^{op} \le b_i < \boldsymbol{A_i} \cdot \boldsymbol{s}^{cur}, \\ -w(con_i), & if\ \boldsymbol{A_i} \cdot \boldsymbol{s}^{cur} \le b_i < \boldsymbol{A_i} \cdot \boldsymbol{s}^{op}, \\ w(con_i)/2, & if\ b_i < \boldsymbol{A_i} \cdot \boldsymbol{s}^{op} < \boldsymbol{A_i} \cdot \boldsymbol{s}^{cur}, \\ -w(con_i)/2, & if\ b_i < \boldsymbol{A_i} \cdot \boldsymbol{s}^{cur} < \boldsymbol{A_i} \cdot \boldsymbol{s}^{op}, \\ 0, & else. \end{cases} \tag{7}$$

Overall Progress Score

▶ **Definition 6.** *Given an operation op, the progress score of op, denoted as $score_{progress}(op)$,*

$$score_{progress}(op) = score^{obj}_{progress}(op) + \sum_{i=1}^{m} score^{con_i}_{progress}(op) \tag{8}$$

Second Level: Bonus Score

Breakthrough Bonus for the Objective Function

▶ **Definition 7.** *Given an operation op, and the best-found solution* $\boldsymbol{s}^*$. *Let* $\boldsymbol{s}^{op}$ *be the new candidate solution generated by performing op on the current solution. The breakthrough bonus of op for breaking through the objective value of* $\boldsymbol{s}^*$, *denoted as* $bonus_{break}(op)$,

$$bonus_{break}(op) = \begin{cases} w(obj), & if\ obj(\boldsymbol{s}^{op}) < obj(\boldsymbol{s}^*), \\ 0, & otherwise. \end{cases} \tag{9}$$

Robustness Bonus for Each Constraint

▶ **Definition 8.** *Given an operation op, a constraint* con_i. *Let* $\boldsymbol{s}^{op}$ *be the new candidate solution generated by performing op on the current solution. The robustness bonus of op of the constraint* con_i, *denoted as* $bonus_{robust}^{con_i}(op)$,

$$bonus_{robust}^{con_i}(op) = \begin{cases} w(con_i), & if\ \boldsymbol{A_i} \cdot \boldsymbol{s^{op}} < b_i, \\ 0, & otherwise. \end{cases} \tag{10}$$

Overall Bonus Score

▶ **Definition 9.** *Given an operation op, the bonus score of op, denoted as* $score_{bonus}(op)$,

$$score_{bonus}(op) = bonus_{break}(op) + \sum_{i=1}^{m} bonus_{robust}^{con_i}(op) \tag{11}$$

Search Process of Local-MIP

Algorithm 1 The Local-MIP Algorithm

Input: MIP instance Q, time limit *cutoff*

Output: Best-found solution s^* of Q and its objective value $obj(s^*)$

$\boldsymbol{s^{cur}} \leftarrow$ all variables are set to the value closest to 0 within their global bounds;
$\boldsymbol{s^*} \leftarrow \emptyset$; $obj(\boldsymbol{s^*}) \leftarrow +\infty$;
while *running time* $<$ *cutoff* **do**
    if $\boldsymbol{s^{cur}}$ *is feasible* **then**
        Improve the objective value while maintaining feasibility by lift move process;
        if $obj(\boldsymbol{s^{cur}}) < obj(\boldsymbol{s^*})$ **then**
            $\boldsymbol{s^*} \leftarrow \boldsymbol{s^{cur}}$; $obj(\boldsymbol{s^*}) \leftarrow obj(\boldsymbol{s^{cur}})$;
    $\boldsymbol{candOP} \leftarrow$ $\boxed{Get\_Candidate\_Operations(Q, \boldsymbol{s^{cur}})}$;
    $\boldsymbol{candOP^+} \leftarrow$ operation(s) with the greatest progress score in $\boldsymbol{candOP}$;
    $\boldsymbol{op} \leftarrow$ an operation with the greatest bonus score in $\boldsymbol{candOP^+}$;
    $\boldsymbol{s^{cur}} \leftarrow$ a new solution generated by performing $\boldsymbol{op}$ to modify $\boldsymbol{s^{cur}}$;
return $\boldsymbol{s^*}$ and $obj(\boldsymbol{s^*})$;

Breakthrough Move
Mixed Tight Move

Two-level Scoring
Function Structure

Experiment Preliminary

HiGHS

SCIP
Solving Constraint Integer Programs

IBM
CPLEX

GUROBI
OPTIMIZATION

Feasibility Jump

Competitors

Benchmark	#Inst
MIPLIB-BP	66
MIPLIB-IP	32
MIPLIB-MBP	195
MIPLIB-MIP	62
BBP	60
JSP	80
OSP	60
Total	555

Benchmark

10s

60s

300s

Time Limits

Experiments

Comparison with SOTA Solvers

Comparison

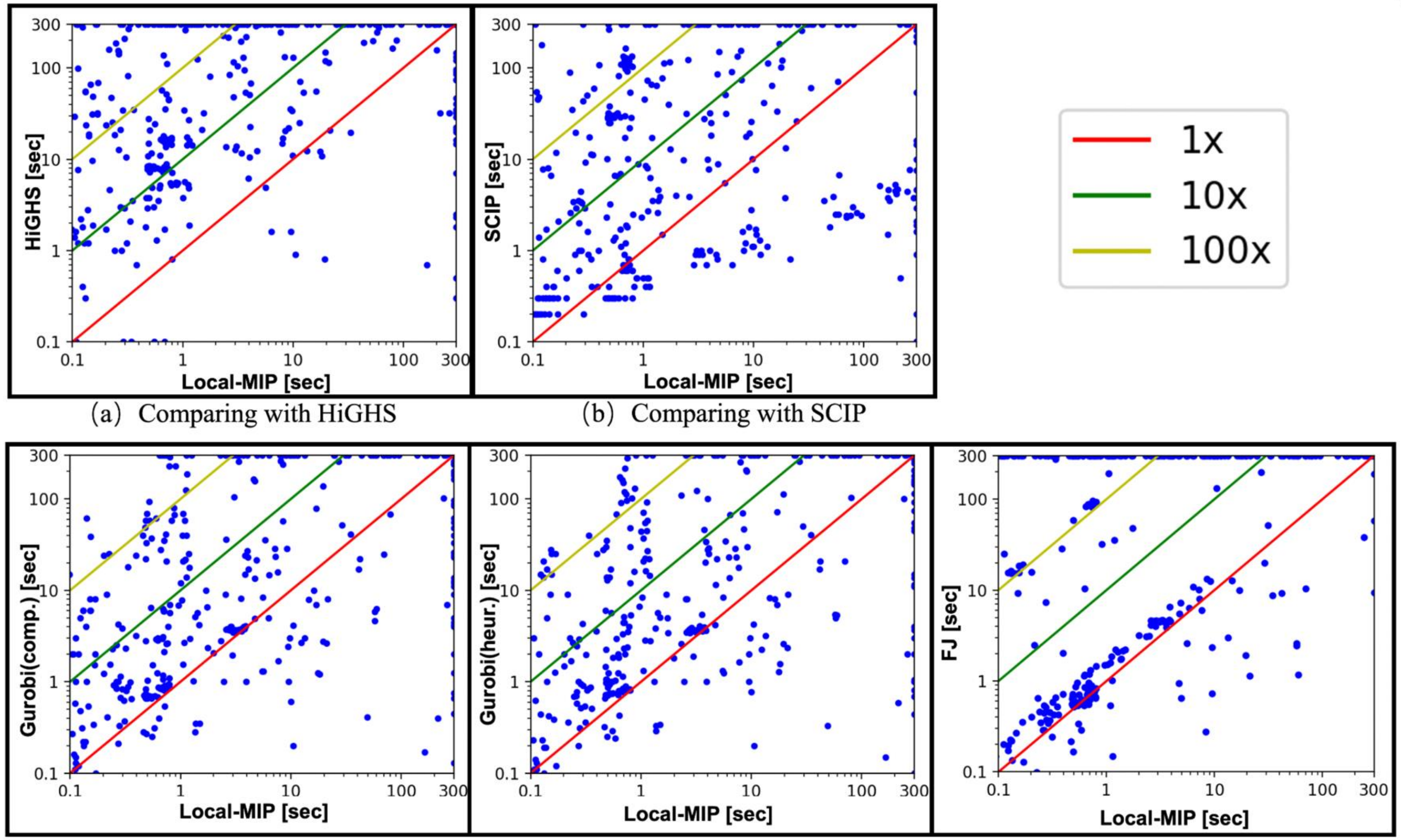
in terms of

$\#Feas$ and $\#Win$

- Outperforms CPLEX , HiGHS, SCIP and Feasibility Jump
- Competitive with the most powerful commercial solver Gurobi

Benchmark	#Inst	*HiGHS*		*SCIP*		*CPLEX*		$\text{Gurobi}_{\text{comp}}$		$\text{Gurobi}_{\text{heur}}$		*FJ*		*Local-MIP*	
		#Feas	#Win	#Feas	#Win	#Feas	#Win	#Feas	#Win	#Feas	#Win	#Feas	#Win	#Feas	#Win
time limit 10 seconds															
MIPLIB-BP	66	6	0	29	2	42	11	**44**	9	**44**	7	42	4	**44**	**31**
MIPLIB-IP	32	7	0	11	2	**17**	4	**17**	**8**	**17**	**8**	11	1	**17**	7
MIPLIB-MBP	195	57	1	80	7	116	31	117	43	**119**	**51**	56	10	103	35
MIPLIB-MIP	62	9	2	21	0	32	10	**37**	11	**37**	13	18	4	35	**15**
BPP	60	9	0	0	0	**60**	0	**60**	0	**60**	0	**60**	0	**60**	**60**
JSP	80	22	0	**70**	25	31	0	10	1	12	8	0	0	45	**36**
OSP	60	48	22	**60**	20	28	7	47	27	42	25	1	0	**60**	**45**
Total	555	158	25	271	56	326	63	332	99	331	112	188	19	**364**	**229**
time limit 60 seconds															
MIPLIB-BP	66	14	0	35	2	43	8	46	10	47	15	**49**	2	48	**28**
MIPLIB-IP	32	12	1	14	1	20	6	20	7	20	**9**	12	1	**21**	6
MIPLIB-MBP	195	96	6	109	2	129	32	**137**	49	134	**65**	62	9	119	23
MIPLIB-MIP	62	15	3	28	1	36	7	41	8	41	**18**	20	3	**43**	17
BPP	60	40	0	20	0	**60**	11	**60**	13	**60**	15	**60**	0	**60**	**33**
JSP	80	41	0	**70**	15	52	1	23	3	26	13	1	0	54	**38**
OSP	60	58	27	**60**	20	30	10	53	37	51	31	9	0	**60**	**42**
Total	555	276	37	336	41	370	75	380	127	379	166	213	15	**405**	**187**
time limit 300 seconds															
MIPLIB-BP	66	22	1	42	4	43	6	47	10	48	**23**	**49**	0	**49**	17
MIPLIB-IP	32	14	2	17	2	21	5	21	10	**22**	**14**	12	1	**22**	4
MIPLIB-MBP	195	115	7	122	7	137	22	150	59	**152**	**69**	67	11	123	14
MIPLIB-MIP	62	24	1	34	1	38	7	44	10	43	**23**	21	3	**45**	16
BPP	60	47	0	40	0	**60**	**31**	**60**	30	**60**	13	**60**	0	**60**	3
JSP	80	49	0	**70**	0	68	1	36	10	34	20	1	0	**70**	**41**
OSP	60	**60**	38	**60**	24	33	13	55	40	**60**	43	19	0	**60**	**44**
Total	555	331	49	385	38	400	85	413	169	419	**205**	229	15	**429**	139

Run time comparison on each instance for finding the first feasible solution



(a) Comparing with HiGHS
(b) Comparing with SCIP
(c) Comparing with Gurobi(comp.)
(d) Comparing with Gurobi(heur.)
(e) Comparing with FJ

There are obviously more instances above the red line, which confirms the powerful solving ability of Local-MIP.

New Records to Open Instances

Instance name	#var	#cons	constraint types	Previous best	*Local-MIP*
genus-sym-g31-8	3484	32073	knapsack, precedence, etc.	-21	-23
genus-sym-g62-2	12912	78472	set partitioning, set covering, etc.	-34	-38
genus-g61-25	14380	94735	cardinality, general linear, etc.	-34	-40
neos-4232544-orira	87060	180600	aggregations, variable bound, etc.	17540506.0	15108527.512195

Each of these 4 instances contains multiple different constraint types, simultaneously indicating the powerful solving ability and its extensive applicability.

- https://miplib.zib.de/instance_details_genus-sym-g31-8.html
- https://miplib.zib.de/instance_details_genus-sym-g62-2.html
- https://miplib.zib.de/instance_details_genus-g61-25.html
- https://miplib.zib.de/instance_details_neos-4232544-orira.html

Analysis of Local-MIP

Breakthrough Move Operator

V_{no-bm}
- removing all the breakthrough move operations

Weighting Scheme

$V_{no-weight}$
- removing the activation of the weighting scheme

Bonus Score

V_{random} and V_{age}
- utilizing the random selection and the age strategy instead of bonus score to break ties

Benchmark	#Inst	10 seconds		60 seconds		300 seconds		10 seconds		60 seconds		300 seconds	
		#better	#worse	#better	#worse	#better	#worse	#better	#worse	#better	#worse	#better	#worse
		Comparison with V_{no-bm}						Comparison with $V_{no-weight}$					
MIPLIB-BP	66	**28**	7	**23**	17	**24**	18	**33**	4	**34**	7	**34**	8
MIPLIB-IP	32	**10**	2	**11**	5	**12**	4	**15**	0	**19**	0	**20**	0
MIPLIB-MBP	195	**61**	20	**68**	26	**63**	35	**95**	5	**112**	5	**116**	5
MIPLIB-MIP	62	**22**	7	**27**	8	**27**	8	**35**	0	**40**	1	**41**	0
BPP	60	**59**	0	**59**	0	**58**	0	**35**	10	**56**	0	**60**	0
JSP	80	**32**	13	**45**	8	**60**	10	**45**	0	**54**	0	**70**	0
OSP	60	**48**	5	**49**	1	**46**	3	**60**	0	**60**	0	**60**	0
Total	555	**260**	54	**282**	65	**290**	78	**318**	19	**375**	13	**401**	13
		Comparison with V_{random}						Comparison with V_{age}					
MIPLIB-BP	66	**26**	10	**27**	15	**26**	15	**27**	10	**23**	16	**26**	13
MIPLIB-IP	32	**10**	4	**13**	3	**15**	4	**11**	3	**15**	1	**15**	1
MIPLIB-MBP	195	**69**	27	**80**	30	**71**	43	**66**	28	**68**	42	**71**	41
MIPLIB-MIP	62	**22**	12	**19**	18	**18**	17	**23**	10	**23**	13	**22**	15
BPP	60	**28**	15	11	**27**	11	**34**	**34**	11	16	**29**	13	**25**
JSP	80	**29**	16	**31**	23	**39**	31	**30**	15	**31**	23	**42**	26
OSP	60	**25**	20	**27**	15	**21**	12	**29**	18	**31**	13	**22**	13
Total	555	**209**	104	**208**	131	**201**	156	**220**	95	**207**	137	**211**	134

Conclusion and Future Work

Deep cooperation of local search and branch-and-bound

More sophisticated operators and scoring functions to improve Local-MIP

Local-MIP: an efficient local search solver for mixed integer programming

Our code: https://github.com/shaowei-cai-group/Local-MIP

Thank You!
Q&A