GPU-MIP: A GPU-Native Local Search Solver for Mixed-Integer Programming

Yiran Zhu syuizen@gmail.com
Independent Researcher*
*The author is employed by Amazon. This work was conducted independently and is not affiliated with, endorsed by, or representative of Amazon or its affiliates.

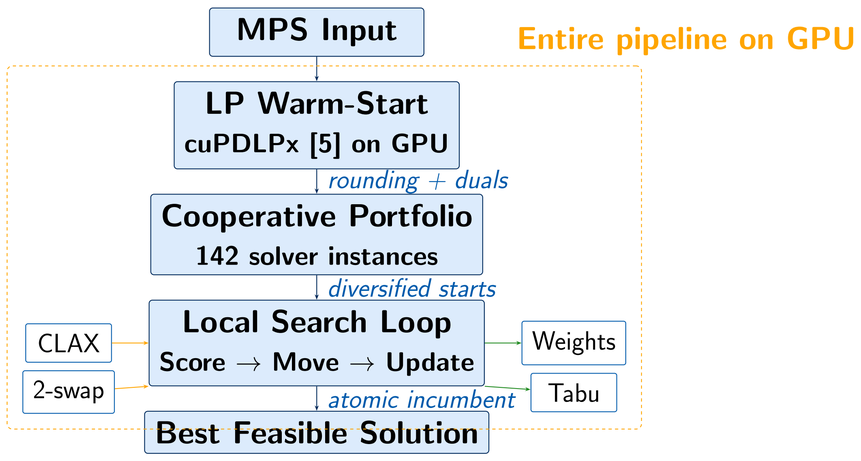
Problem & Motivation

Mixed-integer programming (MIP) is fundamental to combinatorial optimization. Commercial solvers (Gurobi, Xpress) rely on branch-and-bound with LP relaxations — inherently sequential.

Opportunity: GPUs offer massive parallelism (>10,000 threads) underutilized by traditional MIP algorithms [1]. Local search is naturally parallelizable.

Goal: GPU-native MIP solver competitive with commercial solvers within 300 s on the competition test set.

System Architecture



Unified Scoring Function

Each candidate move $x_j \leftarrow x_j + \delta$ is scored:

$$\text{score}(j,\delta) = \sum_{c:A_{cj}\neq 0} w_c \cdot \Delta_c + \begin{cases} w_{\text{obj}} \cdot \text{sign}(-\Delta_{\text{obj}}) & \text{if feasible} \\ 0 & \text{otherwise} \end{cases}$$

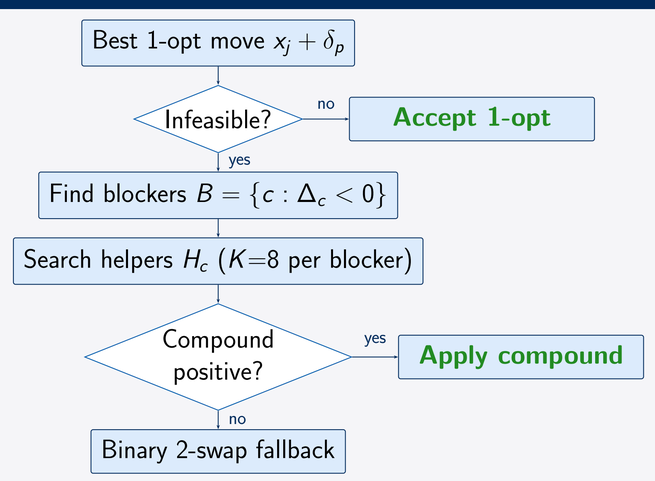
x_j — variable j's current value
δ — proposed change (move size)
w_c — integer weight of constraint c (updated adaptively)
Δ_c — satisfaction state transition for constraint c
w_{obj} — adaptive objective weight (grows while feasible)
Δ_{obj} — objective change $= c_j \cdot \delta$

Half-credit scoring extends Boolean SLS to linear constraints:

$$\Delta_c = \begin{cases} +1 & \text{violated} \to \text{satisfied}, \\ -1 & \text{satisfied} \to \text{violated}, \\ +\frac{1}{2} & \text{violated, moving closer}, \\ -\frac{1}{2} & \text{satisfied, moving toward violation}. \end{cases}$$

No phase separation — adaptive w_{obj} grows as solver stays feasible, providing smooth transition from repair to optimization.

CLAX: Compound Moves



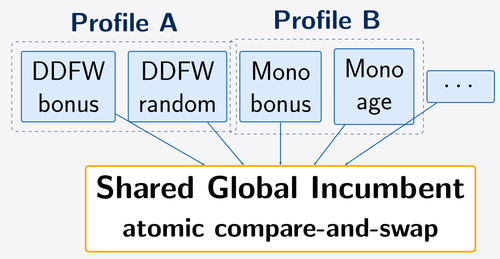
Neighborhood Strategies

1. **Repair:** Sample violated constraints (weighted by $w_c \cdot d_c$), compute tight δ to satisfy each
2. **Diversify:** Sample satisfied constraints, consume slack to explore new regions
3. **Reset:** Move variables toward natural values (zero, bounds, midpoint)
4. **Fallback:** 1 random violated constraint, weak tabu (only blocks immediate reverse)

exit on first positive score

Priority order with early exit on first positive score. Weight update before Fallback. Reservoir sampling ensures uniform candidate selection.

Cooperative Portfolio



Cooperation: stall resets · threshold tightening · pool crossover

142 instances across 4 profiles varying weight method (DDFW [3] vs. Monotone), tiebreak, tabu tenure. Perturbation: None, Gaussian, flip, RINS-destroy, zero-start.

Results

Setup: 50 instances, 300 s, L40S GPU, median of 5 seeds. Baselines: CPU Local-MIP [4], Xpress 9.8, Gurobi 12.

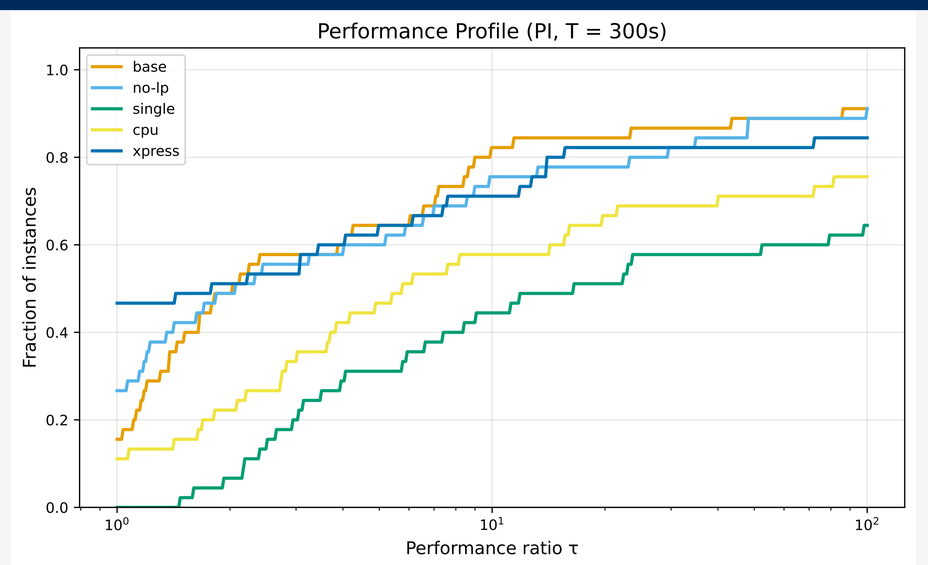
	GPU-MIP	Gurobi	Xpress	CPU
Feasible	42 / 50	**44 / 50**	39 / 50	40 / 50
Best obj. at T	16	**24**	22	4
Geomean PI (%)	**20.5**	—	19.3	68.8
Mean t_{first} (s)	19.5	—	28.6	**1.6**

Gurobi: final objectives only (no trajectory); "—" = N/A.

Highlights:

- Gurobi: most feasible (**44/50**), best objectives on 24 instances
- GPU-MIP: **3.4**× better PI than CPU; comparable to Xpress
- GPU-MIP uniquely feasible on 2 instances where Gurobi fails

Performance Profile



Performance profiles [2] (T=300 s). Higher = better. GPU-MIP (full) leads at all performance ratios.

Ablation Study

Variant	Feasible	W / L / T	vs. full PI ratio
No LP warm-start	42 / 50	20 / 22 / 8	0.85
Single solver	33 / 50	1 / 41 / 8	0.31

Portfolio is critical: 142 → 1 solver drops feasibility from 42 to 33 and degrades PI by ≈3×.

Constraint Learning

Two per-constraint signals persist across restarts:

Difficulty d_c (EMA, $\gamma = 0.999$):

$$d_c \leftarrow \gamma \cdot d_c + (1-\gamma) \cdot \mathbf{1}[c \text{ changed satisfaction state}]$$

Biases constraint sampling: violated constraints with higher d_c are sampled more frequently during Repair. Seeds restart weights.

Fragility f_c (EMA of inverse normalized slack):

$$f_c \leftarrow \beta \cdot f_c + (1-\beta) \cdot \frac{1}{1 + s_c/s_{\max}}$$

Tiebreaks lift moves: prefer moves that don't consume slack on structurally fragile constraints.

Symmetry: Color refinement (1-WL) detects variable orbits. Group A uses orbit operator (2 candidates per orbit); Group B adds ordering constraints $x_{i_1} \geq \cdots \geq x_{i_s}$.

Conclusions

- **GPU-native local search** matches commercial solvers on primal heuristic quality (300 s budget)
- **Cooperative portfolio** is the key enabler — parallel diversity, not raw speed
- **Half-credit scoring + CLAX** adapt SAT local search to linear constraints
- **On-GPU LP** (cuPDLPx [5]) keeps entire pipeline on accelerator
- Gurobi (44/50) sets the bar; GPU-MIP (42/50) is competitive and complementary

[1] Akif Çördük, Piotr Sielski, Alice Boucher, and Kumar Aatish. GPU-accelerated primal heuristics for mixed integer programming, 2025.
[2] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.
[3] Abdelraouf Ishtaiwi, John Thornton, Abdul Sattar, and Duc Nghia Pham. Neighbourhood clause weight redistribution in local search for SAT. In *CP 2005*, pages 772–776. Springer, 2005.
[4] Peng Lin, Mengchuan Zou, and Shaowei Cai. An Efficient Local Search Solver for Mixed Integer Programming. In *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, pages 19:1–19:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
[5] Haihao Lu, Zedong Peng, and Jinwen Yang. cuPDLPx: A further enhanced GPU-based first-order solver for linear programming. *arXiv preprint arXiv:2507.14051*, 2025.