Learning to Solve the Quadratic Assignment Problem with Warm-Started MCMC Finetuning

Haijun Zou

State Key Laboratory of Mathematical Sciences
Institute of Computational Mathematics and Scientific/Engineering Computing
Academy of Mathematics and Systems Science
Chinese Academy of Sciences, China

Joint work with Yicheng Pan, Ruisong Zhou, Tianyou Li, Zaiwen Wen

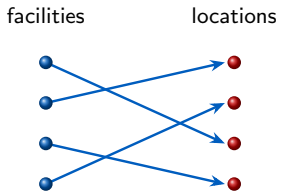
May 28, 2026

Outline

Quadratic Assignment Problem (QAP)

- A QAP instance is specified by a flow matrix $\boldsymbol{F} = (F_{ij})$ and a distance matrix $\boldsymbol{D} = (D_{kl})$.
- A permutation $\pi \in \Pi_n$ assigns each facility to one location.
- We use the permutation formulation:

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



π maps facilities to locations

Equivalent matrix form

$$f(\pi; \mathcal{P}) = \langle \boldsymbol{F}, \boldsymbol{X}_\pi \boldsymbol{D} \boldsymbol{X}_\pi^\top \rangle, \qquad \boldsymbol{X}_\pi \mathbf{1} = \boldsymbol{X}_\pi^\top \mathbf{1} = \mathbf{1}.$$

Here $\boldsymbol{X}_\pi$ is the permutation matrix induced by π.

Why is QAP difficult?

- Broad modeling power: facility layout, keyboard design, graph matching, graph partitioning, TSP reductions, and bandwidth minimization.
- Severe computational barrier: QAP is NP-hard and NP-hard to approximate; exact methods are often limited beyond $n \approx 20$.
- Strong heuristics exist, e.g., simulated annealing, tabu search, and memetic algorithms, but their performance is instance-dependent.
- Learning is attractive: reuse structure across instances and exploit GPU parallelism.
- The key challenge is that a QAP instance contains two coupled graphs, not a single graph.

Need a solver that keeps the speed of learned priors while adapting to unseen instance structures.

Brief literature review

- **Traditional QAP heuristics**
  - Ro-TS [Taillard 1991] and BMA [Benlic et al. 2015] are highly competitive baselines.
  - Robustness can degrade on hard structured families such as Taixxeyy [Drezner et al. 2005].
- **Learning-based QAP / graph matching**
  - Association-graph models use n^2 pair nodes [Wang et al. 2021; Liu et al. 2023].
  - Solution-aware Transformer learns swap improvement, but assumes complete initial solutions [Tan et al. 2024].
- **Deployment-time refinement**
  - Active search finetunes at test time [Bello et al. 2016; Hottung et al. 2022; Qiu et al. 2022].
  - Auto-regressive solvers restart construction after each update; prior high-quality solutions are hard to reuse.

Outline

Probabilistic learning formulation

- Replace direct search over Π_n by an optimization over distributions:

$$\min_{p \in \Delta(\Pi_n)} \mathbb{E}_{\pi \sim p} \left[f(\pi; \mathcal{P}) \right].$$

- This is equivalent to the deterministic QAP:

optimal probability distributions are supported on optimal permutations.

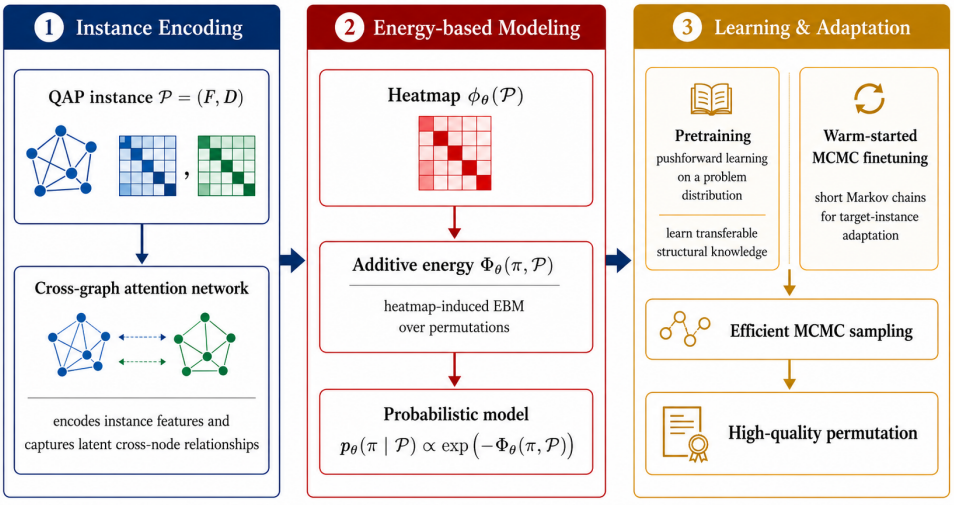
- Use an instance-conditioned model $p_\theta(\cdot \mid \mathcal{P})$ and learn over a distribution of instances Γ:

$$\min_{\theta} \mathbb{E}_{\mathcal{P} \sim \Gamma} \mathbb{E}_{\pi \sim p_\theta(\cdot|\mathcal{P})} \left[f(\pi; \mathcal{P}) \right].$$

- To smooth the rugged landscape, apply a 2-swap local improvement map $\mathcal{T} : \Pi_n \to \Pi_n$:

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

PLMA framework at a glance



Energy-based model over permutations

- Given a heatmap $\phi(\theta, \mathcal{P}) \in \mathbb{R}^{n \times n}$:

$$p_\theta(\pi \mid \mathcal{P}) \propto \exp(\langle \boldsymbol{X}_\pi, \phi \rangle) = \exp(\Phi_\theta(\pi)) .$$

- Additive score:

$$\Phi_\theta(\pi) = \sum_{i=1}^{n} \phi_{i,\pi(i)}.$$

- For a symmetric 2-swap proposal (a, b):

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

- Each MH acceptance step has $O(1)$ score-update cost.

Why this formulation?

- Directly samples feasible permutations.
- Additivity makes swap proposals cheap.
- Warm-started MCMC naturally refines previous high-quality solutions.

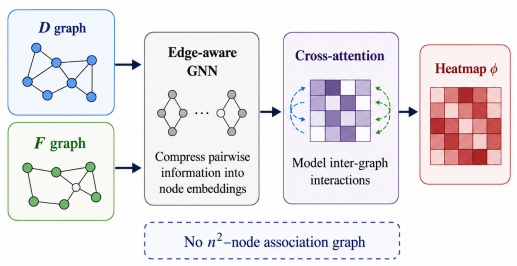
Cross-graph attention network

- A QAP instance is represented as two weighted graphs: location graph $\boldsymbol{D}$ and facility graph $\boldsymbol{F}$.
- Edge-aware GNN layers compress pairwise information into node embeddings:

$$\boldsymbol{D}, \boldsymbol{F} \quad \Longrightarrow \quad \boldsymbol{H}_D, \boldsymbol{H}_F \in \mathbb{R}^{n \times d}.$$

- Cross-attention fills the missing interactions between the two disconnected graphs.
- A dot product followed by log-Sinkhorn produces the heatmap:

$$\phi_\theta(\mathcal{P}) \;=\; \text{log-Sinkhorn}\left(C \tanh\left(\frac{\boldsymbol{H}_F \boldsymbol{H}_D^{\mathrm{T}}}{\sqrt{d}} \right) \right).$$



Scalability

The node space remains $O(2n)$, avoiding an $O(n^2)$ association graph while still modeling inter-graph coupling.

Stage I: pretraining with pushforward transformation

- Goal: learn transferable structural priors from a training distribution Γ.
- At each step:
  1. sample a batch $\{\mathcal{P}_i\}_{i=1}^{B} \sim \Gamma$;
  2. draw N permutations by parallel MH chains;
  3. refine each sample by $\hat{\pi}_{i,k} = \mathcal{T}(\pi_{i,k})$;
  4. update θ using the improved costs $f(\hat{\pi}_{i,k}; \mathcal{P}_i)$.
- The gradient estimator uses an instance-wise baseline:

$$\hat{G} = \frac{1}{B(N-1)} \sum_{i=1}^{B} \sum_{k=1}^{N} \left(\hat{f}_{i,k} - b_i \right) \nabla_\theta \Phi_\theta(\pi_{i,k}), \qquad b_i = \frac{1}{N} \sum_{k=1}^{N} \hat{f}_{i,k}.$$

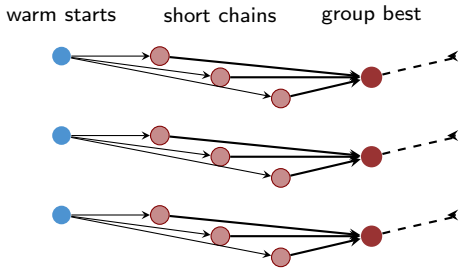
- Long chains are used here to stabilize sampling in the cold-start regime.

Stage II: batch-wise warm-started MCMC finetuning

- Start from the pretrained EBM and target instances $\{\mathcal{P}_i\}_{i=1}^{B}$.
- Generate K high-quality initial states with long-run MH sampling.
- For each state, launch M short chains:

$$\pi_{i,k} \quad \Longrightarrow \quad \{\pi_{i,k}^{m}\}_{m=1}^{M} \quad \Longrightarrow \quad \{\mathcal{T}(\pi_{i,k}^{m})\}_{m=1}^{M}.$$

- Update the shared model with one batch gradient over all instances and chains.
- Retain the best sample within each group k as the next warm start.



preserve quality while keeping diversity

Design principle

Short chains keep exploration near promising regions already discovered by the previous iteration.

Outline

Experimental setup

- **Hardware:** NVIDIA Tesla A800 GPUs and Intel Xeon Gold 6326 CPUs.
- **Datasets:**
  - synthetic QAP: geometrically structured instances, $n \in \{20, 50, 100\}$;
  - out-of-distribution large-scale random instances, $n \in \{200, 500\}$;
  - QAPLIB and Taixxeyy benchmarks.
- **Baselines:** Ro-TS, BMA, Connolly's SA, IPFP, SM, RRWM, NGM, SAWT.
- **Metrics:** average objective, relative gap, total runtime; for Taixxeyy, mean and range over 10 independent runs.

Synthetic QAP results

	QAP20			QAP50			QAP100		
Algorithm	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
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
BMA	**54.37**	**0.00%**	1m57s	375.60	0.03%	15m53s	1591.58	0.02%	2h21m
C-SA	54.53	0.28%	1m57s	376.56	0.29%	21m32s	1592.95	0.11%	2h45m
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
RRWM	71.30	31.09%	11.30s	428.78	14.14%	39.23s	1700.33	6.86%	6m32s
SM	64.38	18.45%	0.22s	426.92	13.70%	7.14s	1753.10	10.17%	1m40s
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

- The pretrained model already provides strong zero-shot solutions.
- Finetuning quickly closes the remaining gap while retaining a large runtime advantage.

Large-scale QAP results

	QAP200		QAP500	
Algorithm	Gap	Time	Gap	Time
Ro-TS	0.00%	3h48m	0.00%	29h59m
BMA	0.01%	2h58m	-0.09%	23h16m
SAWT (10k)	1.44%	8m33s	1.55%	48m36s
PLMA ($T = 1$)	0.53%	7.50s	0.38%	1m31s
PLMA ($T = 50$)	**-0.01%**	6m15s	**-0.12%**	1h15m

PLMA remains competitive or better than strong handcrafted baselines at $n = 200, 500$, with substantially lower runtime.

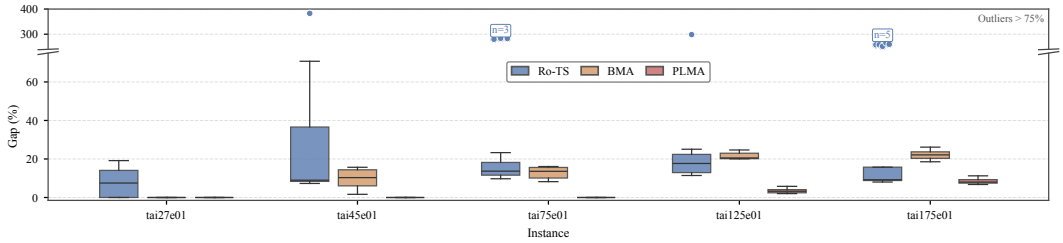
QAPLIB benchmark

	Ro-TS		BMA		SAWT		IPFP		PLMA	
Class	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
bur (26)	**0.00%**	0.12	**0.00%**	0.03	3.95%	14.67	0.05%	0.33	**0.00%**	0.08
chr (12-25)	0.48%	0.16	0.18%	0.14	147.54%	14.18	14.90%	0.26	**0.00%**	0.31
els (19)	**0.00%**	0.02	**0.00%**	0.02	47.37%	14.24	10.18%	0.36	**0.00%**	0.09
esc (16-128)	**0.00%**	0.62	**0.00%**	0.01	43.29%	15.07	0.39%	0.63	**0.00%**	0.07
had (12-20)	**0.00%**	0.00	**0.00%**	0.00	5.17%	14.23	0.08%	0.39	**0.00%**	0.05
kra (30-32)	**0.00%**	0.24	**0.00%**	0.07	32.92%	14.77	0.65%	0.62	**0.00%**	0.31
lipa (20-90)	0.03%	3.28	**0.02%**	2.35	1.40%	17.32	1.07%	4.93	0.08%	1.57
nug (12-30)	**0.00%**	0.02	**0.00%**	0.02	19.25%	14.39	0.05%	0.38	**0.00%**	0.17
rou (12-20)	**0.00%**	0.04	**0.00%**	0.04	15.09%	14.25	0.77%	0.33	**0.00%**	0.08
scr (12-20)	**0.00%**	0.01	**0.00%**	0.01	33.92%	14.22	1.24%	0.28	**0.00%**	0.08
sko (42-100)	0.05%	29.22	**0.03%**	21.00	16.17%	19.06	0.30%	8.40	**0.03%**	7.01
ste (36)	0.01%	0.53	**0.00%**	0.30	107.95%	14.95	1.81%	0.70	**0.00%**	0.55
tai (10-256)	0.23%	25.16	0.21%	19.06	34.67%	16.62	0.92%	3.33	**0.20%**	4.73
tho (30-150)	**0.04%**	38.74	**0.04%**	26.59	24.05%	17.19	0.42%	12.46	**0.04%**	6.42
wil (50-100)	**0.02%**	26.06	**0.02%**	18.41	9.52%	17.64	0.12%	9.66	**0.02%**	6.82
Average	0.11%	9.68	0.07%	7.06	37.82%	15.85	2.15%	2.73	**0.06%**	2.16

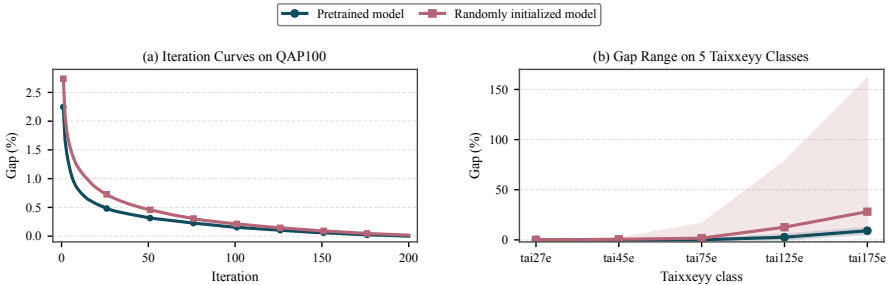
PLMA attains the lowest average gap and the shortest average time among the compared heuristic baselines.

Robustness on Taixxeyy instances

	Ro-TS			BMA			IPFP			PLMA		
Class	mean	[min,max]	time	mean	[min,max]	time	mean	[min,max]	time	mean	[min,max]	time
tai27e	41.50	[0.11,221.08]	0.57	0.12	[0.00,0.67]	0.18	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	8.39	[0.74,17.63]	2.21	22.01	[8.26,36.98]	1.03	**0.00**	**[0.00,0.00]**	0.18
tai75e	111.28	[6.20,280.01]	18.49	15.41	[5.77,22.76]	11.21	27.64	[17.56,36.78]	2.52	**0.08**	**[0.00,0.63]**	1.27
tai125e	82.53	[7.65,265.54]	72.52	16.51	[10.56,21.72]	66.01	26.93	[20.64,32.67]	8.52	**3.65**	**[0.78,6.32]**	8.65
tai175e	67.86	[9.11,260.98]	158.18	21.07	[15.15,26.56]	193.65	23.32	[17.70,28.11]	16.86	**9.09**	**[5.96,12.24]**	14.43
Average	81.01	[4.82,285.64]	50.72	12.30	[6.44,17.87]	54.65	23.87	[14.06,33.73]	5.86	**2.56**	**[1.35,3.84]**	**4.92**



Ablation: why the two-stage design matters



- Pretrained finetuning reaches lower gaps in fewer iterations on in-distribution random QAP100.
- On Taixxeyy, pretraining reduces both mean gap and variability across hard instance classes.
- The warm-started short-chain stage can exploit the pretrained prior instead of restarting search from scratch.

Conclusions

- PLMA formulates QAP solving as learning an instance-conditioned distribution over permutations.
- The pushforward learning objective trains the sampler toward regions that become high-quality after local improvement.
- The additive EBM enables constant-time 2-swap MH score updates and naturally supports warm starts.
- Two-stage learning is central:
  - pretraining learns transferable structural priors;
  - deployment-time finetuning adapts through short warm-started MCMC chains.
- Experiments show strong performance on synthetic QAP, QAPLIB, Taixxeyy and large-scale instances.

Many Thanks For Your Attention!