# Learning-Based Algorithms for Solving Combinatorial Optimization


## Zaiwen Wen

**Beijing International Center For Mathematical Research**
**Peking University**

**Thanks: Tianyou Li, Haijun Zou, Ruisong Zhou, Yichen Pan, Zichen Wang, Chenyi Li**

**Greater Bay University**

# The Bitter Lesson (March 13, 2019)

**Richard Sutton**
**2024 Turing Award**

**The biggest lesson from 70 years of AI research**

- **General methods** that **leverage computation** are ultimately the most effective, and by a large margin

**Historical Evidence**

- Games: expert rules and strategies → surpassed by search and learning
- Speech/vision: feature engineering → end-to-end deep learning
- Control/robotics: human modeling intuition → reinforcement learning and general optimization

**One thing that should be learned**

- The great power of general purpose methods, of methods that continue to **scale with increased computation** even as the available computation becomes very great.
- The two methods that seem to **scale arbitrarily** in this way are **search** and **learning**

# AI for Optimization

## Deep Learning Model Free Method

- Learn GD by GD
- Intuitive, easy to implement
- No guarantee of generalization

UNIVERSITY OF OXFORD

DeepMind

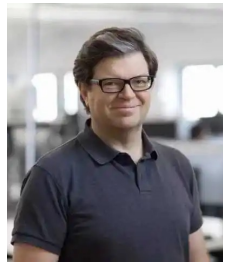
CIFAR

## Core Goal

- Representation
- Different sizes and types
- Decision-making and Reasoning

## Learn to Optimize

- Algorithm Unrolling
- High parameter efficiency
- Dependence on classical algorithms

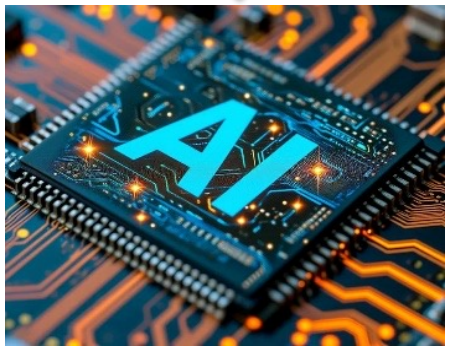


Turing Award
LeCun

## End-to End Learning methods

- point-network
- High efficiency
- No generalization to large problem

DeepMind
Oriol Vinyals




## Foundational Model for Optimization

## Reinforcement Learning

- Learn branch and cutting
- Thereotical guarantee
- Dependence on award function

Farkas Prize
Andrea Lodi

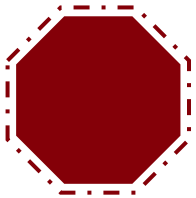
# Foundational Model for Optimization?

- **Traditional methods:** $\min_{x \in X} f(x)$
- **My foundational model (Generative paradigm):**

$$\min_{\theta} \mathbb{E}_{f \sim \mathcal{D}} [\mathbb{E}_{x \sim p_\theta, x \in X} [f(x)]]$$

  - The variable $x$ is sampled from a parameterized distribution $p_\theta$
  - The function $f$ is sampled from different problem families $\mathcal{D}$
  - Methodology: Offline training + online tuning

- **Examples:**
  - Binary Optimization (MP) https://github.com/optsuite/MCPG
  - Routing: Lmask (ICLR) https://github.com/optsuite/LMask
  - Scheduling: WeCAN https://github.com/optsuite/WeCAN
  - QAP: PLMA https://github.com/optsuite/PLMA



# Binary Optimization MCPG

C. Chen, R. Chen, T. Li, R. Ao, Z. **Wen**, A Monte Carlo Policy Gradient Method with Local Search for Binary Optimization, Mathematical Programming
https://github.com/optsuite/MCPG

# Binary Optimization

# MCPG

**Binary optimization:**

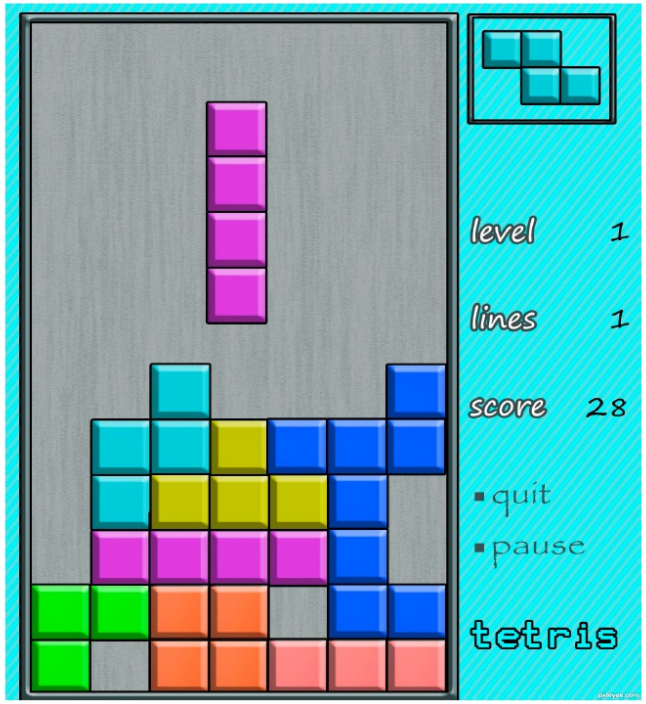
$$\min\ f(x)$$
$$\text{s.t. } x \in \{-1,1\}^n$$

**Probabilistic model:**

$$\min_{\theta}\ \mathbb{E}_{x\sim p_\theta}\left[f(x)\right]$$

- NP-hard Problem
- Difficult for current algorithms
- Industry demand for large-scale problems

**Tetris**




$$\max_{\pi}\ \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right],\ \text{with } s_0\sim\rho_0, a_t\sim\pi(\cdot|s_t), s_{t+1}\sim P(\cdot|s_t,a_t).$$

**Probabilistic Representation**

**Exact Penalty for Constraints**

Probabilistic Model → MCMC Sampling → Compute Obj. fun.

- Build parameterized models
- GPU for fast large-scale sampling
- Decompose computation into matrix operations

Policy gradient

**Efficient Large-Scale Exploration of Solution Space**

**Local Search Improves Solution Quality**

# Probabilistic Method

# MCPG

**Probabilistic Representation for Integer Programming Optimization**

**Target Distribution**

$$q^*(x) = \frac{1}{|\mathcal{X}^*|}\mathbf{1}_{\mathcal{X}^*}(x) = \begin{cases} \frac{1}{|\mathcal{X}^*|}, & x \in \mathcal{X}^*, \\ 0, & x \notin \mathcal{X}^*. \end{cases}$$

**Gibbs Distribution**

$q_\lambda(x) = \frac{1}{Z_\lambda}\exp\left(-\frac{f(x)}{\lambda}\right)$ 其中 $Z_\lambda = \sum_{x\in\mathcal{B}_n}\exp\left(-\frac{f(x)}{\lambda}\right)$

**Parameterized Distribution**

$p_\theta(x)$ Parameterized Distribution...

**Gibbs approx. converges**

**Parameterized Approximation**

**Approximating Distributions to Reduce Complexity**

When $\lambda \to 0$,

$$q_\lambda(x) = \frac{1}{Z_\lambda}\exp\left(-\frac{f(x)}{\lambda}\right) \to \frac{1}{|\mathcal{X}^*|}\mathbf{1}_{\mathcal{X}^*}(x) = q^*(x), \quad x \in \mathcal{B}_n$$

Optimize parameters θ to minimize the KL difference

$$\begin{aligned} \mathrm{KL}\left(p_\theta \,\|\, q_\lambda\right) &= \frac{1}{\lambda}\sum_{x\in\mathcal{B}_n} p_\theta(x) f(x) + \sum_{x\in\mathcal{B}_n} p_\theta(x)\log p_\theta(x) + \log Z_\lambda \\ &= \frac{1}{\lambda}\left(\mathbb{E}_{p_\theta}\left[f(x)\right] + \lambda\mathbb{E}_{p_\theta}\left[\log p_\theta(x)\right]\right) + \log Z_\lambda. \end{aligned}$$

$Z_\lambda$ ***is a constant***

**Training**

**Probabilistic Model Loss Function**

$$\min_\theta \quad L_\lambda(\theta) = \mathbb{E}_{p_\theta}\left[f(x)\right] + \lambda\mathbb{E}_{p_\theta}\left[\log p_\theta(x)\right]$$

**Gradient of Loss**

$$\nabla_\theta L_\lambda(\theta) = \mathbb{E}_{p_\theta}\left[A_\lambda(x;\theta,c)\nabla_\theta \log p_\theta(x)\right]$$

**similar to policy gradient in RL !**

$A_\lambda$ is advantage function

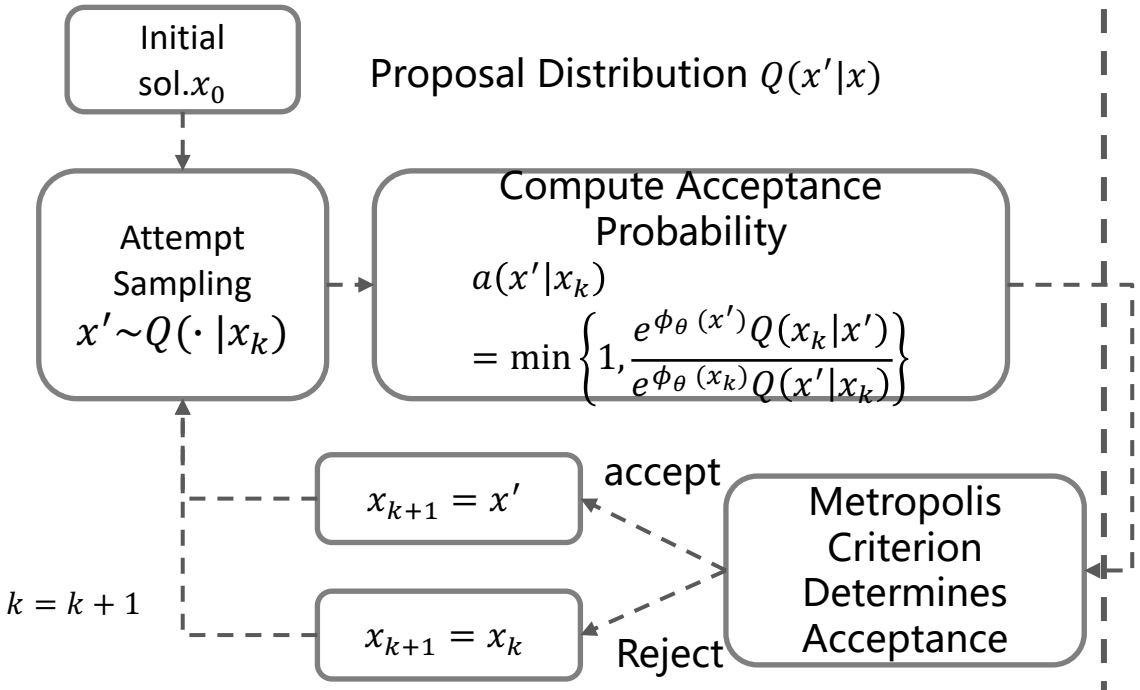
$$A_\lambda(x;\theta,c) := f(x) + \lambda\log p_\theta(x) - c$$

$c$ constant, i.e. $c = \mathbb{E}_{p_\theta}[f(x)]$

# Sampling and Local Search

# MCPG

## Markov Chain Monte Carlo (MCMC) Sampling




- Balance Exploration and Exploitation
- Only requires unnormalized probability distribution
- Large-Scale Parallelism on GPU
- Mean-Field Approximation Improves Sampling Efficiency

## Filter function for local search

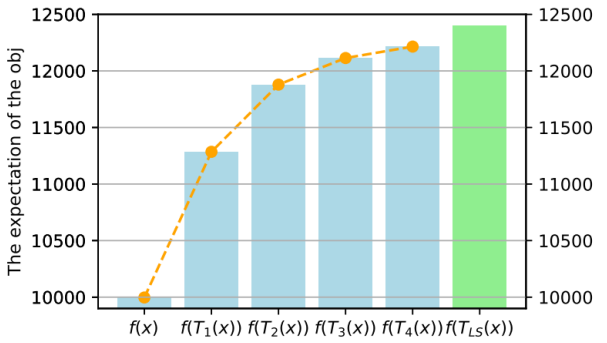
- Let $\mathcal{N}(x)$ be neighborhood of x, define

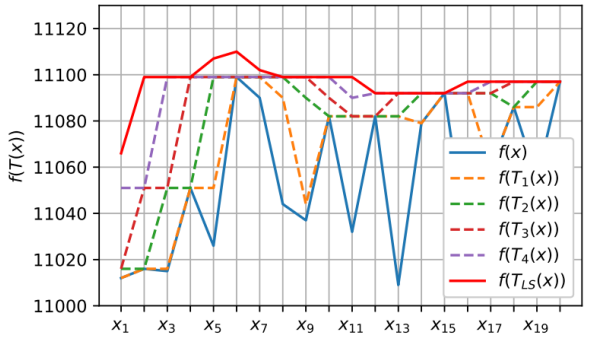
$$T(x) \in \arg\min_{\hat{x} \in \mathcal{N}(x)} f(\hat{x})$$

- Example 1: neighborhood search

$$T_k(x) = \arg\min_{\|\hat{x}-x\|_1 \le 2k} f(\hat{x}), \quad \mathcal{N}(x) = \{\hat{x} \mid \|\hat{x} - x\|_1 \le 2k\}$$

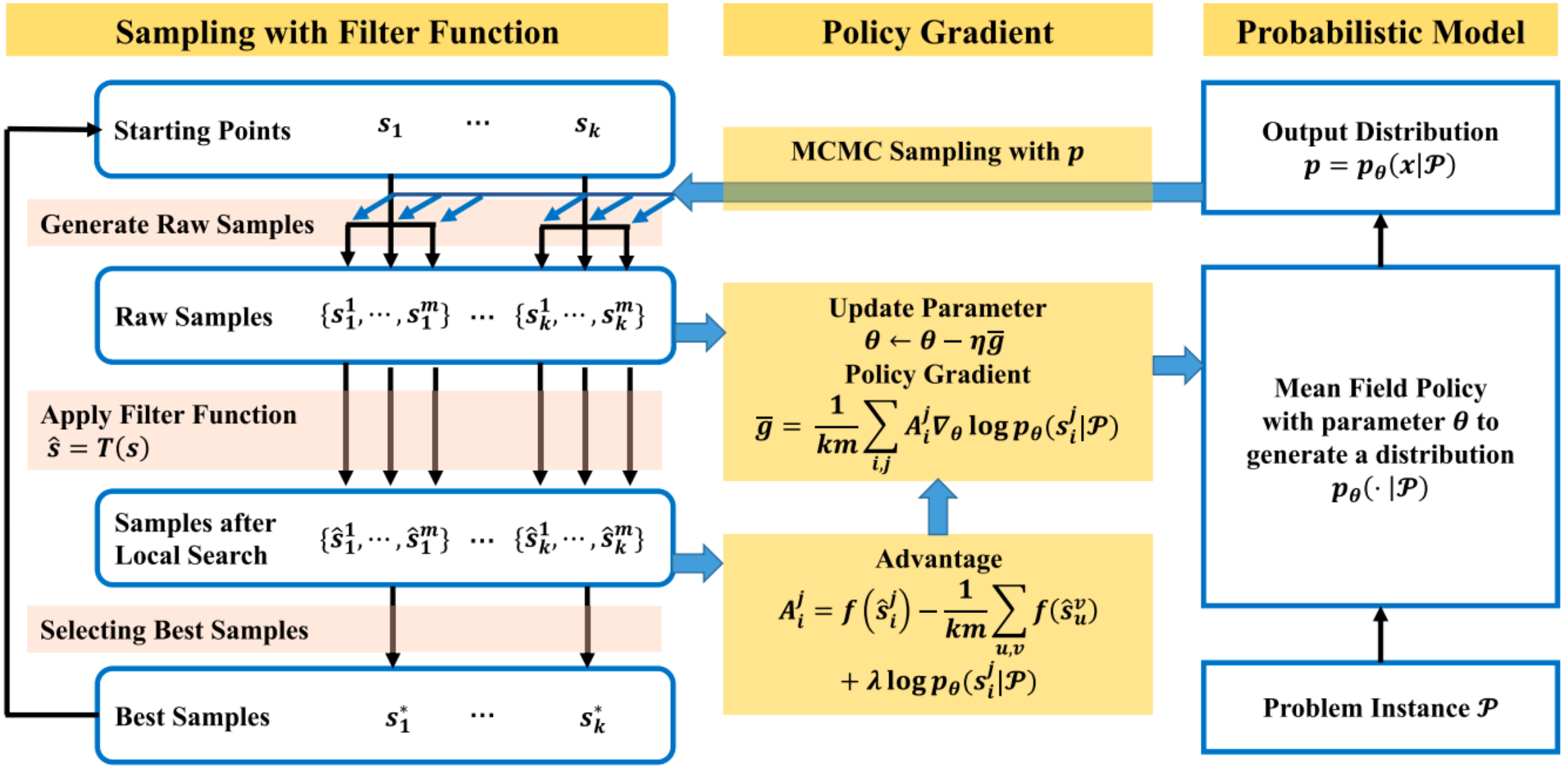
- Example 2: Greedy method $T_{LS}(x) = \texttt{LocalSearch}_f(x)$
- Filter function $T$ projects $x$ to a better solution in the neighborhood



# MCPG

MCPG




Github: https://github.com/optsuite/MCPG

# Convergence

## MCPG

**Assumption**: Let $\phi(x;\theta) = \log p_\theta(x|\mathcal{P})$. There exists some constants $M_1, M_2, M_3 > 0$ such that, for any $x \in \mathcal{B}_n$,

1. $\sup_{\theta\in\mathbb{R}^d} |\phi(x;\theta)| \leq M_1$,
2. $\sup_{\theta\in\mathbb{R}^d} \|\nabla_\theta \phi(x;\theta)\| \leq M_2$,
3. $\|\nabla_{\theta_1}\phi(x;\theta) - \nabla_{\theta_2}\phi(x;\theta)\| \leq M_3 \|\theta_1 - \theta_2\|, \forall \theta_1, \theta_2, \in \mathbb{R}^d$.

**Theorem**

*Let the assumption holds and $\{\theta_t\}$ be generated by MCPG. If the stepsize is chosen as $\eta^t = \frac{c\sqrt{mk}}{\sqrt{t}}$ with $c \leq \frac{1}{2l}$, then we have*

$$\min_{1\leq t\leq \tau} \mathbb{E}\left[\left\|\nabla_\theta L_\lambda(\theta^t)\right\|^2\right] \leq O\left(\frac{\log\tau}{\sqrt{mk\tau}} + \frac{1}{m^2}\right).$$

# Maxcut Problem

# MCPG

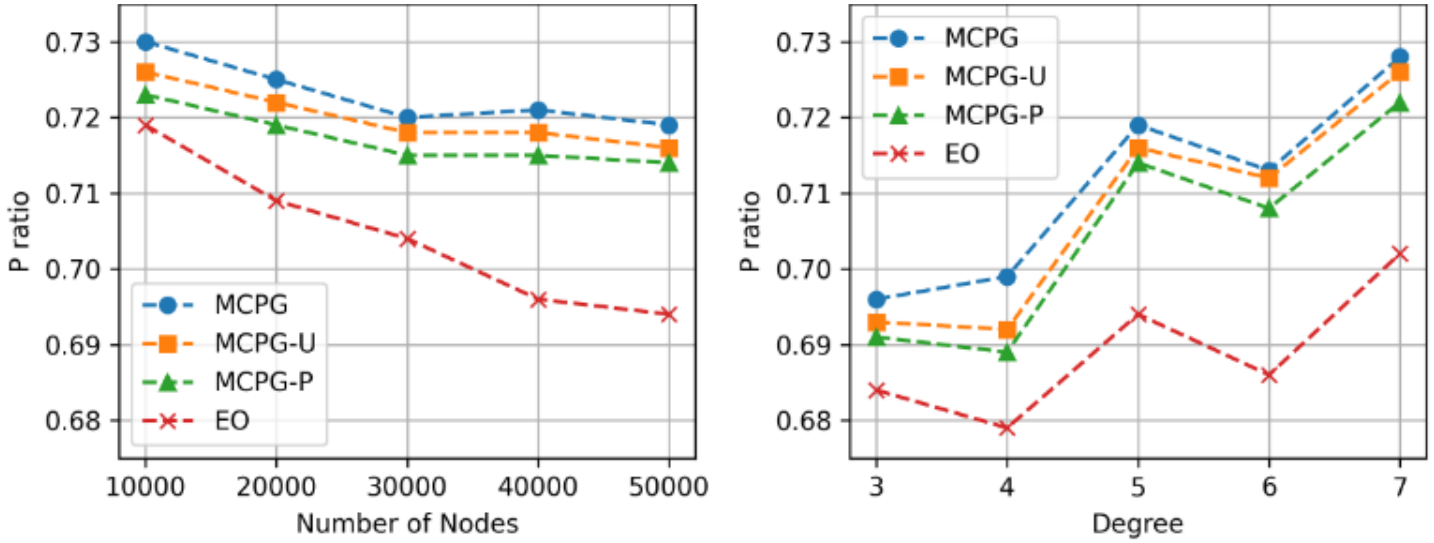
- **Formulation:**

$$\max \quad \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j),$$
$$\text{s.t.} \quad x \in \{-1, 1\}^n.$$

**MCPG: stable & strong on large-scale problems**

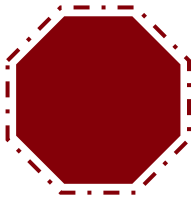
- **50k-node graphs ≈ 10k-node performance**
- **Best results on Gset; surpasses known solutions on G55 & G70**
- **Near-optimal in short time**
- **More complex filters → further gains**

- **Large scale random graphs**




- **Gset Problems**

| Graph | Nodes | Edges | MCPG | BLS | DSDP | RUN-CSP | PI-GNN | EO | EMADM |
|---|---|---|---|---|---|---|---|---|---|
| G14 | 800 | 4,694 | **3,064** | **3,064** | 2,922 | 2,943 | 3,026 | 3047 | 3045 |
| G15 | 800 | 4,661 | **3,050** | **3,050** | 2,938 | 2,928 | 2,990 | 3028 | 3034 |
| G22 | 2,000 | 19,990 | **13,359** | **13,359** | 12,960 | 13,028 | 13,181 | 13215 | 13297 |
| G49 | 3,000 | 6,000 | **6,000** | **6,000** | **6,000** | **6,000** | 5,918 | **6000** | **6000** |
| G50 | 3,000 | 6,000 | **5,880** | **5,880** | **5,880** | **5,880** | 5,820 | 5878 | 5870 |
| G55 | 5,000 | 12,468 | **10,296** | 10,294 | 9,960 | 10,116 | 10,138 | 10107 | 10208 |
| G70 | 10,000 | 9,999 | **9595** | 9,541 | 9,456 | - | 9,421 | 8513 | 9557 |



# Routing: LMask

T. Li, H. Zou, J. Wu, Z. **Wen**, LMask: Learn to Solve Constrained Routing Problems with Lazy Masking, ICLR 2026
https://github.com/optsuite/LMask

# Learn to Solve Constrained Routing

# LMask

NP-hard integer programming

Hard constraints make it difficult to guarantee solution feasibility

demand for fast path planning algorithms

## Routing Problem:

$$\min_{\pi \in \Pi} \quad f(\pi; \mathcal{P})$$
$$\text{s.t.} \quad c(\pi; \mathcal{P}) \leq 0,$$
$$d(\pi; \mathcal{P}) = 0,$$

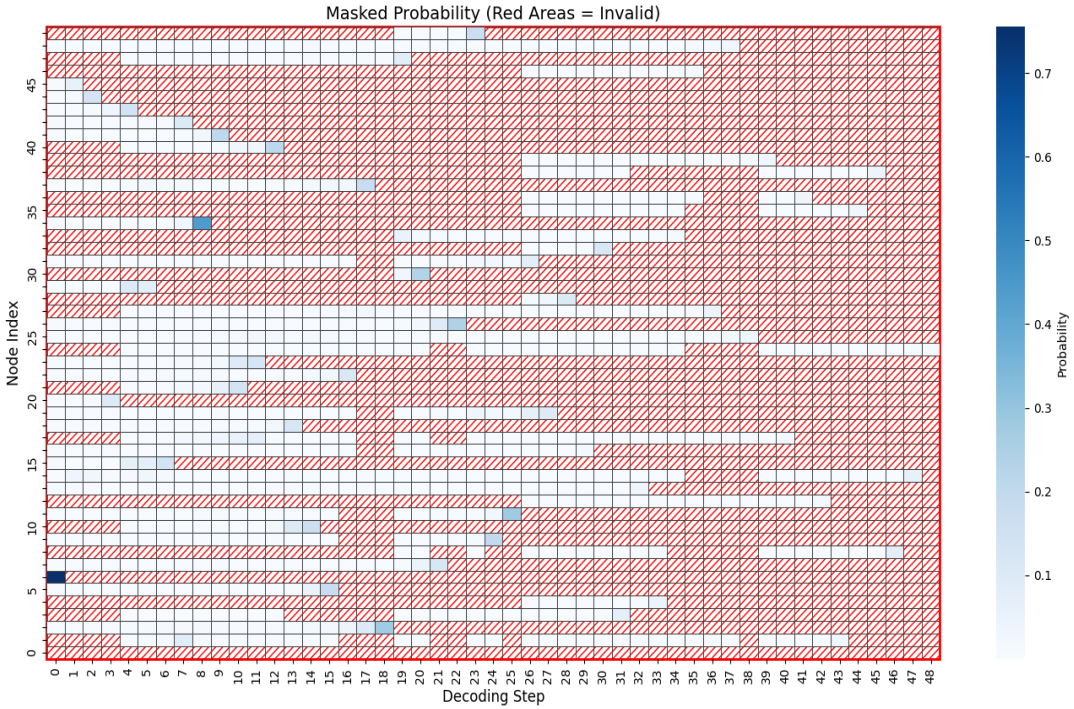
## Probabilistic Model:

$$L(\theta; \mathcal{P}) := \mathbb{E}_{p_\theta(\cdot;\mathcal{P})}[f(\pi; \mathcal{P})] + \lambda \mathbb{E}_{p_\theta(\cdot;\mathcal{P})}[\log p_\theta(\pi; \mathcal{P})]$$

$$p_\theta(\pi_t | \pi_{1:t-1}; \mathcal{P}) = \frac{e^{\phi_\theta(\pi_t | \pi_{1:t-1}; \mathcal{P})} M(\pi_t | \pi_{1:t-1}; \mathcal{P})}{\sum_{k=0}^{n} e^{\phi_\theta(k | \pi_{1:t-1}; \mathcal{P})} M(k | \pi_{1:t-1}; \mathcal{P})}$$

- **Integer programming: high complexity, low efficiency**
- **Learning-based methods with masking may violate constraints**

**M A S K**




### TSP with Time Window (TSPTW)

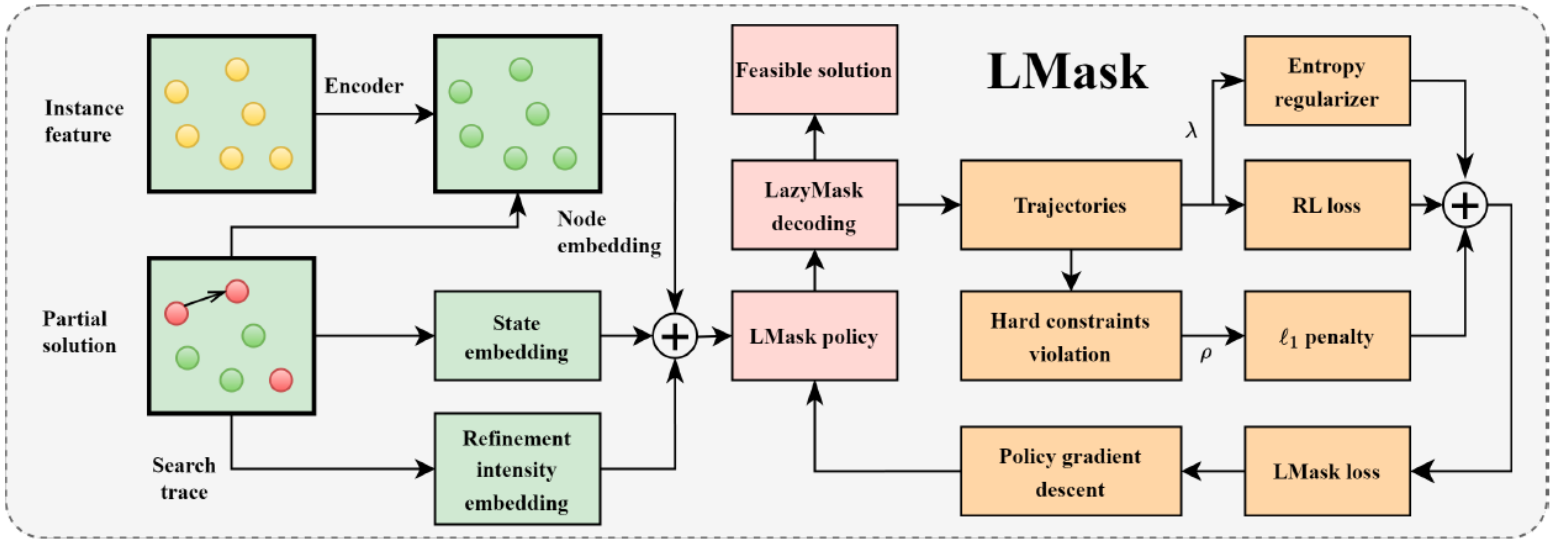
$$\min \quad f(\pi; \mathcal{P}) = \sum_{i=1}^{n} \|x_{\pi_i} - x_{\pi_{i+1}}\|$$
$$\text{s.t.} \quad c_i(\pi; \mathcal{P}) = \sum_{t=1}^{i} t_{\pi_t, \pi_{t+1}} - l_i \leq 0, \quad i = 1, \ldots, n,$$
$$c_{n+i}(\pi; \mathcal{P}) = e_i - \sum_{t=1}^{i} t_{\pi_t, \pi_{t+1}} \leq 0, \quad i = 1, \ldots, n,$$
$$d_i(\pi; \mathcal{P}) = \sum_{t=1}^{n} \mathbb{1}_{\pi_t = i} - 1 = 0, \quad i = 1, \ldots, n.$$

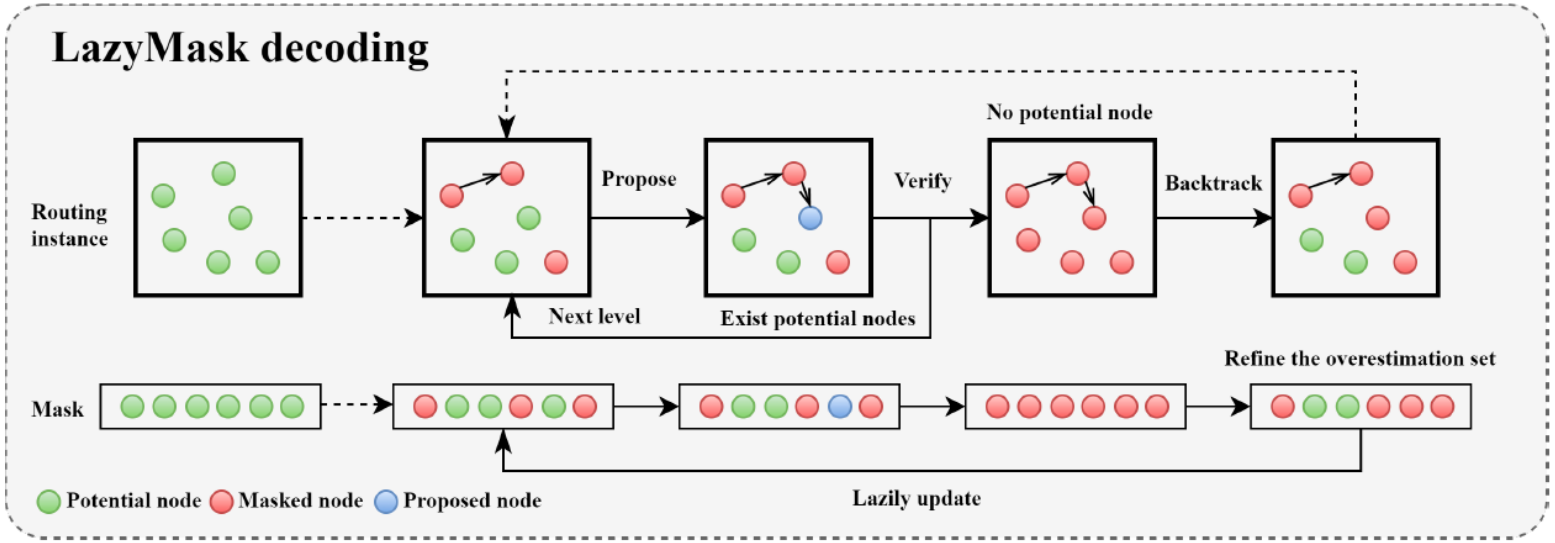
### TSP with Draft Limits (TSPDL)

$$\min \quad f(\pi; \mathcal{P}) = \sum_{i=1}^{n} \|x_{\pi_i} - x_{\pi_{i+1}}\|$$
$$\text{s.t.} \quad c_i(\pi; \mathcal{P}) = \sum_{t=1}^{i} q_{\pi_t} - D_i \leq 0, \quad i = 1, \ldots, n,$$
$$d_i(\pi; \mathcal{P}) = \sum_{t=1}^{n} \mathbb{1}_{\pi_t = i} - 1 = 0, \quad i = 1, \ldots, n.$$

Tianyou Li*, Haijun Zou*, Jiayuan Wu, Zaiwen Wen, "*LMask: Learn to Solve Constrained Routing Problems with Lazy Masking*", ICLR, 2026

# An illustrative overview of LMask



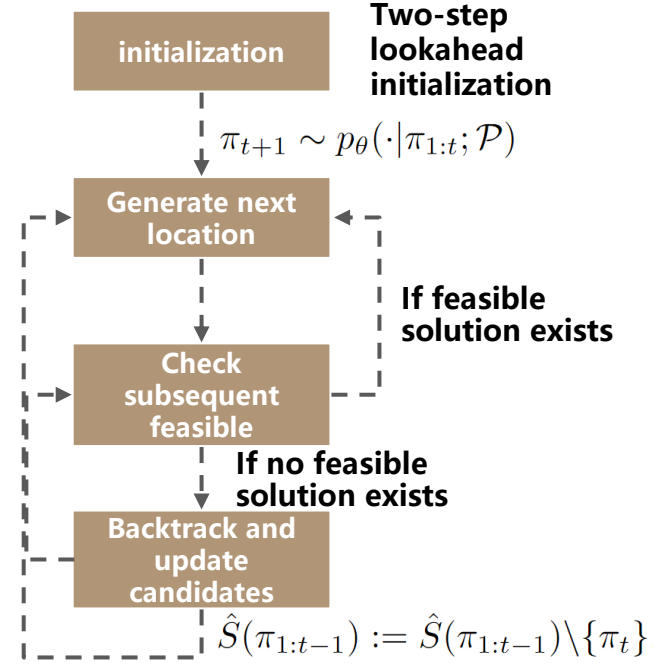




## Loss Function of Probabilistic Model

Path cost loss | Entropy regularization | Constraint violation penalty

$$\min \quad \mathbb{E}_{\pi\sim p_\theta(\cdot;\mathcal{P})}\left[f(\pi;\mathcal{P})+\rho\sum_{j=1}^{J}\max\left(c_j(\pi;\mathcal{P}),0\right)+\lambda\log p_\theta(\pi;\mathcal{P})\right]$$

## LazyMask Decoding



# Numerical Results

# LMask

- **Traditional Solvers**
  PyVRP，LKH，ORTOOL
- **Greedy Methods**
  Greedy-L，Greedy-C
- **Learning Methods**
  PIP, PIP-D, LMask

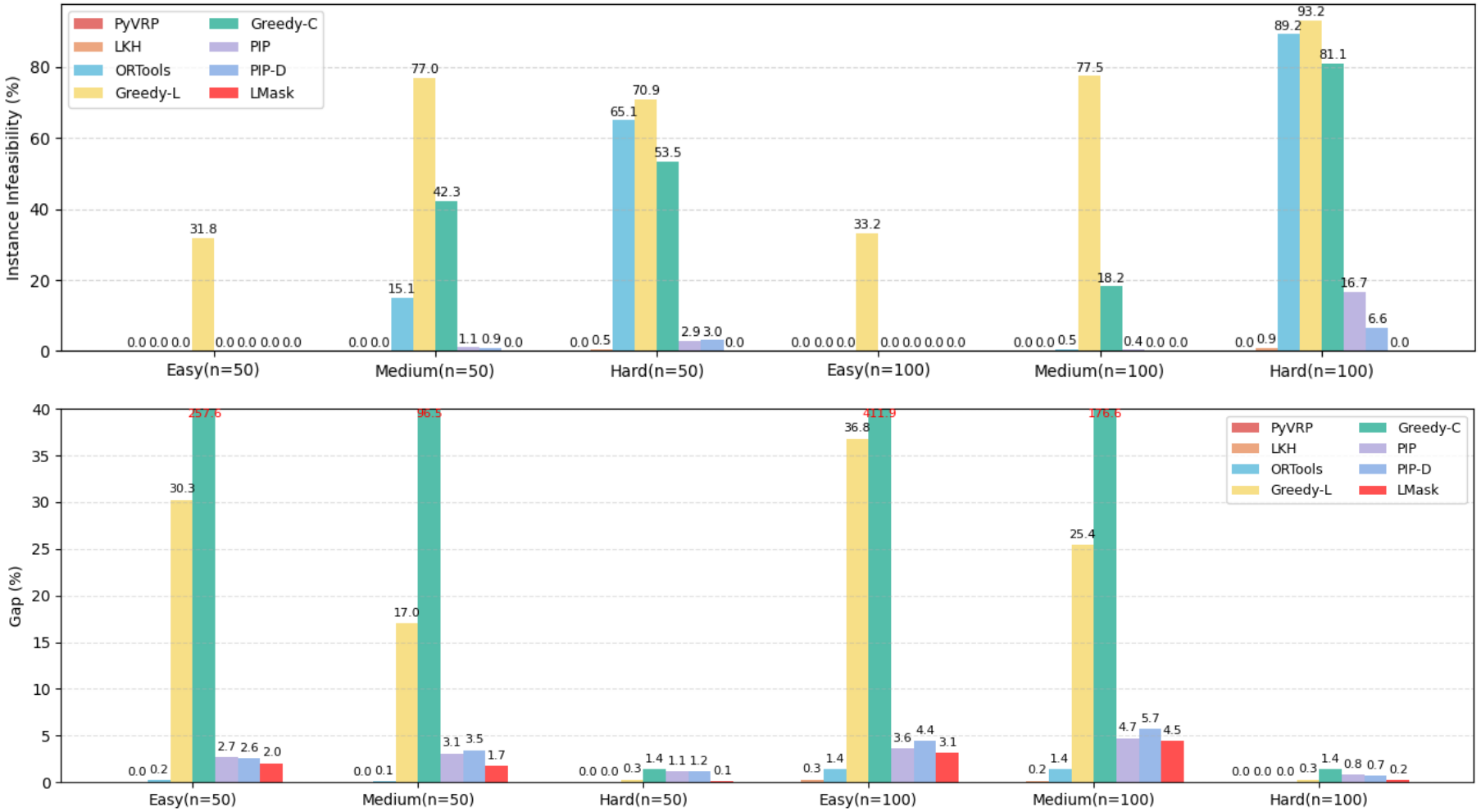
**Achieves feasible solutions for TSPTW**

- LMask infeasibility rate ≈0.0%

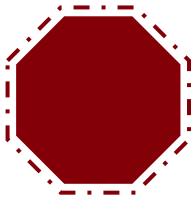
**Lowest optimality gap among learning methods**

- Lmask Significantly reduces optimality gap

**Much faster than traditional solvers**




- Traditional solvers (PyVRP/ORTools): >4h for 10k TSPTW instances (size 100)
- LMask (learning-based): 20s for 10k instances, outperforms others (>30s)



# Scheduling: WeCAN

R. Zhou, H. Zou, L. Zhou, C. Sun, Z. **Wen**, Reinforcement Learning for Heterogeneous DAG Scheduling with Weighted Cross-Attention

# DAG Heterogeneous Scheduling Problems

## Sequential Modelling

$$\min_{x=(s,c)} \quad f(x) := \max_{v \in V}[s(v) + t(v)/K_{acc}(v, c(v))],$$

$$\text{s.t.} \quad s(v) + t(v)/K_{acc}(v, c(v)) \le s(w),\ \forall (v, w) \in E, \qquad \text{Dependency}$$

$$\sum_{v \in F(t,c)} \rho(v) \le \lambda(c),\ \forall t \ge 0,\ c \in C, \qquad \text{Resource}$$

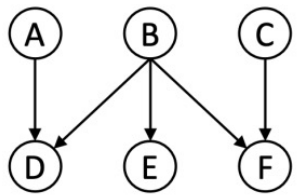
$$K_{acc}(v, c(v)) > 0,\ \forall v \in V, \qquad \text{Matching}$$

$$s(v) \ge 0,\ \forall v \in V.$$

$$F(t, c) := \{v \in V | s(v) \le t;\ s(v) + t(v)/K_{acc}(v, c(v)) > t;\ c(v) = c\}.$$

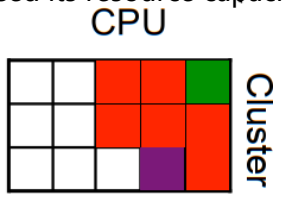
### Dependency

Task execution order must satisfy the directed graph dependencies



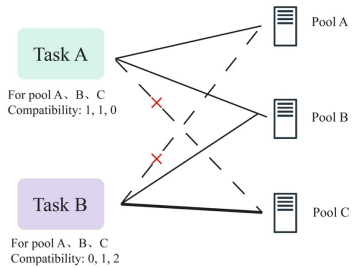

### Resource

At any time, the total resource consumption of running tasks on any resource pool must not exceed its resource capacity




### Matching

Each task must be assigned to a compatible resource pool




## Problem Scale (TPC-H-100, 3 pool)

| | Variables | Constraints |
|---|---|---|
| Sequential model | $2.0 \times 10^3$ | $5.0 \times 10^3 + \lvert E \rvert, \le 1.0 \times 10^6$ |
| MILP model | $6.0 \times 10^6$ | $1.2 \times 10^7$ |

## MILP Modelling

$$\min t_{max},$$

$$\text{s.t. } s_i + t_i \sum_k v_i^k a_{ik} \le t_{max}, \forall i \in V,$$

$$s_i + t_i \sum_{k \in C} v_i^k a_{ik} \le s_j, \forall (i,j) \in E, \qquad \text{Dependency}$$

$$s_i < s_j + C_1^{\text{up}} u_{ij}, \forall i, j \in V,$$

$$s_j \le s_i + C_1^{\text{up}}(1 - u_{ij}), \forall i, j \in V,$$

$$s_i < s_j + t_j \sum_{k \in C} v_j^k a_{jk} + C_1^{\text{up}}(1 - w_{ij}), \forall i, j \in V,$$

$$s_j + t_j \sum_{k \in C} v_j^k a_{jk} \le s_i + C_1^{\text{up}} w_{ij}, \forall i, j \in V,$$

$$u_{ij} + w_{ij} \le 1 + x_{ij}, \forall i, j \in V,$$

$$u_{ij} + w_{ij} \ge 2x_{ij}, \forall i, j \in V,$$

$$x_{ij} + v_j^k \ge 2y_{ijk}, \forall i, j \in V, k \in C,$$

$$x_{ij} + v_j^k \le 1 + y_{ijk}, \forall i, j \in V, k \in C,$$

$$\sum_k v_i^k = 1, \forall i \in V, k \in C,$$

Sequence Characterization

Ordering Representation

$$1 - v_i^k + C_0^{\text{up}} - a_{ik} > 0,\ \forall i \in V, k \in C, \qquad \text{Resource}$$

$$\rho_l^i + \sum_{j \ne i} y_{ijk} \rho_l^j \le C_1^{\text{up}}(1 - v_i^k) + \lambda_l^k, \forall i \in V, k \in C, l \in \{1, ..., r\}, \qquad \text{Matching}$$

$$s_i \ge 0, t_{max} \ge 0,\ u_{ij}, w_{ij}, x_{ij}, y_{ijk}, v_i^k \in \{0, 1\}, \forall i, j \in V, k \in C.$$

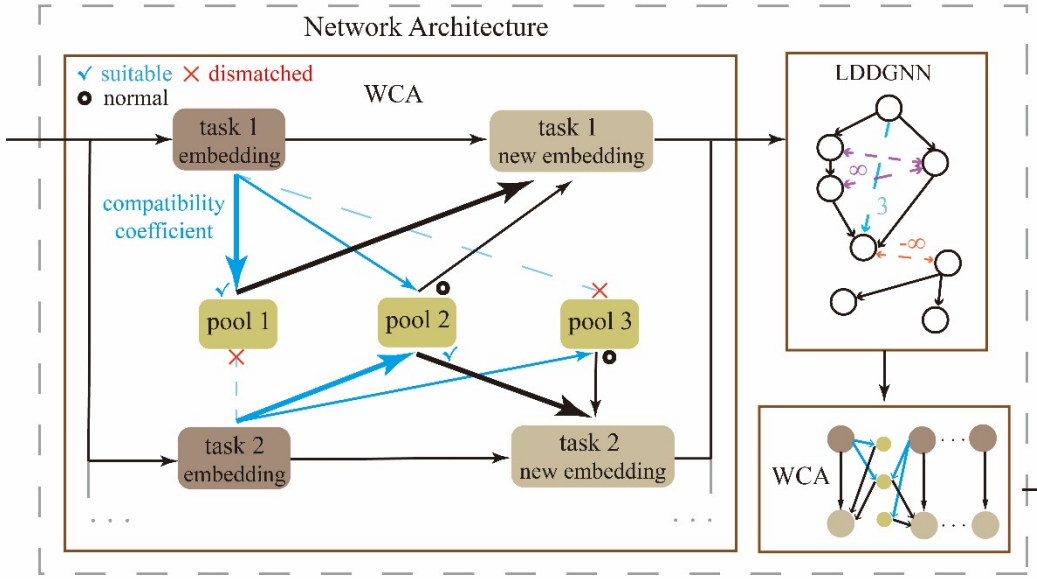
## Challenges in Problem Solving

**Feasible real-valued solutions are difficult to generate directly**

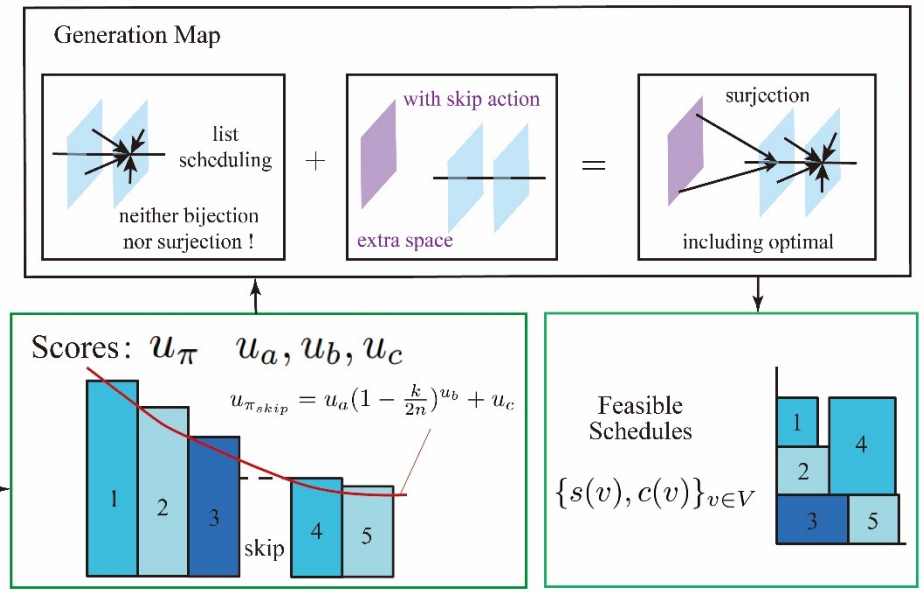
**Directed graph constraints are complex and diverse in form**

**The number of variables and constraints is extremely large**

# Network Architecture




Neural Network Architecture Modules




Generation Mapping with Skip Decision Incorporated

Scheduling $(G, V, C, \rho, t, \lambda, K_{acc})$

Task-Pool Pair Score $u_\pi$ Skip Coefficient $u_a, u_b, u_c$

Score and jumping coefficient → Solution $\{s(v), c(v)\}_{v \in V}$

- The Weighted Cross-Attention Module embeds all compatibility coefficients as attention biases, enabling lossless information encoding while ensuring scalability — a single network can be directly applied to various resource environments (e.g., different numbers of pools).
- The Graph Neural Network based on Longest Directed Distance efficiently encodes dependency relations, guaranteeing long-range information propagation while preserving partial order relationships.

Generation mapping: selecting the highest-scoring constraint-compliant decision based on static task–pool scores and dynamic skip scores. If a task–pool pair (v, c) is selected: assign start time s(v) = t and allocate pool c(v) = c. If "skip" is selected: advance t to the earliest next task completion time.

- The introduced skip decision fixes the flaw where list scheduling may miss the optimal solution, significantly improving solution accuracy in specific scenarios such as heavy tasks.
- The skip-score clustering method groups "poor solutions" together, reducing the difficulty of reinforcement learning.

# Numerical Results

WeCAN

## TPC-H Dataset

Table 1: Experimental results on TPC-H datasets with standard deviation among random seed.

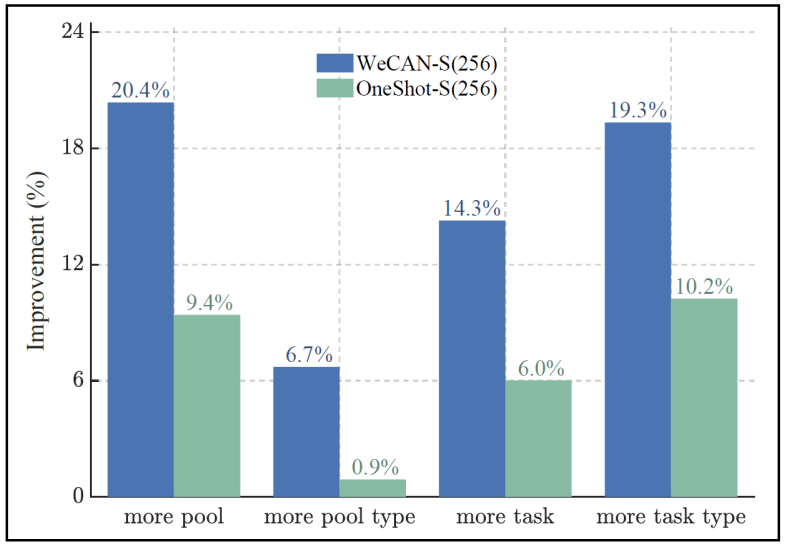
| | TPC-H-30, 3 pools | | TPC-H-50, 3 pools | | TPC-H-100, 3 pools | |
|---|---|---|---|---|---|---|
| | MakeSpan | Time | MakeSpan | Time | MakeSpan | Time |
| SFT | 27404 | 0.23 | 49172 | 0.78 | 84986 | 3.08 |
| MOPNR | 25052 | 0.30 | 43545 | 0.99 | 77362 | 3.34 |
| CP | 23869 | 0.29 | 41597 | 0.90 | 74364 | 3.35 |
| HEFT | 23177 | 0.18 | 39315 | 0.54 | 70137 | 1.86 |
| Tetris | 23170 | 0.21 | 38654 | 0.62 | 71296 | 2.13 |
| PPO-BiHyb | 21941 | 20.48 | 36333 | 55.74 | 67695 | 179.19 |
| One-Shot-S(256) | 20399 ± 181 | 2.26 | 35561 ± 108 | 4.16 | 66173 ± 180 | 9.85 |
| WeCAN-Greedy | 19578 | 0.15 | 33428 | 0.50 | 62587 | 1.72 |
| WeCAN-S(64) | 19053 ± 28 | 1.54 | 32912 ± 40 | 2.86 | 61662 ± 118 | 5.26 |
| WeCAN-S(256) | **18964 ± 10** | 2.43 | **32814 ± 47** | 4.39 | **61373 ± 28** | 10.43 |

## Synthetic Dataset

Table 2: Experimental results on Computation Graphs datasets with 500 tasks.

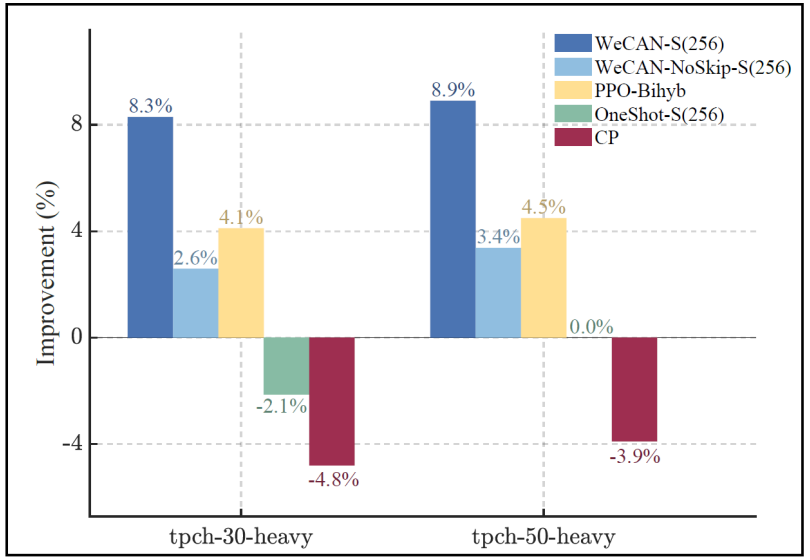
| | Erdős-Rényi | | Layer Graphs | | Stochastic Block | |
|---|---|---|---|---|---|---|
| | MakeSpan | Time | MakeSpan | Time | MakeSpan | Time |
| SFT | 13316.8 | 0.81 | 16158.4 | 0.34 | 14407.9 | 0.53 |
| MOPNR | 12771 | 1.07 | 14714 | 0.38 | 13148 | 0.68 |
| Tetris | 13084 | 0.52 | 14271 | 0.44 | 13666 | 0.64 |
| CP | 12457 | 1.08 | 14797 | 0.40 | 13388 | 0.74 |
| HEFT | 11098 | 0.55 | 12428 | 0.75 | 11260 | 0.57 |
| PPO-BiHyb | 10795 | 65.51 | 11883 | 73.7 | 10885 | 73.7 |
| One-Shot-S(256) | 11144 ± 29 | 4.45 | 12277 ± 49 | 3.83 | 11457 ± 64 | 4.00 |
| WeCAN-Greedy | 10270 | 0.57 | 11173 | 0.26 | 10539 | 0.41 |
| WeCAN-S(64) | 10115 ± 10 | 3.21 | 10862 ± 29 | 3.07 | 10074 ± 18 | 3.06 |
| WeCAN-S(256) | **10083 ± 13** | 4.94 | **10752 ± 27** | 4.30 | **10019 ± 12** | 4.58 |

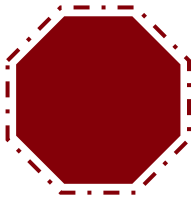
## Good generalization




## Skip decision

## Guaranteed optimal reachability





# QAP: PLMA

Pan Yicheng, Zou Haijun, Zhou Ruisong, Li Tianyou, **Wen Zaiwen**,
Learning the Quadratic Assignment Problem with Warm-Started MCMC Finetuning and Cross-Graph Attention

# Learning the Quadratic Assignment

## QAP: Permutation

$$\min_{\pi \in \Pi_n} f(\pi; \mathcal{P}) := \sum_{i=1}^{n} \sum_{j=1}^{n} F_{ij} D_{\pi(i)\pi(j)}.$$

## QAP: Matrix

$$\min_{X \in \{0,1\}^{n \times n}} \langle F, XDX^{\mathrm{T}} \rangle, \quad \text{s.t.} \quad X\mathbf{1} = \mathbf{1},\ X^{\mathrm{T}}\mathbf{1} = \mathbf{1}.$$

- Permutation incurs high space complexity
- conventional models suffer from inefficient sampling

- Feature extraction is challenging for the problem information matrices D and F

## Energy Models over Permutations:

$$p_\theta(\pi \mid \mathcal{P}) \propto \exp(\Phi_\theta(\pi)), \quad \Phi_\theta(\pi) = \sum_{i=1}^{n} \phi_{i,\pi(i)}.$$

## O(1) Temporal Neighborhood Search:

$$\exp(\Phi_\theta(\pi') - \Phi_\theta(\pi)) = \exp\left(\phi_{a,\pi(b)} + \phi_{b,\pi(a)} - \phi_{a,\pi(a)} - \phi_{b,\pi(b)}\right)$$

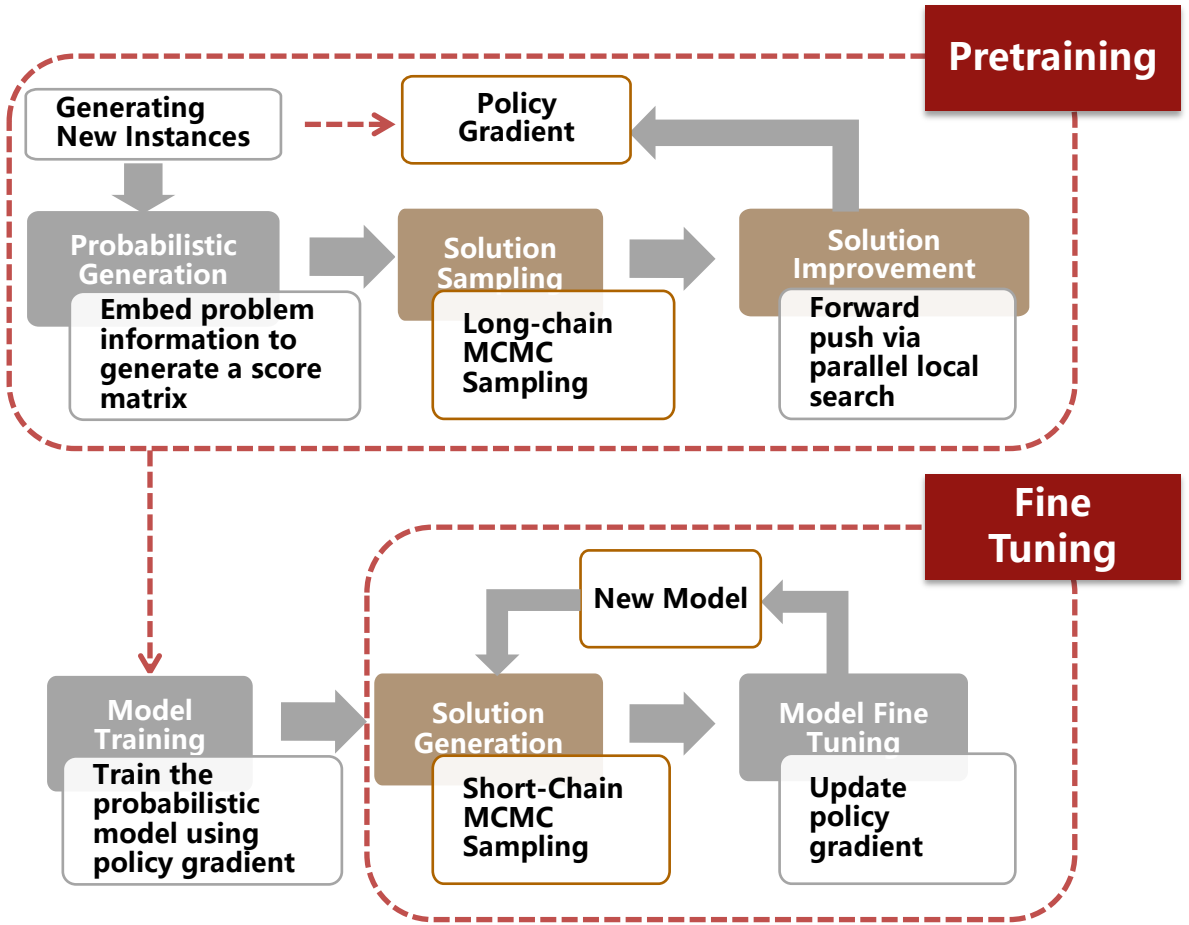
## Estimation of Gradient:

$$\nabla_\theta \mathbb{E}_{\pi \sim p_\theta}[g(\pi)] \approx \frac{1}{N-1} \sum_{i=1}^{N} \left( g(\pi_i) - \frac{1}{N} \sum_{i=1}^{N} g(\pi_i) \right) \nabla_\theta \Phi_\theta(\pi_i).$$

## Forward Training Objective:

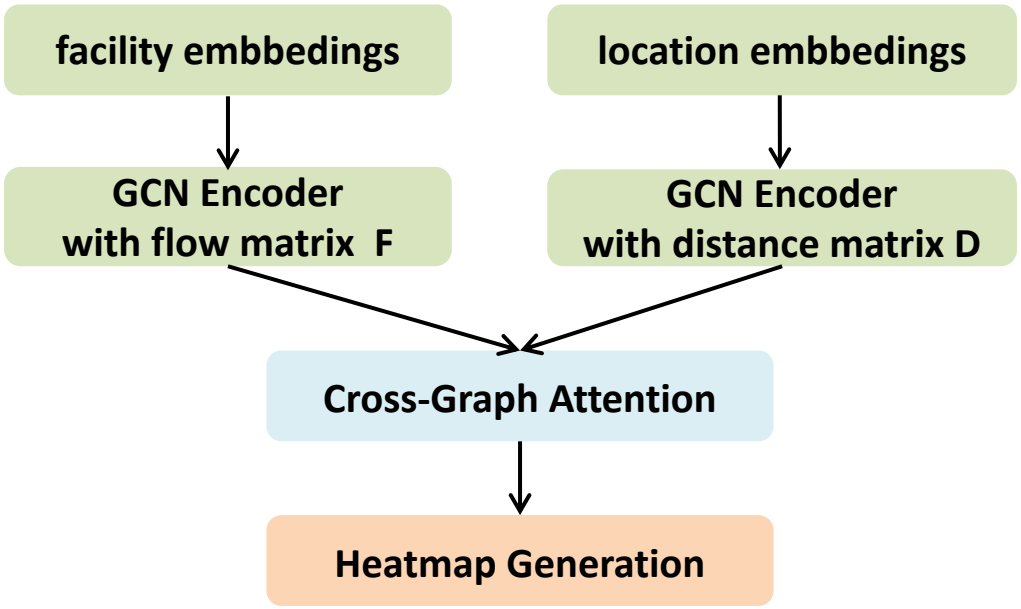
$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta) := \mathbb{E}_{\mathcal{P} \sim \Gamma} \mathbb{E}_{\sigma \sim \mathcal{T}_\sharp p_\theta(\cdot|\mathcal{P})}[f(\sigma; \mathcal{P})] = \mathbb{E}_{\mathcal{P} \sim \Gamma} \mathbb{E}_{\pi \sim p_\theta(\cdot|\mathcal{P})}[f(\mathcal{T}(\pi); \mathcal{P})].$$

## Pretraining and Fine tuning:



# Model Architecture

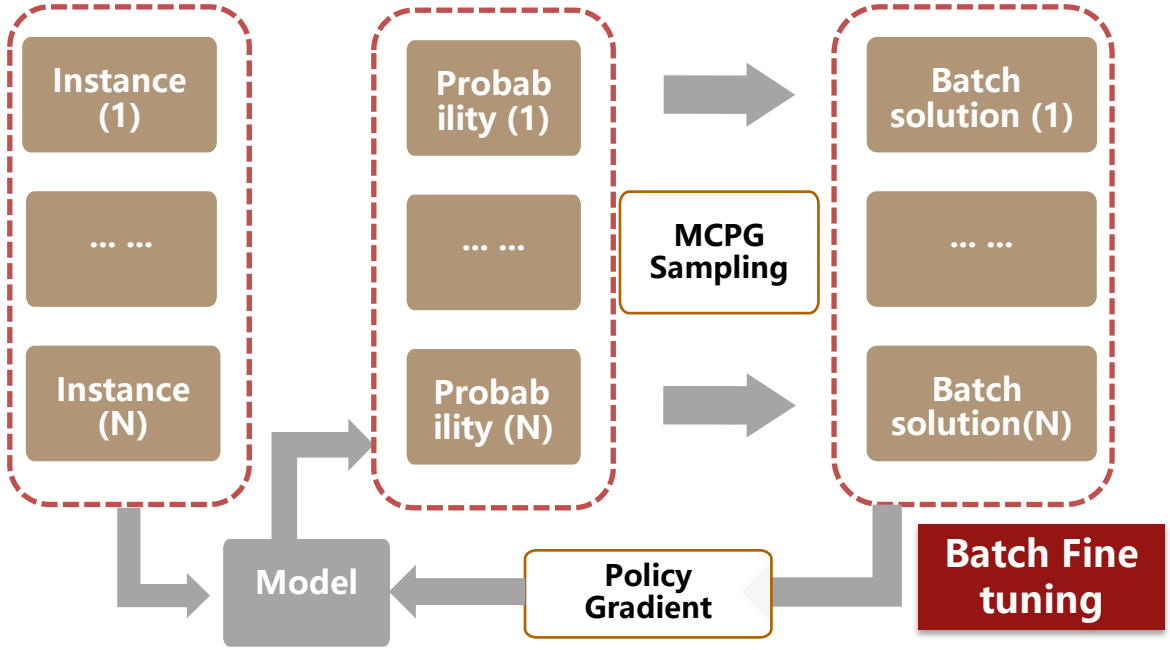
## Network Architecture




- **Information Embedding within Graphs:** Treat matrices D/F as weighted adjacency matrices for separate graph convolution, obtaining two sets of node features.
- **Information Interaction across Graphs:** Perform cross-attention on the two feature sets to enable cross-graph message passing.
- **Matching-Score Generation:** Apply inner product followed by nonlinear activation to the two feature sets.
- **Core Advantage:** The architecture avoids constructing an $n^2$-node correlation graph, ensuring scalability.

## Warm-start MCMC Fine Tuning

- **Native Feasibility:** Probability is modeled directly in the permutation space, using 2-swap as the fundamental improvement operation, ensuring both sampling feasibility and efficiency.
- **Efficient Parallelism:** During fine-tuning, multiple instances are optimized in parallel via batch processing, sharing the same set of network parameters, thereby improving computational efficiency.
- **Support for Pretraining:** Pretraining provides a favorable initial probability distribution for fine-tuning.



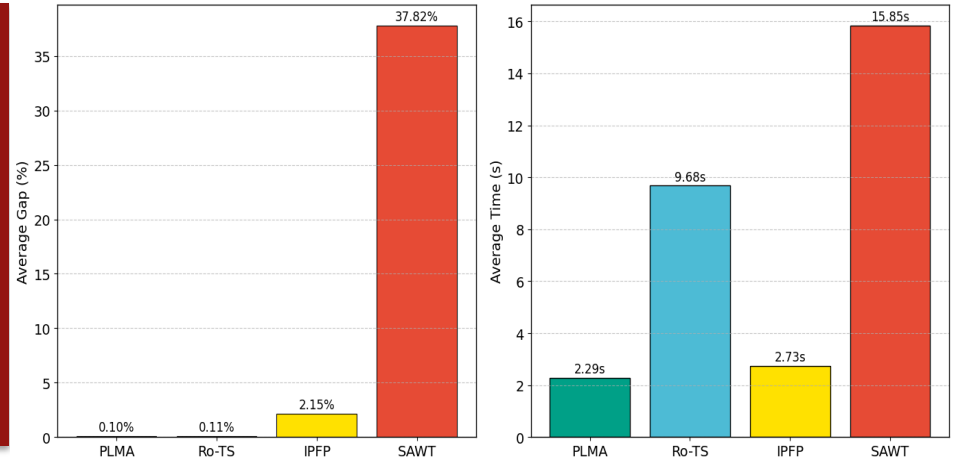
# Numerical Results

## Synthetic datasets

Table 1: Performance comparison on synthetic datasets (256 instances).

| Algorithm | QAP20 Cost | QAP20 Gap | QAP20 Time | QAP50 Cost | QAP50 Gap | QAP50 Time | QAP100 Cost | QAP100 Gap | QAP100 Time |
|---|---|---|---|---|---|---|---|---|---|
| *Geometrically Structured Datasets* | | | | | | | | | |
| Ro-TS (1k) | 54.38 | 0.01% | 17.04s | 375.99 | 0.14% | 4m35s | 1593.27 | 0.13% | 38m56s |
| Ro-TS (5k) | **54.37** | **0.00%** | 1m25s | **375.48** | **0.00%** | 22m53s | 1591.25 | 0.00% | 3h15m |
| IPFP | 55.11 | 1.37% | 2.04s | 378.76 | 0.88% | 11.47s | 1600.27 | 0.57% | 1m34s |
| IPFP (10) | 54.54 | 0.31% | 20.97s | 376.60 | 0.30% | 2m30s | 1594.76 | 0.22% | 17m34s |
| NGM | 62.93 | 15.87% | 24.78s | 429.69 | 14.46% | 1m16s | 1773.71 | 11.47% | 2m29s |
| SAWT (10k) | 54.72 | 0.64% | 3m41s | 380.92 | 1.45% | 5m36s | 1617.30 | 1.64% | 10m43s |
| **PLMA** ($T = 1$) | 54.63 | 0.48% | 0.06s | 379.79 | 1.15% | 0.41s | 1607.84 | 1.04% | 4.27s |
| **PLMA** ($T = 50$) | **54.37** | **0.00%** | 2.57s | 375.55 | 0.20% | 19.88s | 1591.73 | 0.03% | 3m30s |
| **PLMA** ($T = 200$) | **54.37** | **0.00%** | 9.36s | **375.48** | **0.00%** | 1m19s | **1591.23** | **0.00%** | 13m58s |
| *Uniformly Random Datasets* | | | | | | | | | |
| Ro-TS(1k) | 76.61 | 0.07% | 17.56s | 523.08 | 0.22% | 4m36s | 2195.98 | 0.13% | 38m59s |
| Ro-TS(5k) | **76.56** | **0.00%** | 1m27s | 521.91 | 0.00% | 22m59s | 2193.16 | 0.00% | 3h15m |
| IPFP | 79.13 | 3.39% | 2.12s | 530.74 | 1.69% | 7.45s | 2211.38 | 0.83% | 41.95s |
| IPFP(25) | 77.60 | 1.37% | 54.70s | 526.96 | 0.97% | 4m20s | 2203.29 | 0.46% | 19m20s |
| NGM | 88.34 | 15.49% | 25.08s | 594.99 | 14.01% | 1m17s | 2438.52 | 11.19% | 2m29s |
| **PLMA** ($T = 1$) | 78.23 | 2.20% | 0.06s | 538.42 | 3.17% | 0.40s | 2243.43 | 2.29% | 4.32s |
| **PLMA** ($T = 50$) | 76.59 | 0.05% | 2.47s | 523.95 | 0.40% | 19.51s | 2200.40 | 0.34% | 3m30s |
| **PLMA** ($T = 200$) | **76.56** | **0.00%** | 9.60s | **521.83** | **-0.01%** | 1m18s | **2193.13** | **0.00%** | 14m1s |

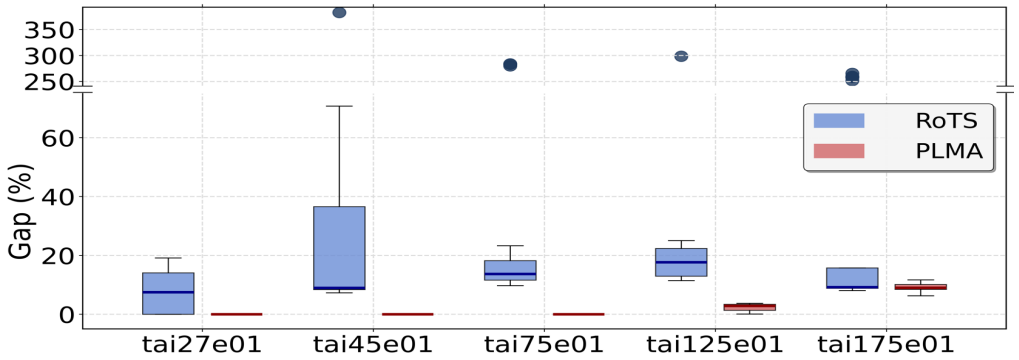
**QAP LIB**



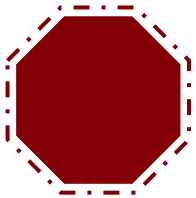

## Robustness Testing

Table 4: Grouped results on Taixxeyy instances.

| Class | Ro-TS mean | Ro-TS [min, max] | Ro-TS Time (s) | IPFP mean | IPFP [min, max] | IPFP Time (s) | PLMA mean | PLMA [min, max] | PLMA Time (s) |
|---|---|---|---|---|---|---|---|---|---|
| tai27e | 41.50% | [0.11%, 221.08%] | 0.57 | 19.47% | [6.14%, 34.11%] | 0.38 | **0.00%** | [0.00%, 0.00%] | 0.08 |
| tai45e | 101.89% | [1.00%, 400.60%] | 3.83 | 22.01% | [8.26%, 36.98%] | 1.03 | **0.03%** | [0.00%, 0.35%] | 0.28 |
| tai75e | 111.28% | [6.20%, 280.01%] | 18.49 | 27.64% | [17.56%, 36.78%] | 2.52 | **0.09%** | [0.00%, 0.45%] | 1.48 |
| tai125e | 82.53% | [7.65%, 265.54%] | 72.52 | 26.93% | [20.64%, 32.67%] | 8.52 | **2.67%** | [-0.08%, 5.62%] | 8.18 |
| tai175e | 67.86% | [9.11%, 260.98%] | 158.18 | 23.32% | [17.70%, 28.11%] | 16.86 | **9.11%** | [5.61%, 12.04%] | 14.63 |
| Average | 81.01% | [4.82%, 285.64%] | 50.72 | 23.87% | [14.06%, 33.73%] | 5.86 | **2.38%** | [1.11%, 3.69%] | 4.93 |




- **PLMA(T=1): Generates solutions in seconds with accuracy far surpassing previous learning-based methods.**
- **PLMA(T=200): Achieves 0.00% optimality gap on synthetic datasets, with significantly faster speed compared to traditional methods (14× faster on QAP100, 4× faster on QAPLIB).**
- **Robustness: No outlier cases with >100% optimality gap on Taixxeyy, and the average gap is substantially lower (2.38% vs. 81.01%).**



# Formalization + Automatic Theorem Proving

# Formalization of the Gradient Method

- minimize a differentiable fun: $f(x)$, i.e., $\min_x f(x)$
- the gradient method: $x_{k+1} = x_k - \alpha \nabla f(x_k)$

```
class GradientDescent_fix_stepsize (f : E → ℝ) (f' : E → E) (initial_x : E) :=
  (x : ℕ → E) (a : ℝ) (l : NNReal)
  (diff : ∀ x₁, HasGradientAt f (f' x₁) x₁) (smooth : LipschitzWith l f')
  (update : ∀ k : ℕ, x (k + 1) = x k - a • f' (x k))
  (hl : l > (0 : ℝ)) (step₁ : a > 0) (initial : x 0 = initial_x)
```

- Suppose $f(x)$ is convex, then the complexity is:

$$f(x_k) - f(x^*) \leq \frac{1}{2\alpha k}\|x_0 - x^*\|^2, \quad \forall k \in \mathbb{N}$$

```
theorem gradient_method_fix_stepsize {alg : GradientDescent_fix_stepsize f f' x₀}
    (hfun: ConvexOn ℝ Set.univ f) (step₂ : alg.a ≤ 1 / alg.l) :
    ∀ k : ℕ, f (alg.x (k + 1)) - f xm ≤ 1 / (2 * (k + 1) * alg.a) * ‖x₀ - xm‖ ^ 2 := by
```

# Formalization of Mathematical Optimization

**Building a Formalized Software Library for Optimization: Optlib**
**https://github.com/optsuite/optlib**

**Over 3,000 formalized theorems**
**Tens of thousands of lines of formalized code**

**Conceptual Extensions**

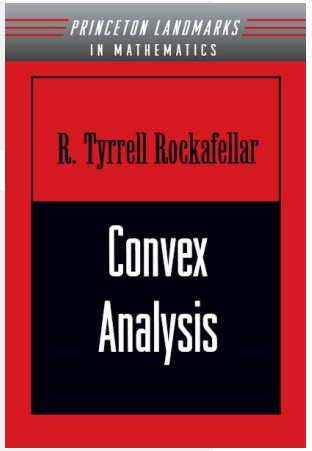
- Definitions and properties of gradients for smooth functions
- Definitions of subgradients and proximal operators for nonsmooth functions

**Property Extensions**

- Properties of convex functions and Lipschitz-smooth functions
- Optimality conditions for convex optimization and general constrained optimization problems

**Algorithm Definitions and Theoretical Proofs**

- Convergence analysis of optimization algorithms
  - Gradient Descent
  - Subgradient Methods
  - Proximal Gradient Methods
  - Nesterov's Accelerated Methods
  - Alternating Direction Method of Multipliers
  - Block Coordinate Descent




Portions of the formalized code have been **adopted into the official Mathlib4**

Chenyi Li, Ziyu Wang, Wanyi He, Yuxuan Wu, Shengyang Xu, and **Zaiwen Wen**. Formalization of Convergence Rates of Four First-order Algorithms. Journal of Automatic Reasoning.. 2025, 69, 4.

**Formal mathematics training programs**
**Over 200 high-quality formalization talents trained**

# Rockafellar《Convex Analysis》

- A classic in convex analysis
- Complete theory, clear structure
- A foundation for formalized optimization

Convex Analysis

BY

R. TYRRELL ROCKAFELLAR

THEOREM 5.8. *Let $f_1, \dots, f_m$ be proper convex functions on $R^n$. Then the following are convex functions also:*

$$f(x) = \inf\{\max\{f_1(x_1), \dots, f_m(x_m)\} \mid x_1 + \cdots + x_m = x\},$$
$$g(x) = \inf\{(f_1\lambda_1)(x) + \cdots + (f_m\lambda_m)(x) \mid \lambda_i \geq 0, \lambda_1 + \cdots + \lambda_m = 1\},$$
$$h(x) = \inf\{\max\{(f_1\lambda_1)(x), \dots, (f_m\lambda_m)(x)\} \mid \lambda_i \geq 0, \lambda_1 + \cdots + \lambda_m = 1\},$$
$$k(x) = \inf\{\lambda_1 f_1(x_1), \dots, \lambda_m f_m(x_m)\}\},$$

*where the last infimum is taken over all representations of $x$ as a convex combination $x = \lambda_1 x_1 + \cdots + \lambda_m x_m$.*

PROOF. In the sense of the preceding discussion, adding in $x$ alone yields $f$. Adding in $\lambda$ and $\mu$ yields $g$. Adding in $\lambda$ alone yields $h$. Adding in $\lambda$ and $x$ yields $k$. ‖

The first operation in Theorem 5.8 can be expressed in "convolution" form when $m = 2$:

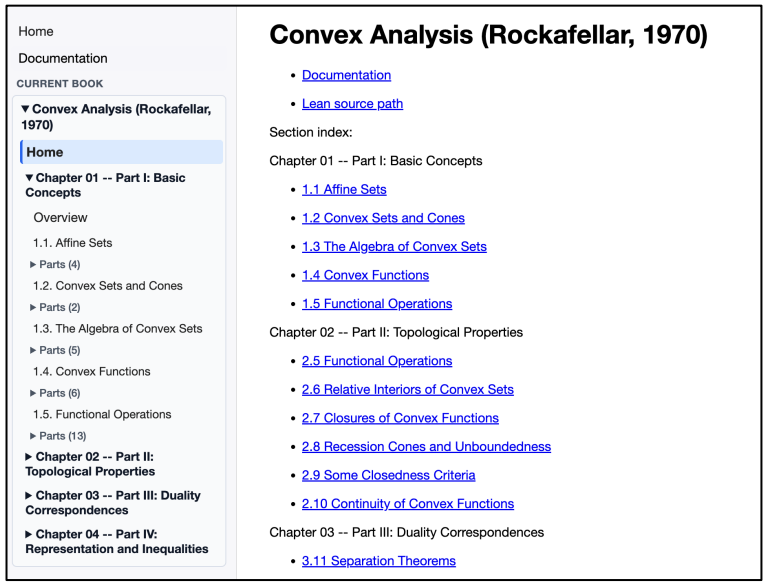
$$f(x) = \inf_y \max\{f_1(x - y), f_2(y)\}.$$

Observe that, with this operation,

$$\{x \mid f(x) < \alpha\} = \{x \mid f_1(x) < \alpha\} + \{x \mid f_2(x) < \alpha\},$$

for any $\alpha$. The third operation amounts to inverse addition of epigraphs.

- Keep book-reading experience
- Chapter browsing
- Docs match Lean source paths




```
theorem convexFunctionOn_inf_iSup_of_proper                    source
    {n m : ℕ} {f : Fin m → (Fin n → ℝ) → EReal}
    (hf : ∀ (i : Fin m), ProperConvexFunctionOn Set.univ (f i)) :
  ConvexFunctionOn Set.univ
   fun (x : Fin n → ℝ) =>

    sInf {z : EReal |
     ∃ (x' : Fin m → Fin n → ℝ), ∑ i : Fin m, x' i = x ∧ z = ⨆ (i : Fin m), f i (x' i)}
```

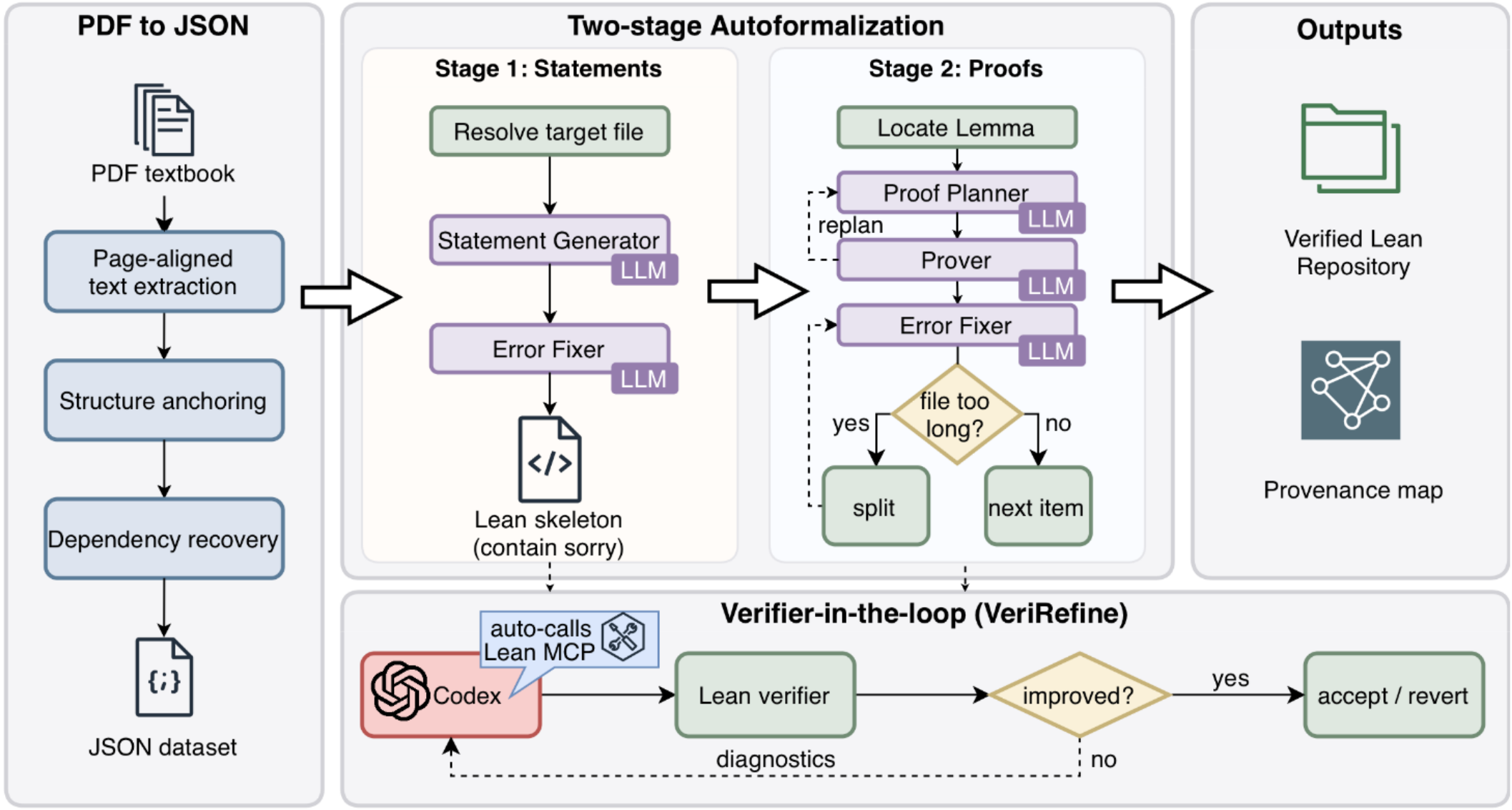
Theorem 5.8.1: Let `f_1, ..., f_m` be proper convex functions on `R^n`. Then `f(x) = inf { max { f_1(x_1), ..., f_m(x_m) } | x_1 + ... + x_m = x }` is convex.

## Classic text to formal knowledge

```
13  /-- Theorem 5.8.1: Let `f_1, ..., f_m` be proper convex functions on `ℝ^n`. Then
14  `f(x) = inf { max { f_1(x_1), ..., f_m(x_m) } | x_1 + ... + x_m = x }` is convex. -/
15  theorem convexFunctionOn_inf_iSup_of_proper {n m : Nat}
16      {f : Fin m → (Fin n → Real) → EReal}
17      (hf : ∀ i, ProperConvexFunctionOn (S := (Set.univ : Set (Fin n → Real))) (f i)) :
18      ConvexFunctionOn (S := (Set.univ : Set (Fin n → Real)))
19        (fun x =>
20          sInf { z : EReal |
21            ∃ x' : Fin m → (Fin n → Real),
22              (Finset.univ.sum (fun i => x' i) = x) ∧
23                z = iSup (fun i : Fin m => f i (x' i)) }) := by
24    classical
25    refine
26      (convexFunctionOn_univ_iff_strict_inequality (f := fun x =>
27          sInf { z : EReal |
28            ∃ x' : Fin m → (Fin n → Real),
29              (Finset.univ.sum (fun i => x' i) = x) ∧
30                z = iSup (fun i : Fin m => f i (x' i)) })).2 ?_
31    intro x y α β t hfx hfy ht0 ht1
32    set Sx : Set EReal :=
33      { z : EReal |
34        ∃ x' : Fin m → (Fin n → Real),
35          (Finset.univ.sum (fun i => x' i) = x) ∧
36            z = iSup (fun i : Fin m => f i (x' i)) }
37    set Sy : Set EReal :=
38      { z : EReal |
39        ∃ x' : Fin m → (Fin n → Real),
40          (Finset.univ.sum (fun i => x' i) = y) ∧
41            z = iSup (fun i : Fin m => f i (x' i)) }
42    set Sxy : Set EReal :=
43      { z : EReal |
44        ∃ x' : Fin m → (Fin n → Real),
```

**Makes a classic optimization text browsable, verifiable, and reusable**

# M2F: Automated Formalization at Scale




Wang Zichen, Ma Wanli, Ming Zhenyu, Zhang Gong, Yuan Kun, Wen Zaiwen,
**M2F: Automated Formalization of Mathematical Literature at Scale**

https://github.com/optsuite/M2F

# ReasBook: A textbook-/paper-scale Lean4 knowledge base

- A Lean 4 knowledge base for textbooks and research papers (compilable, verifiable, reusable)
- Goal: convert long-form math into a buildable Lean project, while keeping the original structure
- Content coverage: A continuously expanding collection of Books and Papers. 2026 target is **1,500 books** and **100,000 papers**.

  Examples: Tao, Analysis II; Rockafellar, Convex Analysis; Lebl, Real Analysis.
- **https://github.com/optsuite/ReasBook**

**Books**

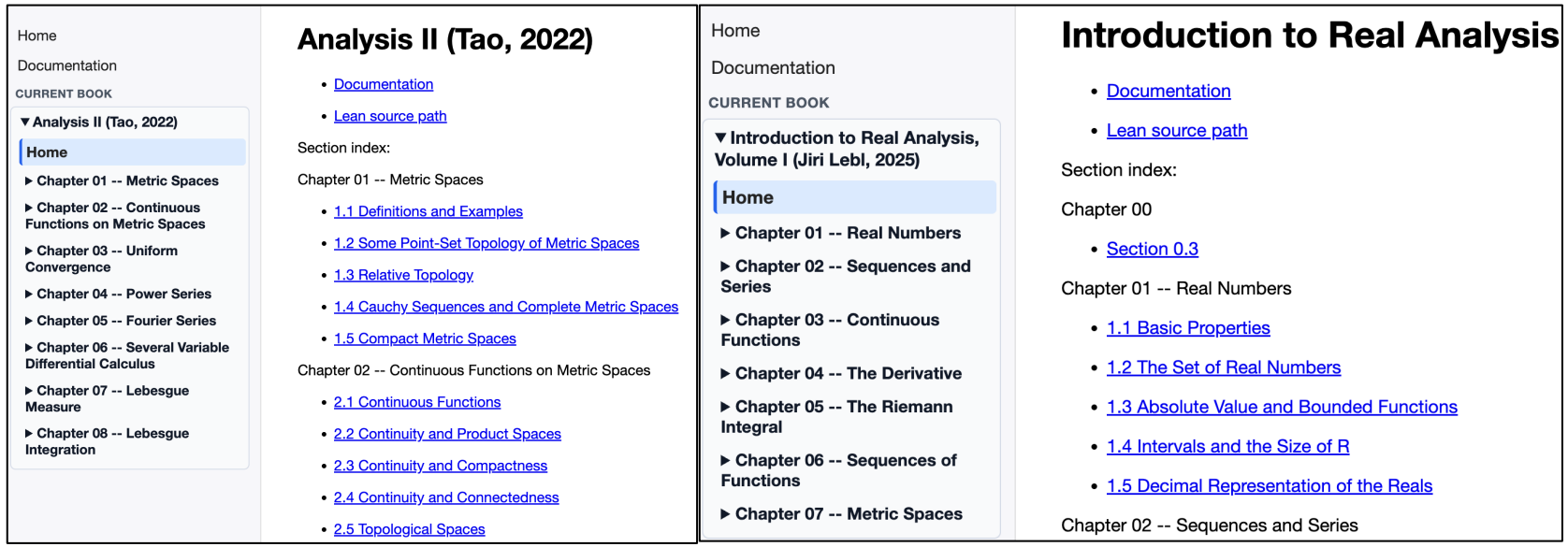
- Terence Tao, *Analysis II*, 4th ed., Hindustan Book Agency / Springer, Singapore, 2022, ISBN 978-981-19-7284-3.
  - Contributors: Chenyi Li, Min Cui, Qiming Dai, Shu Miao, Wanli Ma, Yi Yuan, Zichen Wang, Ziyu Wang.
  - Links: Documentation | Lean source | Verso
- R. Tyrrell Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, 1970, ISBN 0-691-08069-0.
  - Contributors: Changyu Zou, Chenyi Li, Guangxuan Pan, Pengfei Hao, Qiming Dai, Shu Miao, Siyuan Shao, Suwan Wu, Wanli Ma, Weiran Shi, Xinyi Guo, Xuran Sun, Yifan Bai, Yijie Wang, Yunfei Zhang, Yunxi Duan, Yuhao Jiang, Zebo Liu, Zhiyan Wang, Zichen Wang.
  - Links: Documentation | Lean source | Verso
- Jiri Lebl, *Introduction to Real Analysis, Volume I*, version 6.2, May 23, 2025, (TBD: publisher/city), (TBD: ISBN).
  - Contributors: Zichen Wang.
  - Links: Documentation | Lean source | Verso

**The goal is a buildable library that follows the book structure.**

# Repository organization: Lean source + documentation site

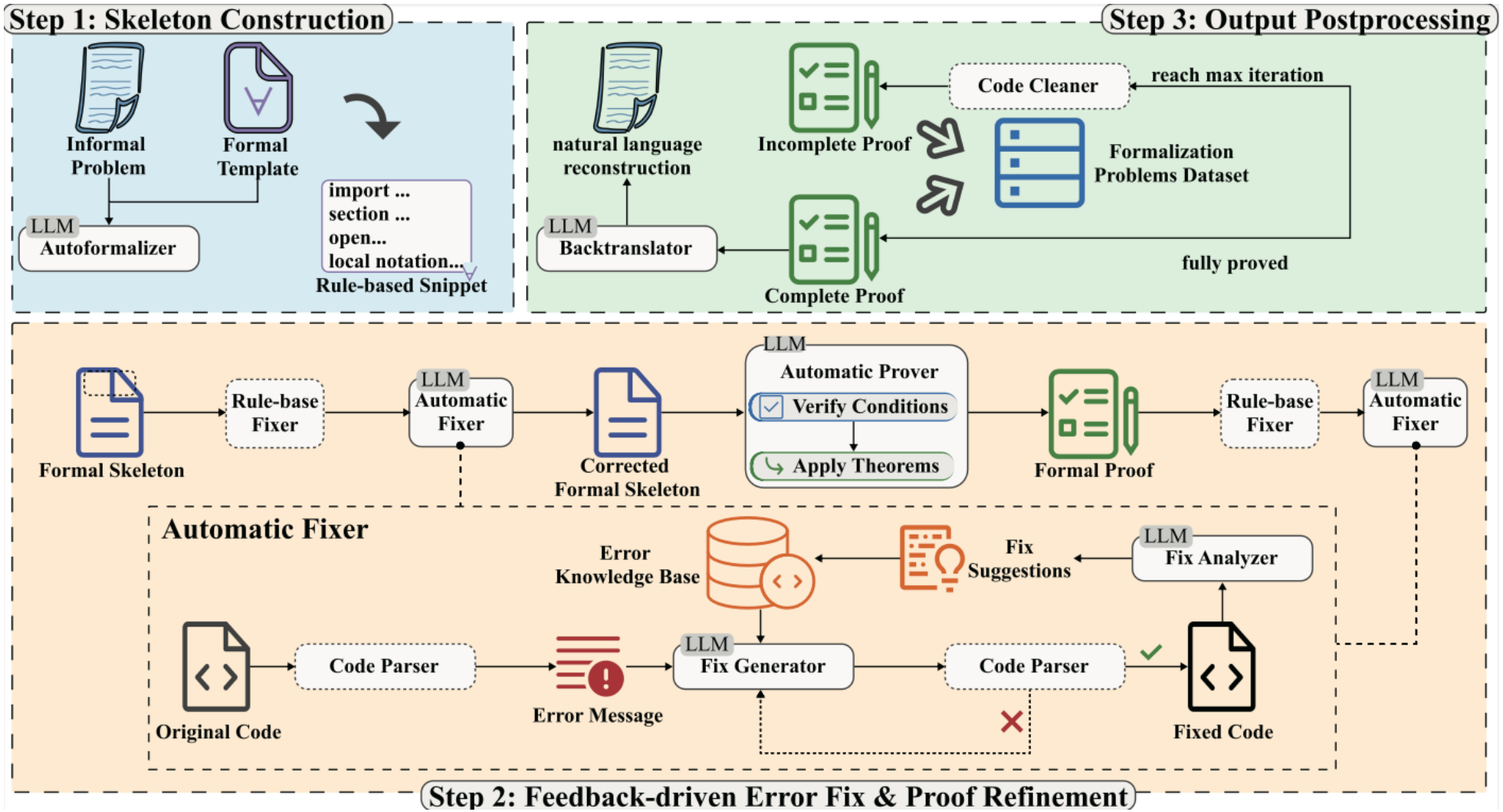
- **Browse like a book**: Organized by book/paper contents, with clear chapter structure for easy lookup and navigation
- **Two views of the same artifact:** Compilable Lean code, together with a web documentation site for a better reading experience
- **Reusable knowledge base:** Formalized definitions and lemmas can be directly imported by later chapters and other projects
- **Designed for continuous growth:** Add a new book or paper with a standard template and extend it incrementally
- **Easier collaboration:** Team members can review, fill proof gaps, refine details, and expand coverage together

- **Standard templates**
- **Unified docs**
- **Unified source management**




**ReasBook enables scalable formal math libraries with low-cost collaboration and high reuse**

# Automatic Theorem Formalization: From Structure to Instances




- Multi-agent collaborative formalization system (SITA);
- Structure-template-driven instance formalization
- Automatic instantiation of concrete optimization problems

Chenyi Li, Wanli Ma, Zichen Wang, **Wen Zaiwen**, SITA: A Framework for Structure-to-Instance Theorem Autoformalization. AAAI 2026.

# Automated formalization: $\min_x \frac{1}{2}||Ax-b||^2+\mu||x||_1$

## Abstract structure and property templates

```
class composite_pro (f : E → ℝ) (h : E
    → ℝ)
def composite_pro.target (_ :
    composite_pro f h) := f + h
class pg (pro : composite_pro f h) (x₀ :
    E) :=
  t : ℝ
  x : ℕ → E
  update : ∀ k : ℕ, prox_prop (t • h) (x
    k - t • (gradient f) (x k)) (x (k +
    1))
  initial : x 0 = x₀
```

**Proximal methods**

```
theorem pg_converge
  (xm : E) (L : NNReal)
  (fconv : ConvexOn ℝ univ f) (hconv :
    ConvexOn ℝ univ h)
  (h₁ : Differentiable ℝ f) (h₂ :
    LipschitzWith L (gradient f))
  (tpos : 0 < alg.t) (step : alg.t ≤ 1
    / L) (hL : L > (0 : ℝ)) :
  ∀ (k : ℕ+), (pro.target (alg.x k) -
    pro.target xm)
    ≤ 1 / (2 * k * alg.t) * ||x₀ - xm ||
    ^ 2 := by sorry
```

**Formal Convergence**

## Formal def: problems & algorithms

```
class Lasso_pro {m n : ℕ} (A : Matrix
    (Fin m) (Fin n) ℝ) (b : (Fin m) → ℝ
    ) (mu : ℝ) where
  (hA : A ≠ 0)
  (hmu : mu > 0)

def Lasso_pro.f (_ : Lasso_pro A b mu) :
    EuclideanSpace ℝ (Fin n) → ℝ :=
  fun x ↦ 1 / 2 * || A *ᵥ x - b ||₂ ^ 2

def Lasso_pro.g (_ : Lasso_pro A b mu) :
    EuclideanSpace ℝ (Fin n) → ℝ :=
  fun x ↦ mu * || x ||₁
```

**Formal def. of Lasso**

```
class pg_Lasso (pro : Lasso_pro A b mu)
    (x₀ : EuclideanSpace ℝ (Fin n)) where
  t : ℝ
  x : ℕ → EuclideanSpace ℝ (Fin n)
  y : ℕ → EuclideanSpace ℝ (Fin n)
  ht : t > 0
  update1 : ∀ k : ℕ,
    let grad : EuclideanSpace ℝ (Fin n)
    := Aᵀ *ᵥ (A *ᵥ x k - b)
    y k = x k - t • grad
  update2 : ∀ (k : ℕ), ∀ i, x (k + 1) i
    = (Real.sign (y k i) * (max (abs (y
    k i) - t * mu) 0))
  initial : x 0 = x₀
```

**Formalized algorithm For LASSO**

## Property verification

```
lemma Lasso_problem.lip_f (self :
    Lasso_problem A b mu) :
    LipschitzWith self.l (gradient
    self.f) := by
  rw [lipschitzWith_iff_norm_sub_le];
    intro x y
  rw [self.gradient_f, self.gradient_f]
  rw [← Matrix.mulVec_sub, ← sub_add,
    sub_add_eq_add_sub, sub_add_cancel]
  rw [← Matrix.mulVec_sub]
  simp
  apply Matrix.l2_opNorm_mulVec (Aᵀ * A)
```

**Gradient Lip. continuity**

```
theorem Lasso_convergence (alg :
    pg_Lasso pro x₀)
    (xm : EuclideanSpace ℝ (Fin n))
    (ht2 : alg.t ≤ 1 / pro.l):
    ∀ (k : ℕ+), (pro.target (alg.x k) -
    pro.target xm)
      ≤ 1 / (2 * k * alg.t) * || x₀ - xm
    || ^ 2 := by
intro k
  apply proximal_gradient_converge (alg
    := alg.pg)
    xm pro.l pro.ConvexOn_f
    pro.ConvexOn_g pro.diff_f pro.lip_f
    alg.ht ht2 pro.lpos k
```

**Formalized Convergence for Lasso**

## Full reports generation

**Definition 1** *Let $m$, $n$ be natural numbers, $A$ be an $m \times n$ real matrix, $b$ be an $m$-dimensional real vector, and $\mu$ be a real number. Define the* ***Lasso problem*** *class with conditions $A \neq 0$ and $\mu > 0$.*

*For a Lasso problem instance, define the function $f$ as $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ for $x \in \mathbb{R}^n$.*

*For a Lasso problem instance, define the function $g$ as $g(x) = \mu\|x\|_1$ for $x \in \mathbb{R}^n$.*

.....

**Theorem 1 (Lasso Convergence)** *For any proximal gradient method with step size $t \leq 1/l$, and any minimizer $x_m$, after $k$ iterations:*

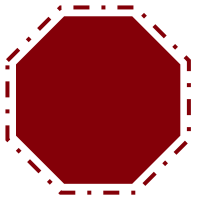
$$target(x_k) - target(x_m) \leq \frac{\|x_0 - x_m\|^2}{2kt}$$

**Proof.** Direct application of the proximal gradient convergence theorem to the Lasso problem, using the established properties: differentiability of $f$, convexity of $f$ and $g$, $L$-Lipschitz gradient for $f$, and valid proximal updates. □

**Report for Lasso**

- Basic def. & prop.
- Algorithm
- Convergence proofs

**Flexible domain knowledge framework via formal templates** → **Automated formalization: definitions, properties, theorems** → **Natural language report from formal**



# ReasLab

## https://model.reaslab.io

# ReasLab: Intelligent Mathematical Reasoning System

## Modeling Agent

## Modeling

**Multi-paradigm modeling**
**+**
**One-click Python scripts**

**Smarter than an LLM**

- Automatically detects problem classes
- Adaptively selects modeling paradigms
- Generates Python code and solves

## Solver Agent

## Algorithm Design

**Builds a solution pipeline**
**+**
**Autonomously calls algorithms**

**More efficient than manual tuning**

- Matches solution strategies and algorithms
- Supports diverse optimization tasks
- Generates code and solver output

## Natural language + formalization

## Theorem Proving

**Reads statements in natural language**
**+**
**Generates compilable Lean scripts**

**Mathlib-integrated proof checking**

- Translates Lean propositions from natural language
- Generates verifiable formal proofs
- Improves rigor and research efficiency

## Prompt-driven

## Interactive Writing

**AI-assisted exploration and conjecturing**
**+**
**Live LaTeX compilation**

**Beyond Overleaf**

- AI-assisted exploration and conjecturing
- Auto-generates TeX with live preview
- Supports AI-based document refinement


# Thank You!