ParalLP: A Parallel Local Search Framework for Integer Linear Programming with Cooperative Evolution Mechanism

Peng Lin, Mengchuan Zou, Zhihan Chen, and **Shaowei Cai***

Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science,

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, chenzh, caisw}@ios.ac.cn

08.08.2024

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

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

Outline

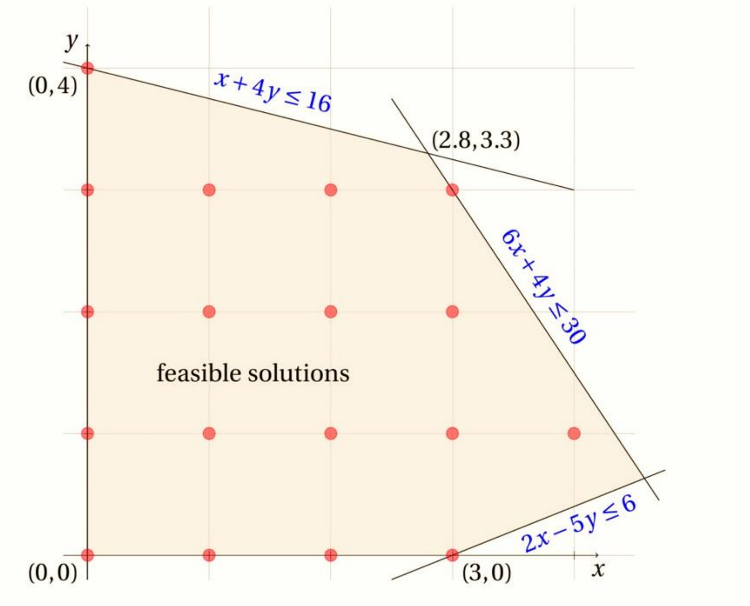
- **Background and Motivation**
- **Parallel Framework for Solving ILP**
- **Experiments and Conclusions**

Integer Linear Programming

Let $m, n \in \mathbb{N}^+$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^n$, and $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n$. The optimization problem described by:

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to:} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \\ & \boldsymbol{x} \in \mathbb{Z}^n \end{aligned} \tag{1}$$

is an instance of general integer linear programming (ILP).



- **A fundamental model in operations research**
- **NP-Hard**
- **Strong descriptive capability**

Complete and Incomplete Algorithms

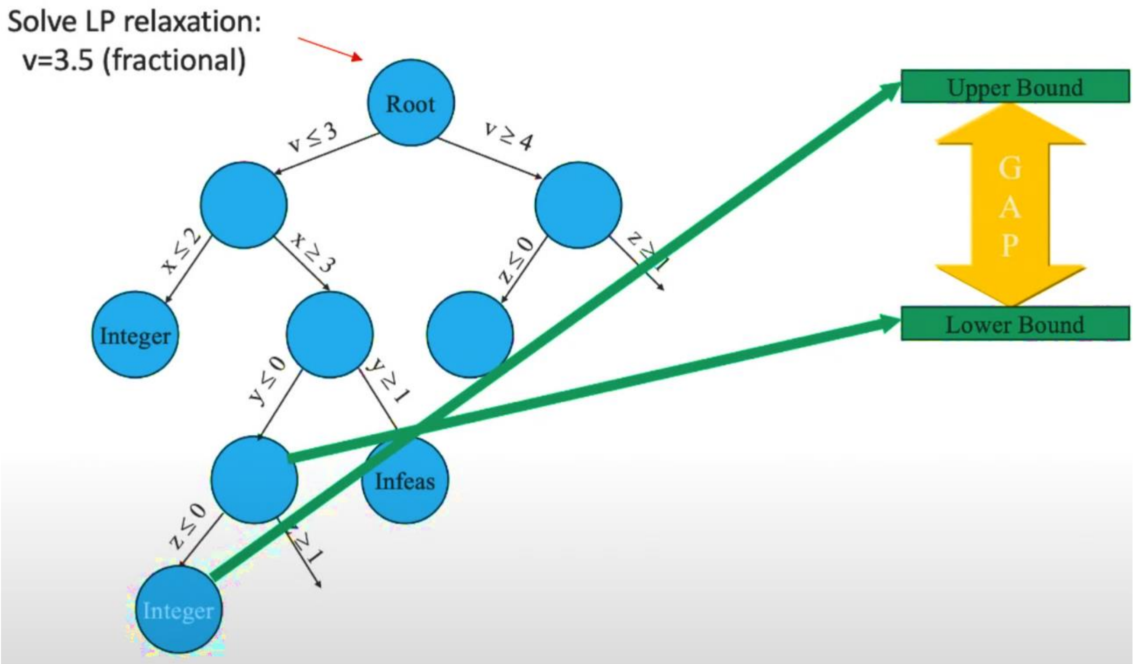
Complete Algorithms

- compute the optimal solution
- prove infeasibility

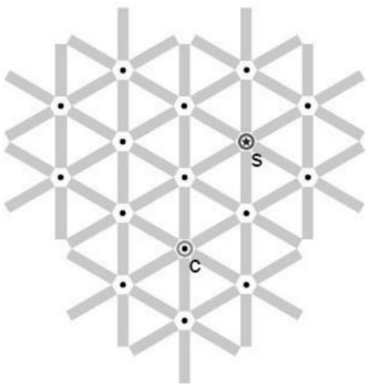
Incomplete Algorithms

- find high-quality solutions quickly

Branch-and-Bound



Local Search



Local-ILP [Lin et al., 2023]

Academic Solver

- SCIP [Achterberg, 2009]
- HiGHS [Huangfu and Hall, 2018]

Commercial Solver

- Gurobi
- CPLEX

Source code

- https://github.com/shaowei-cai-group/Local-ILP

Motivation

- Incomplete algorithms are important in industry applications of ILP
- The increasing computing power of multicore computer structures
- Although good serial local search algorithm has been proposed for ILP, the parallel local search has not been investigated
- Thus, we try to propose the first parallel local search framework for ILP

Outline

- Background and Motivation
- **Parallel Framework for Solving ILP**
- Experiments and Conclusions

Parallel Framework for Solving ILP

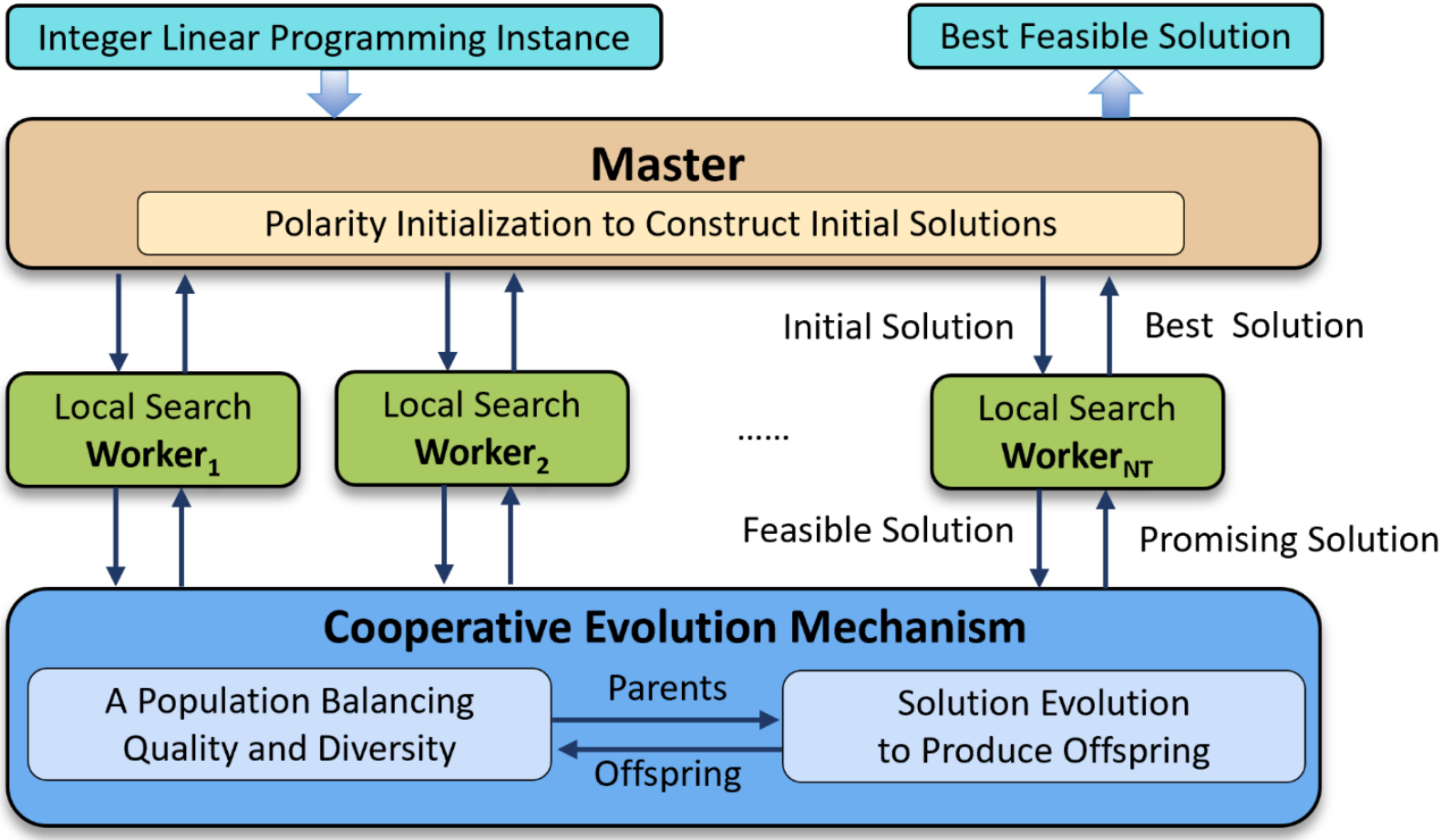


Figure 1: Our parallel local search framework for solving general integer linear programming with cooperative evolution mechanism.

How to generate high-quality initial solutions for each local search thread?

Definition 1. *The **influence factor** of variable x_j in the objective function $\mathbf{c}^\top \boldsymbol{x}$, denoted as IO_j, is defined as*

$$IO_j = \frac{(u_j - l_j) \cdot c_j}{\sum_{k=1}^{n} (u_k - l_k) \cdot |c_k|} \tag{2}$$

Similarly, the influence factor of variable x_j in constraint $\boldsymbol{A_i x} \le b_i$, denoted as IC_{ij}, is defined as

$$IC_{ij} = \frac{(u_j - l_j) \cdot A_{ij}}{\sum_{k=1}^{n} (u_k - l_k) \cdot |A_{ik}|} \tag{3}$$

Definition 2. *Let m_j denote the number of constraints containing x_j, the **polarity** of variable x_j, denoted as P_j, is defined as*

$$P_j = \frac{\sum_{i=0}^{m} IC_{ij}}{m_j} + IO_j \tag{4}$$

Polarity Initialization
- Polarity of variables (for high-quality)
- Random perturbations (for diversified solutions)

Manage a population balancing quality and diversity

The Fitness Function

$$R(\boldsymbol{s}) = R_Q(\boldsymbol{s}) \cdot p + R_D(\boldsymbol{s}) \cdot (1 - p)$$

The Objective Value

$Q(s)$ is initially set to $-obj(s)$

The Informative Degree:

- Whenever a solution's information is utilized, $Q(s)$ is penalized
- If $Q(s) > 0$, $Q(s)$ is updated as $Q(s) \times (1 - \beta)$
- if $Q(s) < 0$, $Q(s)$ is updated as $Q(s) \times (1 + \beta)$

The Differences from Other Solutions

$$D(\boldsymbol{s}) = \sum_{s' \in \text{Population}, s' \neq s} \sum_{j=1}^{n} |s_j - s'_j|$$

Generate new solutions from high-quality solutions in the population

Solution Evolution Process

Random Parent Selection

- All solutions in the population are highly competitive

Uniform Crossover

- Leverage the information from two selected parents

Bound-aware Mutation

- Explore new search space

- Background and Motivation
- Parallel Framework for Solving ILP
- **Experiments and Conclusions**

ParaILP is significantly better than the state-of-the-art academic parallel solvers FiberSCIP and HiGHS

Benchmark Domain	#Ins	#Feas									#Win									P(T)								
		HiGHS			FiberSCIP			ParaILP			HiGHS			FiberSCIP			ParaILP			HiGHS			FiberSCIP			ParaILP		
		10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s
Singleton	2	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1**	0	0	0	0	**1**	**1**	**2**	**1**	0.788	0.473	0.324	0.485	0.371	0.188	**0.109**	**0.088**	**0.081**
Aggregations	2	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	**1**	**1**	**1**	0	0	0	0	1.000	0.615	0.523	**0.672**	**0.531**	**0.508**	0.761	0.633	0.555
Bin Packing	2	0	**2**	**2**	0	1	1	**1**	**2**	**2**	0	**2**	0	0	0	**1**	**1**	0	**1**	1.000	**0.905**	**0.715**	1.000	1.000	0.993	**0.979**	0.923	0.773
Equation Knapsack	3	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Knapsack	4	3	3	3	3	3	3	**4**	**4**	**4**	1	1	**2**	1	1	0	**2**	**2**	**2**	0.529	0.388	0.304	0.412	0.316	0.282	**0.111**	**0.071**	**0.048**
Set Packing	5	2	**4**	**4**	3	**4**	**4**	**4**	**4**	**4**	1	1	1	0	0	1	**3**	**3**	**2**	1.000	0.802	0.609	0.759	0.549	0.323	**0.369**	**0.260**	**0.219**
Cardinality	6	1	1	2	1	1	2	**2**	**2**	**3**	0	0	**1**	0	0	**1**	**2**	**2**	**1**	0.958	0.820	0.746	0.866	0.830	0.753	**0.628**	**0.571**	**0.522**
Hybrid	7	3	3	4	2	4	4	**4**	**5**	**5**	0	0	1	0	0	0	**4**	**5**	**5**	1.000	1.000	1.000	1.000	1.000	0.699	**0.689**	**0.548**	**0.524**
Mixed Binary	8	0	0	0	2	2	2	**4**	**5**	**6**	0	0	0	1	2	2	**4**	**4**	**5**	1.000	1.000	1.000	1.000	0.980	0.976	**0.890**	**0.861**	**0.762**
Set Partitioning	9	1	2	4	3	4	6	**4**	**6**	**7**	0	0	0	0	2	3	**4**	**4**	**4**	1.000	0.999	0.927	0.930	0.884	0.805	**0.887**	**0.827**	**0.765**
Set Covering	11	5	6	8	5	8	7	**9**	**10**	**10**	2	0	2	0	**5**	**6**	**7**	**5**	3	0.861	0.780	0.656	0.780	**0.648**	**0.523**	**0.733**	0.650	0.610
Precedence	13	1	3	4	10	11	**12**	**12**	**12**	**12**	0	0	0	1	1	2	**11**	**11**	**10**	0.996	0.974	0.860	0.889	0.813	0.653	**0.576**	**0.454**	**0.345**
General Linear	15	3	4	6	7	7	9	**10**	**10**	**10**	2	2	4	2	1	3	**6**	**7**	**7**	0.780	0.700	0.591	0.712	0.575	0.425	**0.355**	**0.317**	**0.301**
Variable Bound	16	7	9	9	11	13	13	**14**	**15**	**15**	0	0	1	0	0	2	**14**	**15**	**14**	0.895	0.784	0.653	0.851	0.688	0.472	**0.381**	**0.278**	**0.228**
Invariant Knapsack	18	4	7	11	10	11	**13**	**12**	**13**	**13**	2	1	3	2	3	2	**12**	**12**	**11**	0.924	0.847	0.736	0.778	0.706	0.664	**0.549**	**0.475**	**0.421**
Total	121	32	47	60	60	72	79	**83**	**91**	**95**	9	7	16	8	16	24	**71**	**72**	**67**	0.911	0.831	0.730	0.818	0.724	0.610	**0.582**	**0.508**	**0.452**

Table 1: Performance evaluation between SOTA academic solvers HiGHS, FiberSCIP and ParaILP.

Comparison with Commercial Solver

ParaILP is competitive with the state-of-the-art commercial parallel solver Gurobi.

Benchmark Domain	#Ins	*#Feas*									*#Win*									*P(T)*								
		Gurobi$_{\text{comp}}$			Gurobi$_{\text{heur}}$			ParaILP			Gurobi$_{\text{comp}}$			Gurobi$_{\text{heur}}$			ParaILP			Gurobi$_{\text{comp}}$			Gurobi$_{\text{heur}}$			ParaILP		
		10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s	10s	60s	300s
Singleton	2	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	0	**2**	**1**	**1**	0	**1**	**1**	1	0	0.237	0.113	**0.042**	0.240	0.115	0.045	**0.109**	**0.088**	0.081
Aggregations	2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	0	0	0	**0.514**	**0.502**	**0.500**	0.519	0.503	0.501	0.761	0.633	0.555
Bin Packing	2	**2**	**2**	**2**	**2**	**2**	**2**	1	**2**	**2**	**2**	**2**	0	**2**	**2**	**1**	0	0	**1**	**0.963**	**0.748**	**0.303**	0.963	0.757	0.305	0.979	0.923	0.773
Equation Knapsack	3	0	0	**1**	0	0	0	0	0	**1**	0	0	**1**	0	0	0	0	0	**1**	1.000	1.000	**0.952**	1.000	1.000	1.000	1.000	1.000	1.000
Knapsack	4	3	3	**4**	3	3	**4**	**4**	**4**	**4**	1	**2**	2	1	1	**4**	**2**	**2**	1	0.317	0.271	0.100	0.315	0.274	0.093	**0.111**	**0.071**	**0.048**
Set Packing	5	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	2	1	**2**	**3**	**2**	**2**	1	**2**	**2**	0.408	0.271	0.231	0.407	0.271	0.230	**0.369**	**0.260**	**0.219**
Cardinality	6	**2**	**3**	**3**	**2**	**3**	**3**	**2**	2	**3**	**1**	**2**	1	0	1	**3**	**1**	1	0	0.741	**0.543**	0.364	0.741	0.543	**0.359**	**0.628**	0.571	0.522
Hybrid	7	**4**	**5**	**5**	**4**	**5**	**5**	**4**	**5**	**5**	1	1	2	1	1	**3**	**3**	**5**	2	0.777	0.660	0.548	0.784	0.664	0.546	**0.689**	**0.548**	**0.524**
Mixed Binary	8	2	3	3	2	3	3	**4**	**5**	**6**	1	0	1	1	1	1	**4**	**4**	**5**	0.986	0.975	0.960	0.986	0.970	0.958	**0.890**	**0.861**	**0.762**
Set Partitioning	9	**7**	**7**	**7**	**7**	**7**	**7**	4	6	**7**	4	**4**	3	**5**	**4**	**4**	2	2	2	**0.815**	0.748	0.625	0.815	**0.738**	**0.617**	0.887	0.827	0.765
Set Covering	11	**9**	9	9	**9**	9	9	**9**	**10**	**10**	**5**	5	4	2	**6**	**7**	4	2	2	0.616	0.384	0.291	**0.613**	**0.371**	**0.264**	0.733	0.650	0.610
Precedence	13	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	4	3	4	1	4	**9**	**8**	**6**	2	0.684	0.529	0.350	0.688	0.534	0.361	**0.576**	**0.454**	**0.345**
General Linear	15	9	**11**	**11**	8	**11**	**11**	**10**	10	10	**6**	5	6	3	**8**	**8**	4	3	3	0.421	**0.286**	**0.251**	0.460	0.295	0.254	**0.355**	0.317	0.301
Variable Bound	16	**14**	14	14	13	14	14	**14**	**15**	**15**	2	4	7	2	5	5	**12**	**10**	**8**	0.612	0.430	0.297	0.595	0.399	0.291	**0.381**	**0.278**	**0.228**
Invariant Knapsack	18	**12**	12	**15**	**12**	12	**15**	**12**	**13**	13	6	6	7	4	6	7	**9**	**10**	**9**	0.699	0.602	0.367	0.700	0.604	**0.367**	**0.549**	**0.475**	0.421
Total	121	**83**	88	93	81	88	92	**83**	**91**	**95**	36	38	42	26	42	**56**	**51**	**48**	38	0.653	0.526	0.398	0.656	0.522	**0.396**	**0.582**	**0.508**	0.452

Table 2: Performance evaluation between SOTA commercial solver Gurobi (both the exact and heuristic version) and ParaILP.

Conclusions

- The first parallel local search framework and an efficient parallel solver for solving general ILP.
- Our code: https://github.com/shaowei-cai-group/ParaILP
- Our framework could be easily extended by plugging other sequential local search algorithms as a subroutine to create new solvers.

Thank You!
Q&A