NJUAI 南京大学 人工智能学院
SCHOOL OF ARTIFICIAL INTELLIGENCE, NANJING UNIVERSITY

LAMDA
Learning And Mining from DatA
http://www.lamda.nju.edu.cn

南京大學 NANJING UNIVERSITY 1902

Some Preliminary Attempts on Learning to Optimize

Chao Qian

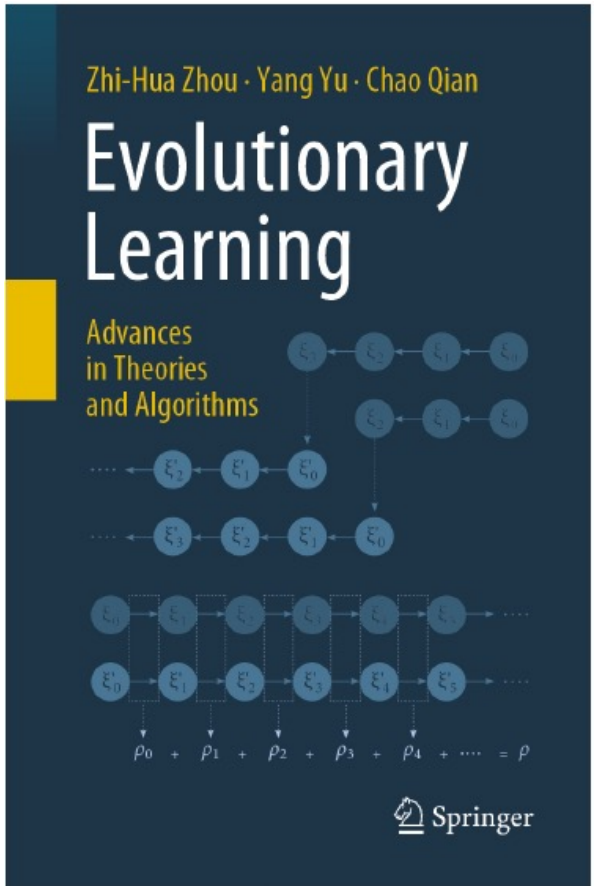
http://www.lamda.nju.edu.cn/qianc/

Email: qianc@nju.edu.cn

LAMDA Group

Nanjing University, China

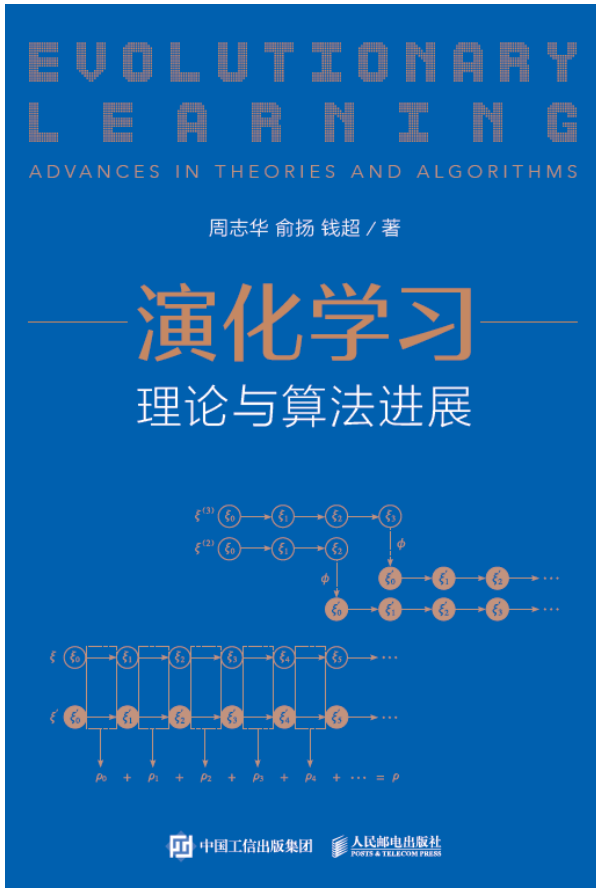
Build Theoretical Foundation of Black-box Optimization



Zhi-Hua Zhou, Yang Yu, Chao Qian

Evolutionary Learning: Advances in Theories and Algorithms

- Presents theoretical results for evolutionary learning
- Provides general theoretical tools for analysing evolutionary algorithms
- Proposes evolutionary learning algorithms with provable theoretical guarantees

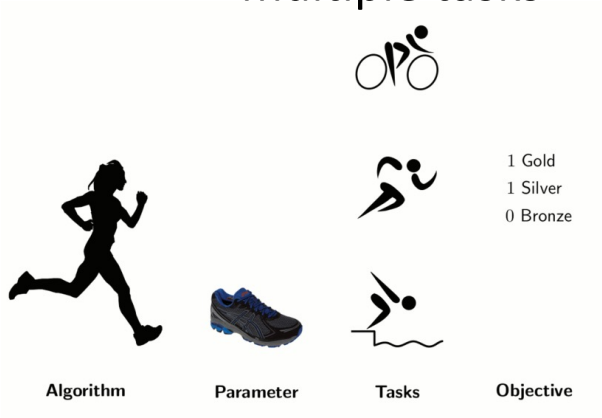


Learning to Optimize

- Can we learn to configure optimization algorithms dynamically?
- Can we learn to construct optimization algorithms directly?
- Can we learn to select proper optimization algorithms automatically?

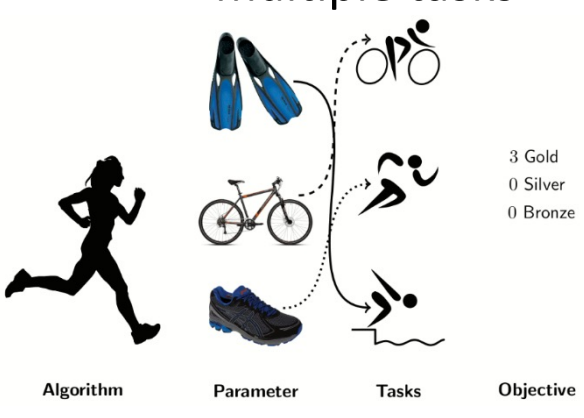
Dynamic Algorithm Configuration

multiple tasks



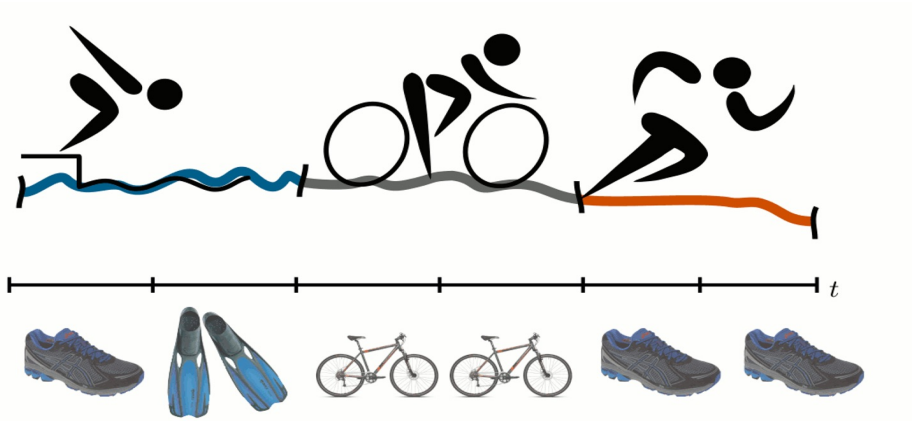
Algorithm Configuration (AC)

multiple tasks



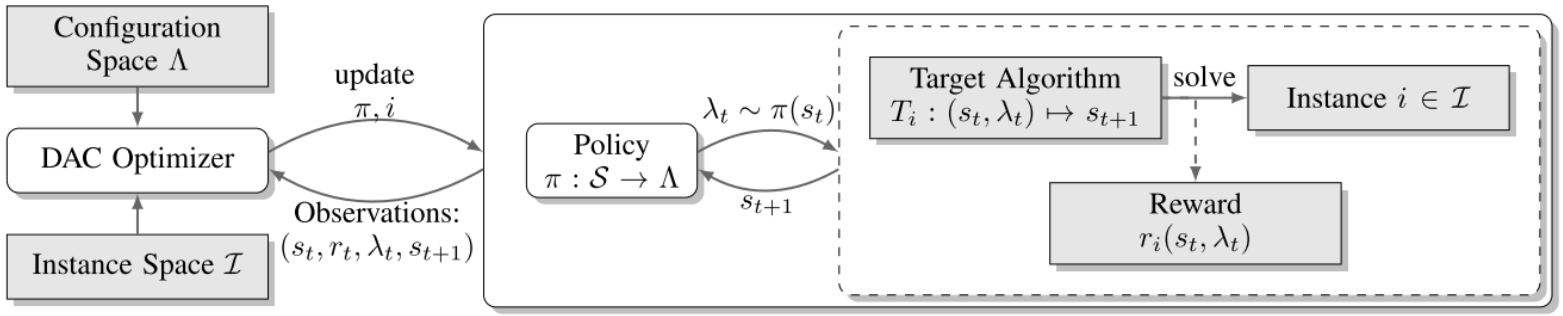
Per-instance AC

a single task



Dynamic AC (DAC)

DAC is a new trend in Auto-ML



$$\pi^* \in \arg\min_{\pi \in \Pi} \int_{\mathcal{F}} p(f) c(\pi, f) \mathrm{d}f$$

from AutoML.org

Dynamic Algorithm Configuration

Frank Hutter

ELLIS Institute & Professor, University of Freiburg, Germany
在 cs.uni-freiburg.de 的电子邮件经过验证 - 首页

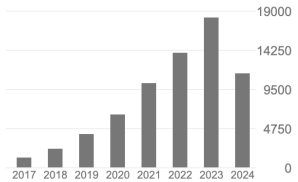
AutoML Meta-Learning Neural Architecture Search Deep Learning Machine Learning

关注

标题	引用次数	年份
Decoupled weight decay regularization I Loshchilov, F Hutter ICLR 2019	19722	2017
Sgdr: Stochastic gradient descent with warm restarts I Loshchilov, F Hutter ICLR 2017	8112	2016

引用次数 查看全部

	总计	2019 年至今
引用	72969	64584
h 指数	87	77
i10 指数	194	183



Automated Dynamic Algorithm Configuration

Steven Adriaensen ADRIAENS@CS.UNI-FREIBURG.DE
André Biedenkapp BIEDENKA@CS.UNI-FREIBURG.DE
Gresa Shala SHALAG@CS.UNI-FREIBURG.DE
Noor Awad AWAD@CS.UNI-FREIBURG.DE
University of Freiburg, Machine Learning Lab

Theresa Eimer EIMER@TNT.UNI-HANNOVER.DE
Marius Lindauer LINDAUER@TNT.UNI-HANNOVER.DE
Leibniz University Hannover, Institute for Information Processing

Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg, Machine Learning Lab & Bosch Center for Artificial Intelligence

[Adriaensen et al., JAIR 2023]

7.2 Limitations and Further Research

While these case studies and other previous applications provide a "proof of concept" for automated DAC, we point out that much remains to be done to unlock its full potential, and we hope that this work may serve as a stepping stone for further exploring this promising line of research. In what remains, we will discuss some of the limitations of contemporary work and provide specific directions for future research.

Jointly configuring many parameters: While static approaches are capable of jointly configuring hundreds of parameters, the configuration space in contemporary DAC is typically much smaller, often considering only a single parameter. While the configuration space is smaller, the candidate solution space (i.e., the dynamic configuration policy space) grows exponentially with the number of reconfiguration points, in the worst case, and is thus typically drastically larger than static configuration policy spaces. Although modern techniques from reinforcement learning scale much better than ever before, we still know too little about the internal structure of DAC problems to handle this exploding space of possible policies. For example, not much is known regarding interaction effects of parameters in the DAC setting. If there should be only a few interaction effects between parameters as in static AC (Hutter et al., 2014; Wang et al., 2016), learning several independent policies might be a way forward.

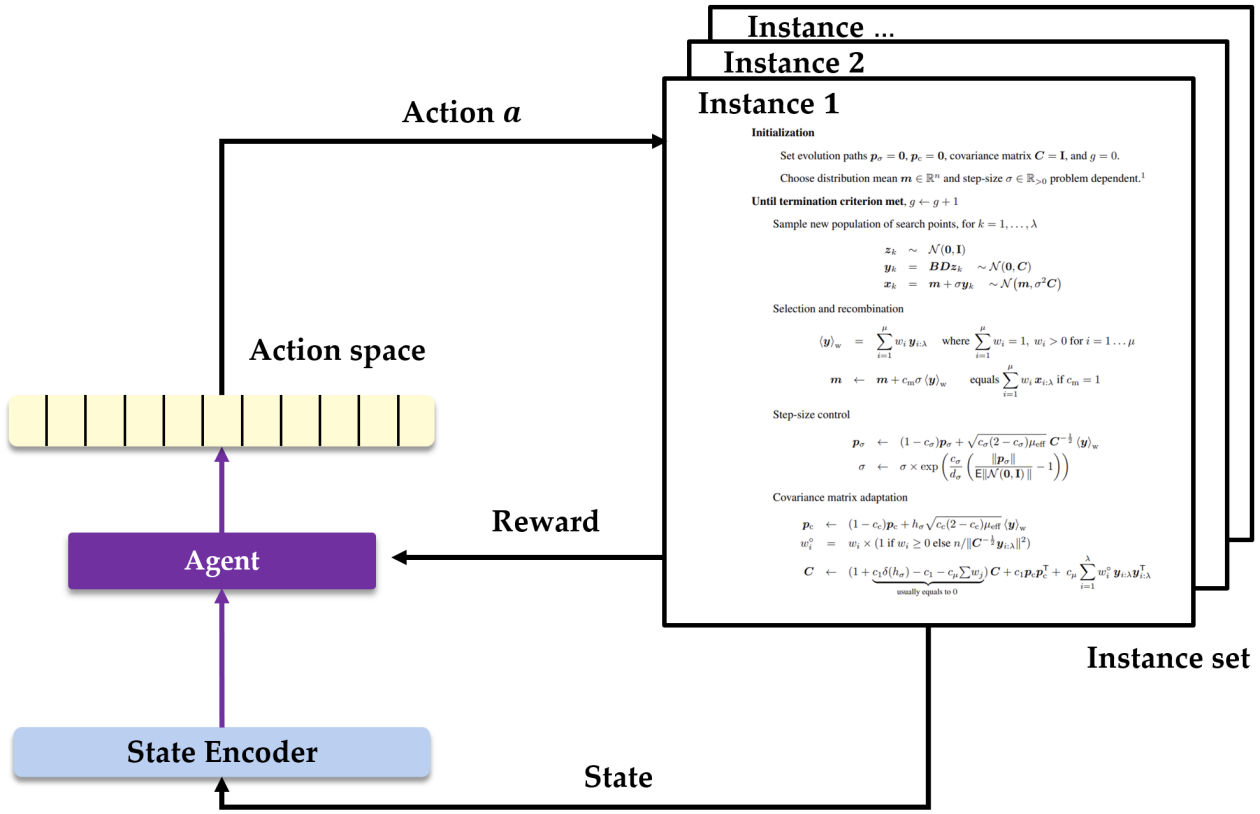
Open Problem: How to adjust multiple heterogeneous hyper-parameters simultaneously?

Open Problem: How to adjust multiple heterogeneous hyper-parameters simultaneously?

One direct method: treat multiple hyper-parameters as a whole, and apply RL methods

Very difficult

because of the large action space!



Dynamic Algorithm Configuration

For example:

Online Selection of CMA-ES Variants

Diederick Vermetten
Leiden Institute for Advanced Computer Science
Leiden, The Netherlands

Sander van Rijn
Leiden Institute for Advanced Computer Science
Leiden, The Netherlands

Thomas Bäck
Leiden Institute for Advanced Computer Science
Leiden, The Netherlands

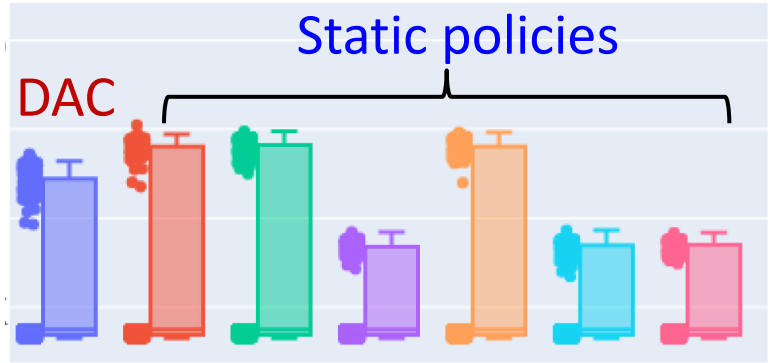
Carola Doerr
Sorbonne University, LIP6, and CNRS
Paris, France

#	Module name	0	1	2
1	Active Update [20]	off	on	-
2	Elitism	(μ, λ)	$(\mu+\lambda)$	-
3	Mirrored Sampling [9]	off	on	-
4	Orthogonal Sampling [29]	off	on	-
5	Sequential Selection [9]	off	on	-
6	Threshold Convergence [24]	off	on	-
7	TPA [12]	off	on	-
8	Pairwise Selection [1]	off	on	-
9	Recombination Weights [3]	$\log(\mu+\frac{1}{2})-\frac{\log(i)}{\sum_j w_j}$	$\frac{1}{\mu}$	-
10	Quasi-Gaussian Sampling	off	Sobol	Halton
11	Increasing Population [2, 13]	off	IPOP	BIPOP

[Vermetten et al., GECCO 2019]

ModEA includes an example of dynamic algorithm selection for variants of CMA-ES on BBOB functions [Vermetten *et al.*, 2019]. In contrast to the CMA-ES benchmark, a combination of 11 EA elements with two to three options each are chosen in each step; this combination makes up the final algorithm. This multi-dimensional, large action space makes the problem very complex. So we expect this to be a hard benchmark, possibly too hard for current DAC approaches to efficiently determine an effective DAC policy.

$2^9 \cdot 3^2 = 4608$ possible actions [Eimer et al., IJCAI 2021]



The smaller the better

DAC policy can be even worse than static policies

Multi-Agent Dynamic Algorithm Configuration

Open Problem: How to adjust multiple heterogeneous hyper-parameters simultaneously?

Our solution:

- Cooperative multi-agent modeling
- Each agent handles one hyper-parameter

Problem Formulation: Contextual Multi-agent Markov Decision Process $\mathcal{M}_I := \{\mathcal{M}_i\}_{i\sim I}$

$\mathcal{M}_i :=< \mathcal{N}, \mathcal{S}, \{\mathcal{A}_j\}_{j=1}^{n}, \mathcal{T}_i, r_i >$ is a single MMDP, corresponding to a problem instance $i \in I$

To find a joint policy $\boldsymbol{\pi}: \mathcal{S} \to \Delta(\mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n)$ maximizing the global value function:

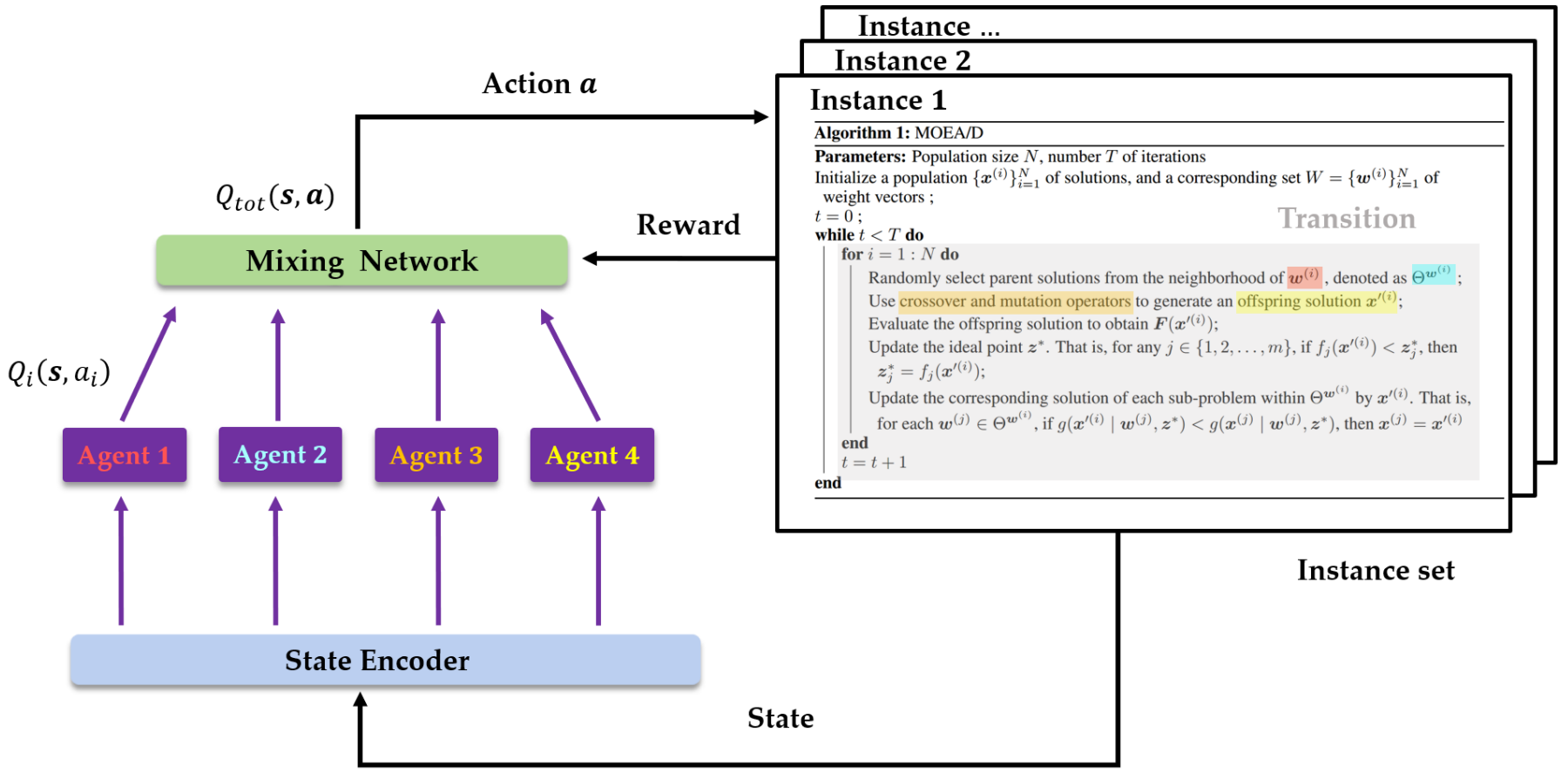
$$Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t) | s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a}\right]$$

[Xue et al., NeurIPS 2022 Spotlight]

Open Problem: How to adjust multiple heterogeneous hyper-parameters simultaneously?

Our solution:

- Cooperative multi-agent modeling
- Each agent handles one hyper-parameter



How to apply multi-agent DAC?

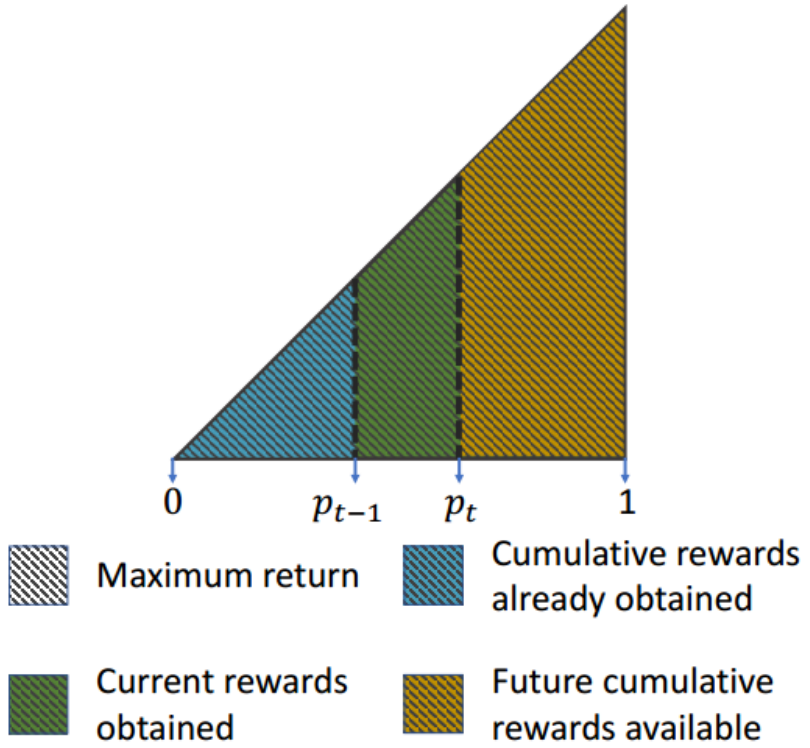
Guidelines for state design

- Accessibility
- Representation
- Generalization

Guidelines for reward design

- Higher reward in the later period

$$r_t = \begin{cases} (1/2) \cdot (p_t^2 - p_{t-1}^2) & \text{if } f(s_{t+1}) < f_t^* \\ 0 & \text{otherwise} \end{cases}$$

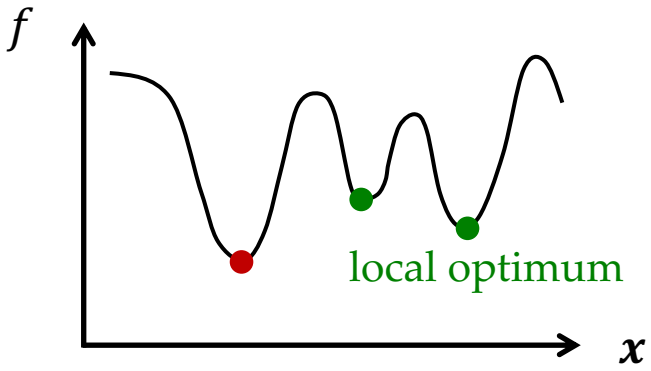


$$p_{t+1} = \begin{cases} \frac{f(s_0) - f(s_{t+1})}{f(s_0)} & \text{if } f(s_{t+1}) < f_t^* \\ p_t & \text{otherwise} \end{cases}$$

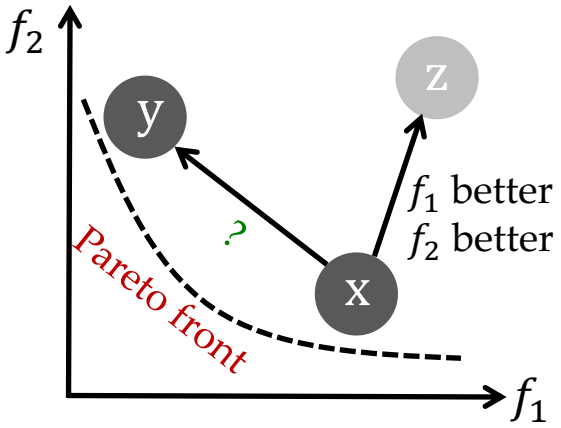
[Xue et al., NeurIPS 2022 Spotlight]

Application of Multi-Agent DAC to MOEA/D

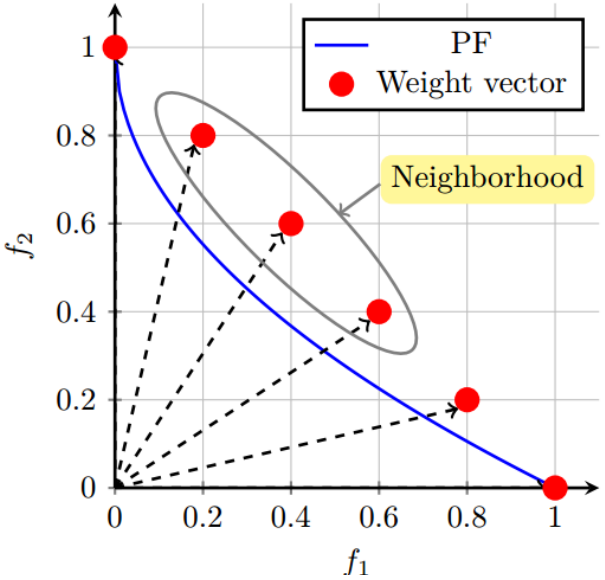
Single-objective



Multi-objective



MOEA/D [Zhang and Li, TEvC 2007]



Step 1: Initialize a population of solutions $P = \{\mathbf{x}^i\}_{i=1}^N$, a set of weight vectors $W = \{\mathbf{w}^i\}_{i=1}^N$ and their neighborhood structure. Randomly assign each solution to a weight vector.

Step 2: For $i = 1, \cdots, N$, do

- Step 2.1: Randomly select a required number of mating parents from $\mathbf{w}^i$'s neighborhood, denoted as $\Theta^{\mathbf{w}^i}$.
- Step 2.2: Use crossover and mutation to reproduce offspring $\mathbf{x}^c$.
- Step 2.3: Update the subproblems within $\Theta^{\mathbf{w}^i}$ by $\mathbf{x}^c$ if $g(\mathbf{x}^c|\mathbf{w}^k, \mathbf{z}^*) < g(\mathbf{x}^k|\mathbf{w}^k, \mathbf{z}^*)$, where $\mathbf{w}^k \in \Theta^{\mathbf{w}^i}$ and $\mathbf{x}^k$ is the current solution associated with $\mathbf{w}^k$.

Step 3: If the stopping criteria is met, then stop and output the population. Otherwise, go to Step 2.

Tune four hyper-parameters dynamically

[Xue et al., NeurIPS 2022 Spotlight]

Application of Multi-Agent DAC to MOEA/D

State

- *Accessibility*. The state should be accessible in each step
- *Representability*. The state should reflect the information in the optimization process
- *Generalizability*. The learned policy is expected to generalize to other instances

Index	Parts of state	Feature	Notes
0	1	$1/m$	m: Number of objectives
1	1	$1/D$	D: Number of variables
2	2	t/T	Computational budget that has been used
3	2	N_{stag}/T	Stagnant count ratio
4	3	HV_t	Hypervolume value
5	3	NDRatio_t	Ratio of non-dominated solutions
6	3	Dist_t	Average distance
7	3	$\text{HV}_t - \text{HV}_{t-1}$	Change of HV between steps t and $t-1$
8	3	$\text{NDRatio}_t - \text{NDRatio}_{t-1}$	Change of NDRatio between steps t and $t-1$
9	3	$\text{Dist}_t - \text{Dist}_{t-1}$	Change of Dist between steps t and $t-1$
10	3	$\text{Mean}(\text{List}(\text{HV}, t, 5))$	Mean of HV in the last 5 steps
11	3	$\text{Mean}(\text{List}(\text{NDRatio}, t, 5))$	Mean of NDRatio in the last 5 steps
12	3	$\text{Mean}(\text{List}(\text{Dist}, t, 5))$	Mean of Dist in the last 5 steps
13	3	$\text{Std}(\text{List}(\text{HV}, t, 5))$	Standard deviation of HV in the last 5 steps
14	3	$\text{Std}(\text{List}(\text{NDRatio}, t, 5))$	Standard deviation of NDRatio in the last 5 steps
15	3	$\text{Std}(\text{List}(\text{Dist}, t, 5))$	Standard deviation of Dist in the last 5 steps
16	3	$\text{Mean}(\text{List}(\text{HV}, t, t))$	Mean of HV in all the steps so far
17	3	$\text{Mean}(\text{List}(\text{NDRatio}, t, t))$	Mean of NDRatio in all the steps so far
18	3	$\text{Mean}(\text{List}(\text{Dist}, t, t))$	Mean of Dist in all the steps so far
19	3	$\text{Std}(\text{List}(\text{HV}, t, t))$	Standard deviation of HV in all the steps so far
20	3	$\text{Std}(\text{List}(\text{NDRatio}, t, t))$	Standard deviation of NDRatio in all the steps so far
21	3	$\text{Std}(\text{List}(\text{Dist}, t, t))$	Standard deviation of Dist in all the steps so far

Application of Multi-Agent DAC to MOEA/D

Action: the value of each hyper-parameter

Reward

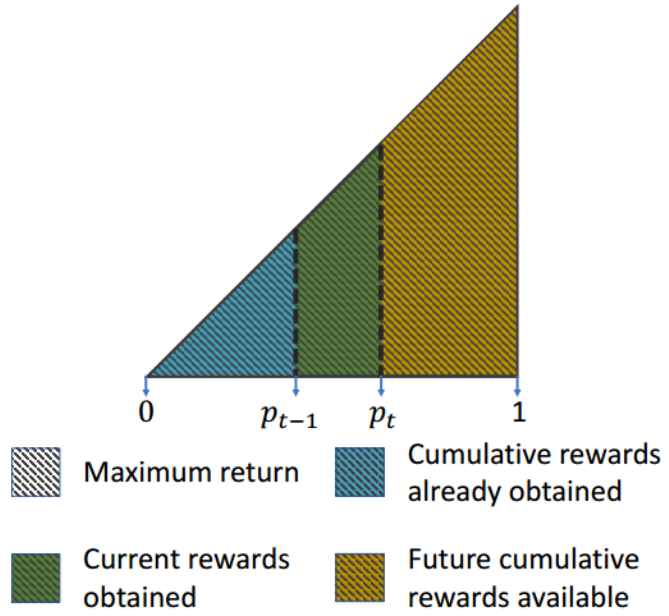
$$r_t^1 = \max\{f(s_t) - f(s_{t+1}), 0\},$$

$$r_t^2 = \begin{cases} 10 & \text{if } f(s_{t+1}) < f_t^* \\ 1 & \text{else if } f(s_{t+1}) < f(s_t) \\ 0 & \text{otherwise} \end{cases},$$

$$r_t^3 = \max\left\{\frac{f(s_t) - f(s_{t+1})}{f(s_{t+1}) - f_{\text{opt}}}, 0\right\},$$

$$p_{t+1} = \begin{cases} \frac{f(s_0) - f(s_{t+1})}{f(s_0)} & \text{if } f(s_{t+1}) < f_t^* \\ p_t & \text{otherwise} \end{cases}$$

$$r_t = \begin{cases} (1/2) \cdot (p_t^2 - p_{t-1}^2) & \text{if } f(s_{t+1}) < f_t^* \\ 0 & \text{otherwise} \end{cases}$$



Transition: one time-step in MARL is one generation in MOEA/D

Application of Multi-Agent DAC to MOEA/D

We investigate the following three research questions (RQs):

- RQ1: How does MA-DAC *perform* compared with the baseline and other tuning algorithms?
- RQ2: How is the *generalization ability* of MA-DAC?
- RQ3: How do the *different parts* of MA-DAC affect the performance?

Experiments – RQ1

Table 2: IGD values obtained by MOEA/D, DQN, MA-UCB and MA-DAC on different problems. Each result consists of the mean and standard deviation of 30 runs. The best mean value on each problem is highlighted in **bold**. The symbols '+', '−' and '≈' indicate that the result is significantly superior to, inferior to, and almost equivalent to MA-DAC, respectively, according to the Wilcoxon rank-sum test with confidence level 0.05.

Problem	M	MOEA/D	DQN	MA-UCB	MA-DAC
DTLZ2	3	4.605E-02 (3.54E-04) −	4.628E-02 (2.96E-04) −	4.671E-02 (3.70E-04) −	**3.807E-02** (5.05E-04)
	5	3.006E-01 (1.55E-03) −	3.016E-01 (1.34E-03) −	3.041E-01 (1.69E-03) −	**2.442E-01** (1.26E-02)
	7	4.455E-01 (1.41E-02) −	4.671E-01 (1.15E-02) −	4.826E-01 (9.59E-03) −	**3.944E-01** (1.17E-02)
WFG4	3	5.761E-02 (5.41E-04) −	6.920E-02 (1.20E-03) −	7.165E-02 (1.83E-03) −	**5.200E-02** (1.19E-03)
	5	3.442E-01 (1.21E-02) −	2.810E-01 (6.86E-03) −	2.859E-01 (6.77E-03) −	**1.868E-01** (2.81E-03)
	7	4.529E-01 (1.79E-02) −	3.725E-01 (1.14E-02) −	3.868E-01 (1.54E-02) −	**3.033E-01** (3.66E-03)
WFG6	3	6.938E-02 (5.50E-03) −	6.834E-02 (1.78E-02) −	6.601E-02 (1.00E-02) −	**4.831E-02** (8.95E-03)
	5	3.518E-01 (2.82E-03) −	3.160E-01 (2.40E-02) −	3.359E-01 (1.47E-02) −	**1.942E-01** (6.90E-03)
	7	4.869E-01 (3.03E-02) −	4.322E-01 (2.95E-02) −	4.389E-01 (3.41E-02) −	**3.112E-01** (4.93E-03)
Train: +/−/≈		0/9/0	0/9/0	0/9/0	
DTLZ4	3	6.231E-02 (8.85E-02) ≈	**5.590E-02** (5.77E-03) −	6.011E-02 (5.08E-03) −	6.700E-02 (6.14E-02)
	5	3.133E-01 (4.45E-02) ≈	3.457E-01 (1.61E-02) −	3.492E-01 (1.69E-02) −	**2.995E-01** (2.10E-02)
	7	4.374E-01 (2.57E-02) −	4.552E-01 (1.47E-02) −	4.756E-01 (2.01E-02) −	**4.182E-01** (1.21E-02)
WFG5	3	6.327E-02 (1.10E-03) −	6.212E-02 (5.54E-04) −	6.118E-02 (7.03E-04) −	**4.730E-02** (7.89E-04)
	5	3.350E-01 (9.77E-03) −	3.077E-01 (6.36E-03) −	3.036E-01 (8.83E-03) −	**1.811E-01** (3.02E-03)
	7	4.101E-01 (2.08E-02) −	4.996E-01 (1.32E-02) −	5.024E-01 (1.38E-02) −	**3.206E-01** (8.04E-03)
WFG7	3	5.811E-02 (6.31E-04) −	5.930E-02 (7.32E-04) −	6.014E-02 (7.11E-04) −	**4.066E-02** (5.31E-04)
	5	3.572E-01 (5.47E-03) −	2.993E-01 (1.43E-02) −	3.207E-01 (1.71E-02) −	**1.858E-01** (2.12E-03)
	7	5.236E-01 (2.19E-02) −	4.576E-01 (2.38E-02) −	4.879E-01 (2.75E-02) −	**3.258E-01** (1.25E-02)
WFG8	3	8.646E-02 (3.44E-03) −	9.280E-02 (1.06E-03) −	9.612E-02 (1.48E-03) −	**7.901E-02** (1.19E-03)
	5	4.258E-01 (8.42E-03) −	3.969E-01 (1.26E-02) −	3.956E-01 (1.32E-02) −	**2.479E-01** (7.20E-03)
	7	5.816E-01 (1.30E-02) −	5.575E-01 (1.39E-02) −	5.642E-01 (1.38E-02) −	**4.127E-01** (5.93E-03)
WFG9	3	5.817E-02 (1.24E-03) −	5.628E-02 (7.29E-04) −	7.953E-02 (2.45E-02) −	**4.159E-02** (6.10E-04)
	5	3.633E-01 (1.20E-02) −	3.258E-01 (1.61E-02) −	3.396E-01 (1.55E-02) −	**1.832E-01** (7.10E-03)
	7	5.538E-01 (2.63E-02) −	5.115E-01 (2.15E-02) −	5.227E-01 (1.79E-02) −	**3.278E-01** (7.21E-03)
Test: +/−/≈		0/13/2	0/15/0	0/15/0	

[Xue et al., NeurIPS 2022 Spotlight]

Train on DTLZ2, WFG4, and WFG6 with m objectives, and test on the other problems with m objectives

Significantly better on almost all the 24 problems

Good generalization ability

Experiments – RQ2

Problem	M	MA-DAC (M)	MA-DAC (3)	MA-DAC (5)	MA-DAC (7)
DTLZ2	3	3.839E-02 (5.35E-04) +	**3.807E-02** (5.05E-04) +	3.830E-02 (7.24E-04) +	3.837E-02 (5.69E-04) +
	5	2.468E-01 (7.55E-03) +	2.472E-01 (1.56E-02) +	**2.442E-01** (1.26E-02) +	2.569E-01 (1.39E-02) +
	7	3.921E-01 (8.84E-03) +	**3.880E-01** (1.02E-02) +	4.081E-01 (1.52E-02) +	3.944E-01 (1.17E-02) +
WFG4	3	5.220E-02 (9.83E-04) +	**5.200E-02** (1.19E-03) +	5.236E-02 (1.10E-03) +	5.302E-02 (9.78E-04) +
	5	**1.850E-01** (3.14E-03) +	1.867E-01 (3.01E-03) +	1.868E-01 (2.81E-03) +	1.853E-01 (2.67E-03) +
	7	3.091E-01 (5.80E-03) +	3.104E-01 (7.14E-03) +	3.100E-01 (5.89E-03) +	**3.033E-01** (3.66E-03) +
WFG6	3	5.078E-02 (1.20E-02) +	4.831E-02 (8.95E-03) +	**4.599E-02** (9.48E-03) +	5.206E-02 (1.64E-02) +
	5	1.971E-01 (6.40E-03) +	2.003E-01 (6.26E-03) +	**1.942E-01** (6.90E-03) +	1.957E-01 (6.67E-03) +
	7	3.114E-01 (5.08E-03) +	3.242E-01 (9.24E-03) +	3.129E-01 (5.71E-03) +	**3.112E-01** (4.93E-03) +
DTLZ4	3	**6.171E-02** (3.67E-02) +	6.700E-02 (6.14E-02) +	6.618E-02 (4.62E-02) +	8.088E-02 (7.12E-02) +
	5	3.044E-01 (1.66E-02) ≈	**2.974E-01** (1.94E-02) +	2.995E-01 (2.10E-02) ≈	3.036E-01 (1.69E-02) ≈
	7	4.271E-01 (1.45E-02) +	4.313E-01 (1.39E-02) ≈	4.327E-01 (2.15E-02) ≈	**4.182E-01** (1.21E-02) +
WFG5	3	**4.721E-02** (7.15E-04) +	4.730E-02 (7.89E-04) +	4.733E-02 (8.10E-04) +	4.746E-02 (5.90E-04) +
	5	1.811E-01 (2.59E-03) +	1.817E-01 (2.96E-03) +	1.811E-01 (3.02E-03) +	**1.808E-01** (2.83E-03) +
	7	3.256E-01 (5.49E-03) +	3.266E-01 (8.98E-03) +	3.263E-01 (9.73E-03) +	**3.206E-01** (8.04E-03) +
WFG7	3	4.076E-02 (5.33E-04) +	**4.066E-02** (5.31E-04) +	4.077E-02 (5.12E-04) +	4.124E-02 (4.98E-04) +
	5	1.839E-01 (2.38E-03) +	1.881E-01 (3.70E-03) +	1.858E-01 (2.12E-03) +	**1.836E-01** (2.21E-03) +
	7	3.368E-01 (1.54E-02) +	3.461E-01 (1.97E-02) +	3.390E-01 (1.38E-02) +	**3.258E-01** (1.25E-02) +
WFG8	3	**7.828E-02** (1.46E-03) +	7.901E-02 (1.19E-03) +	7.921E-02 (1.36E-03) +	7.944E-02 (1.30E-03) +
	5	2.506E-01 (1.11E-02) +	2.653E-01 (1.51E-02) +	**2.479E-01** (7.20E-03) +	2.532E-01 (9.28E-03) +
	7	4.303E-01 (1.49E-02) +	4.364E-01 (1.38E-02) +	4.242E-01 (9.08E-03) +	**4.127E-01** (5.93E-03) +
WFG9	3	4.324E-02 (7.07E-04) +	**4.159E-02** (6.10E-04) +	4.359E-02 (1.00E-02) +	6.415E-02 (2.64E-02) +
	5	1.858E-01 (7.63E-03) +	**1.814E-01** (4.59E-03) +	1.832E-01 (7.10E-03) +	1.918E-01 (1.13E-02) +
	7	3.328E-01 (1.02E-02) +	3.298E-01 (1.03E-02) +	3.307E-01 (1.37E-02) +	**3.278E-01** (7.21E-03) +
Test: average rank	3	1.4	1.8	2.8	4.0
	5	2.6	2.8	2.2	2.4
	7	2.6	3.4	3.0	1.0
Test: +/−/≈	3	5/0/0	5/0/0	5/0/0	5/0/0
	5	4/0/1	5/0/0	4/0/1	4/0/1
	7	5/0/0	4/0/1	4/0/1	5/0/0

Train on DTLZ2, WFG4, and WFG6, and test on all the other problems

Train on DTLZ2, WFG4, and WFG6 with m objectives, and test on all the other problems

MA-DAC (M) is robust

Good generalization ability

[Xue et al., NeurIPS 2022 Spotlight]

Experiments – RQ3

Problem	M	MA-DAC (M) w/o 1	MA-DAC (M) w/o 2	MA-DAC (M) w/o 3	MA-DAC (M) w/o 4	MA-DAC (M)
DTLZ2	3	4.656E-02 (3.80E-04) −	3.914E-02 (8.43E-04) −	3.935E-02 (6.72E-04) −	3.919E-02 (5.91E-04) −	**3.839E-02** (5.35E-04)
	5	3.086E-01 (7.24E-03) −	2.619E-01 (8.99E-03) −	2.503E-01 (1.30E-02) −	**2.433E-01** (1.59E-02) ≈	2.468E-01 (7.55E-03)
	7	4.970E-01 (1.26E-02) −	4.067E-01 (1.20E-02) −	4.003E-01 (1.19E-02) −	4.228E-01 (1.25E-02) −	**3.921E-01** (8.84E-03)
WFG4	3	7.222E-02 (1.93E-03) −	5.484E-02 (1.01E-03) −	5.410E-02 (8.85E-04) −	5.410E-02 (8.85E-04) −	**5.220E-02** (9.83E-04)
	5	2.868E-01 (1.01E-02) −	1.879E-01 (3.76E-03) ≈	**1.845E-01** (2.17E-03) +	1.846E-01 (2.39E-03) +	1.850E-01 (3.14E-03)
	7	3.758E-01 (1.33E-02) −	3.102E-01 (6.34E-03) −	**3.020E-01** (3.99E-03) ≈	3.032E-01 (4.32E-03) ≈	3.091E-01 (5.80E-03)
WFG6	3	6.864E-02 (8.14E-03) −	5.338E-02 (1.37E-02) −	6.543E-02 (1.69E-02) −	6.067E-02 (2.11E-02) ≈	**5.078E-02** (1.20E-02)
	5	3.480E-01 (1.34E-02) −	2.005E-01 (5.21E-03) −	1.996E-01 (6.51E-03) −	1.979E-01 (6.76E-03) −	**1.971E-01** (6.40E-03)
	7	4.784E-01 (3.37E-02) −	3.147E-01 (5.85E-03) −	3.162E-01 (6.15E-03) −	3.147E-01 (5.73E-03) −	**3.114E-01** (5.08E-03)
Train: +/−/≈		0/9/0	0/8/1	1/7/1	1/5/3	
DTLZ4	3	6.463E-02 (3.85E-02) −	6.242E-02 (4.07E-02) −	**4.496E-02** (2.45E-03) +	4.496E-02 (2.45E-03) +	6.171E-02 (3.67E-02)
	5	3.497E-01 (1.41E-02) −	3.061E-01 (2.12E-02) ≈	3.054E-01 (1.55E-02) ≈	3.130E-01 (1.81E-02) −	**3.044E-01** (1.66E-02)
	7	4.853E-01 (1.82E-02) −	4.275E-01 (1.90E-02) −	**4.208E-01** (1.52E-02) ≈	4.289E-01 (2.07E-02) −	4.271E-01 (1.45E-02)
WFG5	3	6.189E-02 (7.40E-04) −	4.827E-02 (6.31E-04) −	4.766E-02 (5.72E-04) ≈	4.766E-02 (5.72E-04) ≈	**4.721E-02** (7.15E-04)
	5	3.202E-01 (9.77E-03) −	1.821E-01 (2.73E-03) ≈	1.835E-01 (2.90E-03) −	1.822E-01 (2.31E-03) −	**1.811E-01** (2.59E-03)
	7	4.948E-01 (1.47E-02) −	3.290E-01 (8.87E-03) −	3.310E-01 (7.70E-03) −	3.261E-01 (9.26E-03) −	**3.256E-01** (5.49E-03)
WFG7	3	6.004E-02 (9.45E-04) −	4.250E-02 (5.82E-04) −	4.150E-02 (6.60E-04) −	4.150E-02 (6.60E-04) −	**4.076E-02** (5.33E-04)
	5	3.402E-01 (2.49E-02) −	1.873E-01 (3.67E-03) ≈	**1.826E-01** (2.60E-03) +	1.847E-01 (2.80E-03) ≈	1.839E-01 (2.38E-03)
	7	4.877E-01 (5.70E-02) −	3.393E-01 (1.23E-02) −	3.373E-01 (1.16E-02) −	3.377E-01 (1.59E-02) −	**3.368E-01** (1.54E-02)
WFG8	3	9.661E-02 (1.87E-03) −	8.374E-02 (1.70E-03) −	8.029E-02 (1.29E-03) −	8.029E-02 (1.29E-03) −	**7.828E-02** (1.46E-03)
	5	4.119E-01 (1.30E-02) −	2.695E-01 (1.49E-02) −	2.571E-01 (1.09E-02) −	2.632E-01 (9.96E-03) −	**2.506E-01** (1.11E-02)
	7	5.830E-01 (1.59E-02) −	4.345E-01 (9.27E-03) −	**4.260E-01** (8.71E-03) −	4.322E-01 (1.12E-02) −	4.303E-01 (1.49E-02)
WFG9	3	5.894E-02 (9.24E-04) −	**4.321E-02** (7.50E-04) −	4.650E-02 (1.40E-02) −	4.650E-02 (1.40E-02) −	4.324E-02 (7.07E-04)
	5	3.148E-01 (2.06E-02) −	1.875E-01 (5.14E-03) −	1.941E-01 (6.45E-03) −	1.865E-01 (9.02E-03) −	**1.858E-01** (7.63E-03)
	7	5.069E-01 (2.53E-02) −	3.433E-01 (1.35E-02) −	3.402E-01 (1.00E-02) −	3.368E-01 (1.05E-02) −	**3.328E-01** (1.02E-02)
Test: +/−/≈		0/15/0	0/12/3	2/10/3	1/12/2	

MA-DAC (M) w/o i denotes without tuning the i-th hyper-parameter

Necessary to tune each hyper-parameter

[Xue et al., NeurIPS 2022 Spotlight]

Experiments – RQ3

$$r_t^1 = \max\{f(s_t) - f(s_{t+1}), 0\},$$

$$r_t^2 = \begin{cases} 10 & \text{if } f(s_{t+1}) < f_t^* \\ 1 & \text{else if } f(s_{t+1}) < f(s_t) \\ 0 & \text{otherwise} \end{cases},$$

$$r_t^3 = \max\left\{\frac{f(s_t) - f(s_{t+1})}{f(s_{t+1}) - f_{\text{opt}}}, 0\right\},$$

$$r_t = \begin{cases} (1/2) \cdot (p_t^2 - p_{t-1}^2) & \text{if } f(s_{t+1}) < f_t^* \\ 0 & \text{otherwise} \end{cases}$$

Our proposed reward function is better

Problem	M	MA-DAC-R1	MA-DAC-R2	MA-DAC-R3	MA-DAC
DTLZ2	3	4.223E-02 (2.50E-03) −	3.853E-02 (5.58E-04) −	3.809E-02 (4.64E-04) ≈	**3.807E-02** (5.05E-04)
	5	2.401E-01 (8.27E-03) −	2.726E-01 (1.51E-02) ≈	**2.364E-01** (1.04E-02) +	2.442E-01 (1.26E-02)
	7	4.142E-01 (1.12E-02) −	4.248E-01 (1.30E-02) −	4.215E-01 (9.03E-03) −	**3.944E-01** (1.17E-02)
WFG4	3	5.989E-02 (5.60E-03) ≈	5.255E-02 (1.14E-03) −	5.309E-02 (8.02E-04) −	**5.200E-02** (1.19E-03)
	5	1.848E-01 (2.61E-03) +	1.851E-01 (2.43E-03) +	**1.846E-01** (2.20E-03) +	1.868E-01 (2.81E-03)
	7	3.028E-01 (3.19E-03) +	**3.008E-01** (3.51E-03) ≈	3.029E-01 (3.36E-03) ≈	3.033E-01 (3.66E-03)
WFG6	3	7.920E-02 (1.81E-02) +	4.909E-02 (1.50E-02) −	**4.814E-02** (1.22E-02) ≈	4.831E-02 (8.95E-03)
	5	1.977E-01 (6.17E-03) −	2.037E-01 (4.49E-03) −	1.975E-01 (5.78E-03) −	**1.942E-01** (6.90E-03)
	7	**3.110E-01** (4.86E-03) −	3.151E-01 (5.01E-03) ≈	3.148E-01 (4.05E-03) −	3.112E-01 (4.93E-03)
Train: average rank		2.67	3.11	2.11	2.11
Train: +/−/≈		3/5/1	1/5/3	2/4/3	
DTLZ4	3	**5.567E-02** (7.33E-03) −	7.236E-02 (6.19E-02) −	6.144E-02 (5.10E-02) ≈	6.700E-02 (6.14E-02)
	5	3.119E-01 (1.91E-02) −	3.221E-01 (2.12E-02) −	3.119E-01 (1.58E-02) −	**2.995E-01** (2.10E-02)
	7	4.354E-01 (1.29E-02) −	4.385E-01 (1.23E-02) −	4.275E-01 (1.60E-02) −	**4.182E-01** (1.21E-02)
WFG5	3	4.841E-02 (7.78E-04) −	4.763E-02 (7.73E-04) −	4.773E-02 (6.58E-04) −	**4.730E-02** (7.89E-04)
	5	1.823E-01 (2.49E-03) ≈	1.818E-01 (2.90E-03) −	1.812E-01 (3.06E-03) ≈	**1.811E-01** (3.02E-03)
	7	3.212E-01 (6.60E-03) ≈	**3.174E-01** (6.43E-03) ≈	3.196E-01 (5.99E-03) ≈	3.206E-01 (8.04E-03)
WFG7	3	4.555E-02 (1.26E-03) ≈	4.076E-02 (5.41E-04) −	4.168E-02 (6.40E-04) −	**4.066E-02** (5.31E-04)
	5	1.842E-01 (3.28E-03) ≈	1.865E-01 (2.93E-03) +	**1.841E-01** (3.95E-03) +	1.858E-01 (2.12E-03)
	7	3.335E-01 (1.09E-02) +	**3.199E-01** (9.86E-03) −	3.271E-01 (9.65E-03) ≈	3.258E-01 (1.25E-02)
WFG8	3	8.914E-02 (2.96E-03) ≈	7.911E-02 (1.06E-03) −	8.199E-02 (1.96E-03) −	**7.901E-02** (1.19E-03)
	5	2.551E-01 (1.02E-02) −	2.628E-01 (1.22E-02) −	2.541E-01 (9.08E-03) −	**2.479E-01** (7.20E-03)
	7	4.163E-01 (9.54E-03) ≈	**4.115E-01** (9.80E-03) ≈	4.197E-01 (7.52E-03) −	4.127E-01 (5.93E-03)
WFG9	3	5.003E-02 (9.00E-03) −	4.208E-02 (6.56E-04) −	4.428E-02 (9.97E-03) −	**4.159E-02** (6.10E-04)
	5	1.929E-01 (8.84E-03) ≈	**1.819E-01** (5.73E-03) −	1.951E-01 (9.83E-03) −	1.832E-01 (7.10E-03)
	7	3.342E-01 (8.56E-03) −	3.322E-01 (8.89E-03) −	3.327E-01 (8.02E-03) −	**3.278E-01** (7.21E-03)
Test: average rank		3.27	2.47	2.67	1.60
Test: +/−/≈		1/7/7	1/12/2	1/10/4	

[Xue et al., NeurIPS 2022 Spotlight]

Experiments – RQ3

Problem	M	DQN-1	VDN	IQL	QMIX
DTLZ2	3	4.100E-02 (7.64E-04)	**3.807E-02** (5.05E-04) +	3.933E-02 (4.29E-04) +	3.916E-02 (7.21E-04) +
	5	2.408E-01 (1.03E-02)	2.442E-01 (1.26E-02) ≈	**2.310E-01** (9.29E-03) +	2.419E-01 (1.37E-02) ≈
	7	4.064E-01 (1.14E-02)	**3.944E-01** (1.17E-02) +	4.138E-01 (9.79E-03) −	4.162E-01 (1.59E-02) −
WFG4	3	6.136E-02 (1.80E-03)	**5.200E-02** (1.19E-03) +	5.438E-02 (8.83E-04) +	5.206E-02 (1.16E-03) +
	5	1.887E-01 (2.15E-03)	1.868E-01 (2.81E-03) +	1.879E-01 (2.78E-03) ≈	**1.859E-01** (2.25E-03) +
	7	3.018E-01 (3.14E-03)	3.033E-01 (3.66E-03) ≈	3.046E-01 (3.55E-03) −	**2.998E-01** (4.21E-03) +
WFG6	3	5.149E-02 (1.17E-02)	4.831E-02 (8.95E-03) +	5.592E-02 (1.57E-02) ≈	**4.542E-02** (3.02E-03) +
	5	1.991E-01 (9.10E-03)	**1.942E-01** (6.90E-03) +	1.981E-01 (6.76E-03) ≈	1.975E-01 (6.98E-03) ≈
	7	3.204E-01 (7.11E-03)	**3.112E-01** (4.93E-03) +	3.148E-01 (3.34E-03) +	3.128E-01 (7.39E-03) +
Train: +/−/≈			7/0/2	4/2/3	6/1/2
DTLZ4	3	**4.697E-02** (3.78E-03)	6.700E-02 (6.14E-02) ≈	6.328E-02 (4.48E-02) −	5.094E-02 (2.38E-03) −
	5	3.074E-01 (1.24E-02)	**2.995E-01** (2.10E-02) +	3.021E-01 (1.62E-02) ≈	3.013E-01 (1.80E-02) +
	7	4.263E-01 (1.28E-02)	**4.182E-01** (1.21E-02) +	4.323E-01 (1.43E-02) ≈	4.303E-01 (1.95E-02) ≈
WFG5	3	4.815E-02 (6.96E-04)	**4.730E-02** (7.89E-04) +	4.818E-02 (6.24E-04) ≈	4.736E-02 (7.49E-04) +
	5	1.833E-01 (3.37E-03)	**1.811E-01** (3.02E-03) +	1.812E-01 (2.21E-03) +	1.813E-01 (2.54E-03) +
	7	3.294E-01 (8.51E-03)	3.206E-01 (8.04E-03) +	**3.173E-01** (6.91E-03) +	3.175E-01 (7.19E-03) +
WFG7	3	4.624E-02 (6.74E-04)	**4.066E-02** (5.31E-04) +	4.313E-02 (7.79E-04) +	4.077E-02 (4.94E-04) +
	5	1.835E-01 (3.02E-03)	1.858E-01 (2.12E-03) −	1.832E-01 (2.34E-03) ≈	**1.825E-01** (3.16E-03) ≈
	7	3.274E-01 (1.15E-02)	3.258E-01 (1.25E-02) ≈	**3.219E-01** (1.09E-02) +	3.226E-01 (1.12E-02) +
WFG8	3	8.658E-02 (1.30E-03)	**7.901E-02** (1.19E-03) +	8.652E-02 (2.75E-03) ≈	7.909E-02 (1.60E-03) +
	5	2.519E-01 (8.57E-03)	**2.479E-01** (7.20E-03) +	2.544E-01 (8.10E-03) ≈	2.496E-01 (9.83E-03) ≈
	7	4.163E-01 (7.50E-03)	4.127E-01 (5.93E-03) +	4.175E-01 (7.32E-03) ≈	**4.006E-01** (9.42E-03) +
WFG9	3	7.747E-02 (2.46E-02)	**4.159E-02** (6.10E-04) +	4.423E-02 (7.08E-04) +	4.167E-02 (5.92E-04) +
	5	2.041E-01 (5.21E-03)	**1.832E-01** (7.10E-03) +	1.915E-01 (8.87E-03) +	1.921E-01 (6.43E-03) +
	7	3.418E-01 (1.18E-02)	**3.278E-01** (7.21E-03) +	3.322E-01 (8.41E-03) +	3.298E-01 (8.46E-03) +
Test: +/−/≈			12/1/2	7/1/7	11/1/3

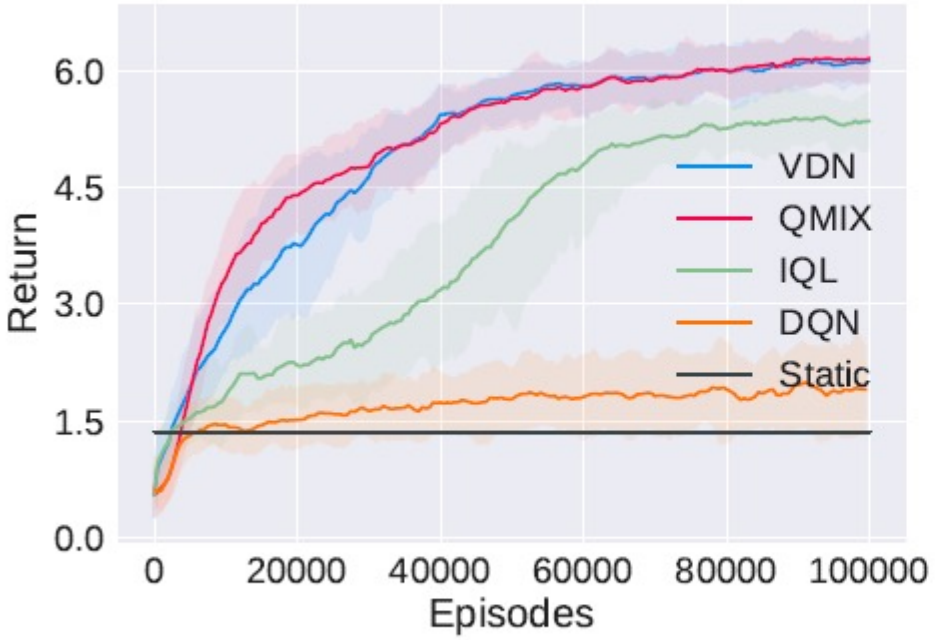
MA-DAC using different MARL algorithms

Significantly better than DQN-1 in most cases

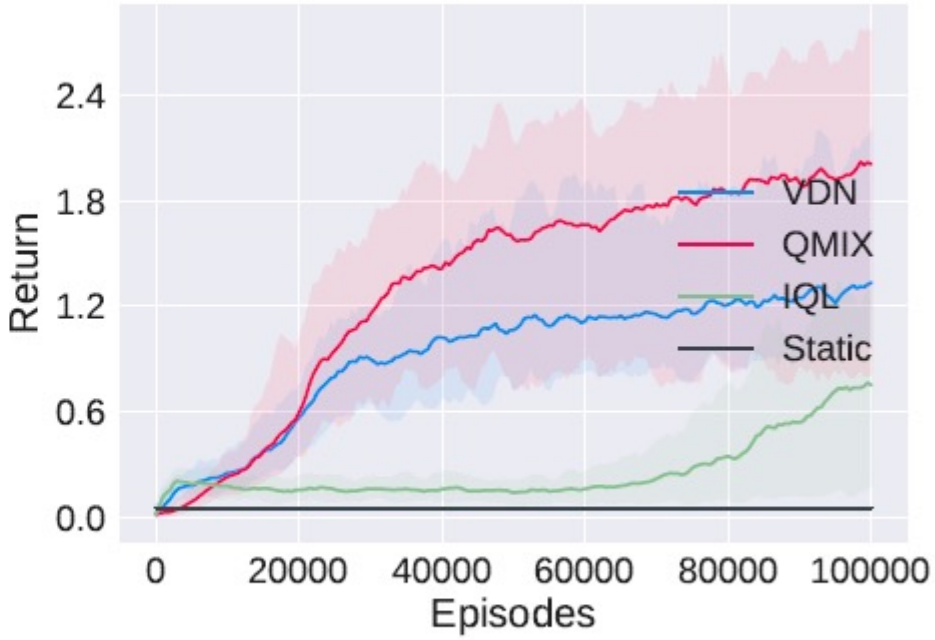
[Xue et al., NeurIPS 2022 Spotlight]

Experiments – Additional Results

Which MARL algorithm should we use?



(a) $5D$-Sigmoid



(b) $10D$-Sigmoid

Experiment on DACBench [Eimer et al., IJCAI 2021]

Dynamic Algorithm Configuration

Frank Hutter

关注

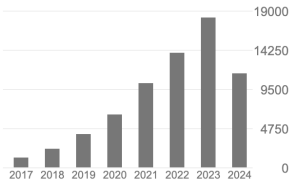
ELLIS Institute & Professor, University of Freiburg, Germany
在 cs.uni-freiburg.de 的电子邮件经过验证 - 首页

AutoML Meta-Learning Neural Architecture Search Deep Learning Machine Learning

标题	引用次数	年份
Decoupled weight decay regularization I Loshchilov, F Hutter ICLR 2019	19722	2017
Sgdr: Stochastic gradient descent with warm restarts I Loshchilov, F Hutter ICLR 2017	8112	2016

引用次数 查看全部

	总计	2019 年至今
引用	72969	64584
h 指数	87	77
i10 指数	194	183



Automated Dynamic Algorithm Configuration

Steven Adriaensen ADRIAENS@CS.UNI-FREIBURG.DE
André Biedenkapp BIEDENKA@CS.UNI-FREIBURG.DE
Gresa Shala SHALAG@CS.UNI-FREIBURG.DE
Noor Awad AWAD@CS.UNI-FREIBURG.DE
University of Freiburg, Machine Learning Lab

Theresa Eimer EIMER@TNT.UNI-HANNOVER.DE
Marius Lindauer LINDAUER@TNT.UNI-HANNOVER.DE
Leibniz University Hannover, Institute for Information Processing

Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg, Machine Learning Lab & Bosch Center for Artificial Intelligence

[Adriaensen et al., JAIR 2023]

7.2 Limitations and Further Research

While these case studies and other previous applications provide a "proof of concept" for automated DAC, we point out that much remains to be done to unlock its full potential, and we hope that this work may serve as a stepping stone for further exploring this promising line of research. In what remains, we will discuss some of the limitations of contemporary work and provide specific directions for future research.

Jointly configuring many parameters: While static approaches are capable of jointly configuring hundreds of parameters, the configuration space in contemporary DAC is typically much smaller, often considering only a single parameter. While the configuration space is smaller, the candidate solution space (i.e., the dynamic configuration policy space) grows exponentially with the number of reconfiguration points, in the worst case, and is thus typically drastically larger than static configuration policy spaces. Although modern techniques from reinforcement learning scale much better than ever before, we still know too little about the internal structure of DAC problems to handle this exploding space of possible policies. For example, not much is known regarding interaction effects of parameters in the DAC setting. If there should be only a few interaction effects between parameters as in static AC (Hutter et al., 2014; Wang et al., 2016), learning several independent policies might be a way forward.

Open Problem: How to adjust multiple heterogeneous hyper-parameters simultaneously? Done by MA-DAC

[Xue et al., NeurIPS 2022 Spotlight]

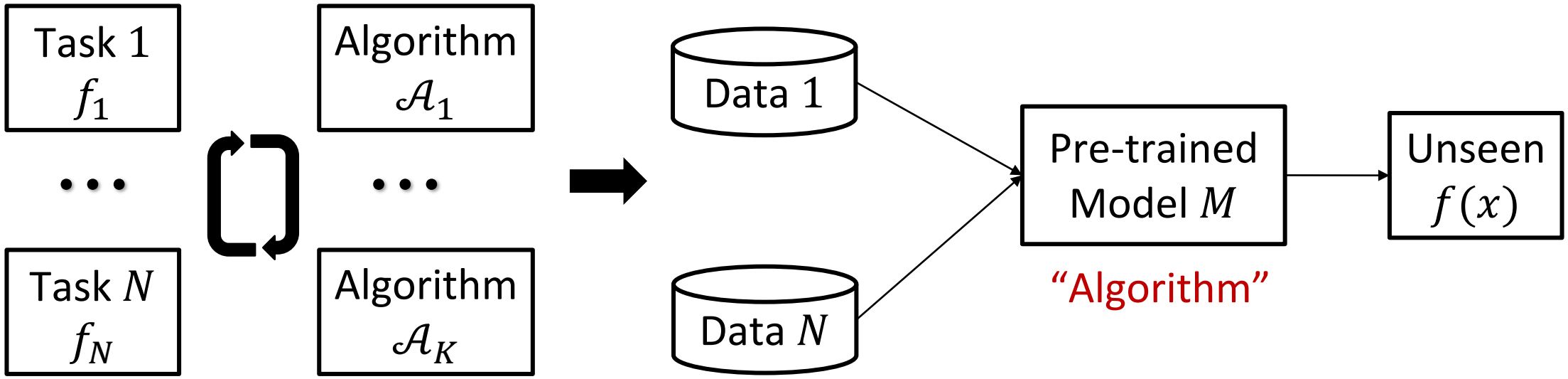
Learning to Optimize

- Can we learn to configure optimization algorithms dynamically?
  - Multi-agent dynamic algorithm configuration [Xue et al., NeurIPS'22 Spotlight]
- Can we learn to construct optimization algorithms directly?
- Can we learn to select proper optimization algorithms automatically?

End-to-end Learning for Black-box Optimization (BBO)

BBO: Optimize an objective function $f(x)$, with the only permission of querying $f(x)$

End-to-end learning for BBO: Utilize data from the task distribution $P(\mathcal{F})$ to pre-train a model M, which performs like an algorithm to optimize unseen objective functions

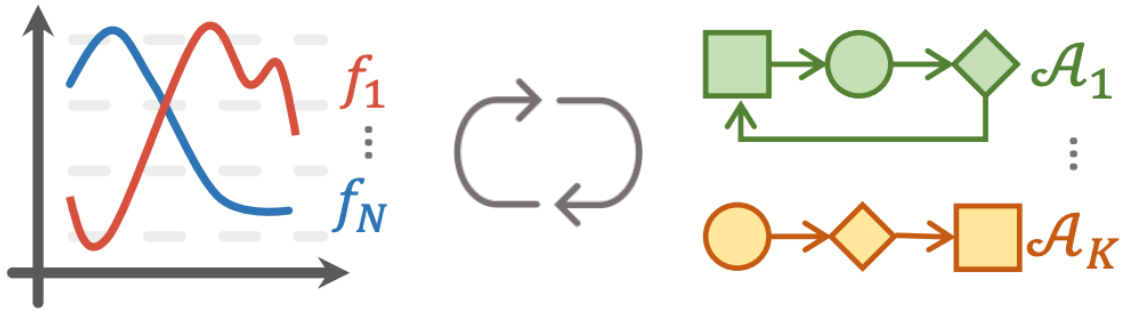


Reinforced In-context BBO

Source task: $f_i \sim P(\mathcal{F})$

Behavior algorithm: $\mathcal{A}_j, j = 1, \cdots K$

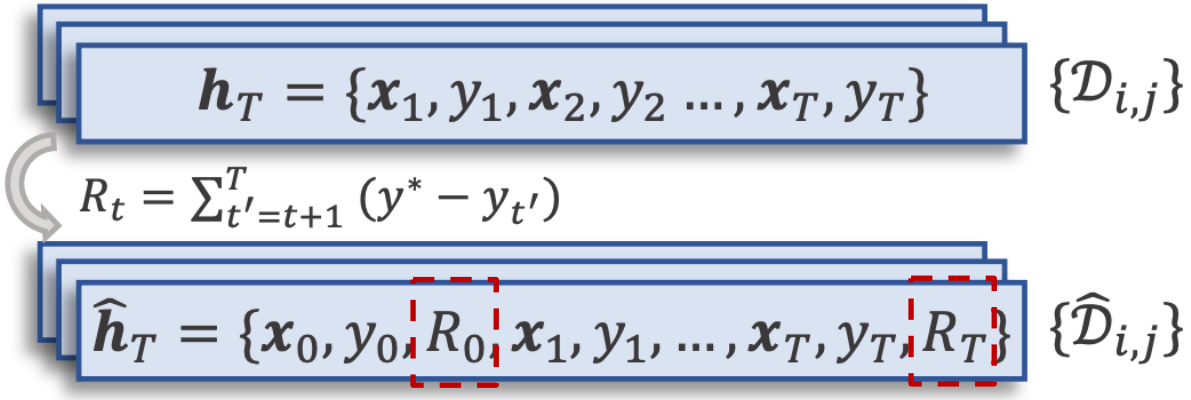
Offline dataset: $\mathcal{D}_{i,j} = \left\{ \boldsymbol{h}_T^{i,j,m} \right\}_{m=1}^{M}$



Collected by executing a behavior BBO algorithm $\mathcal{A}_j$ on f_i

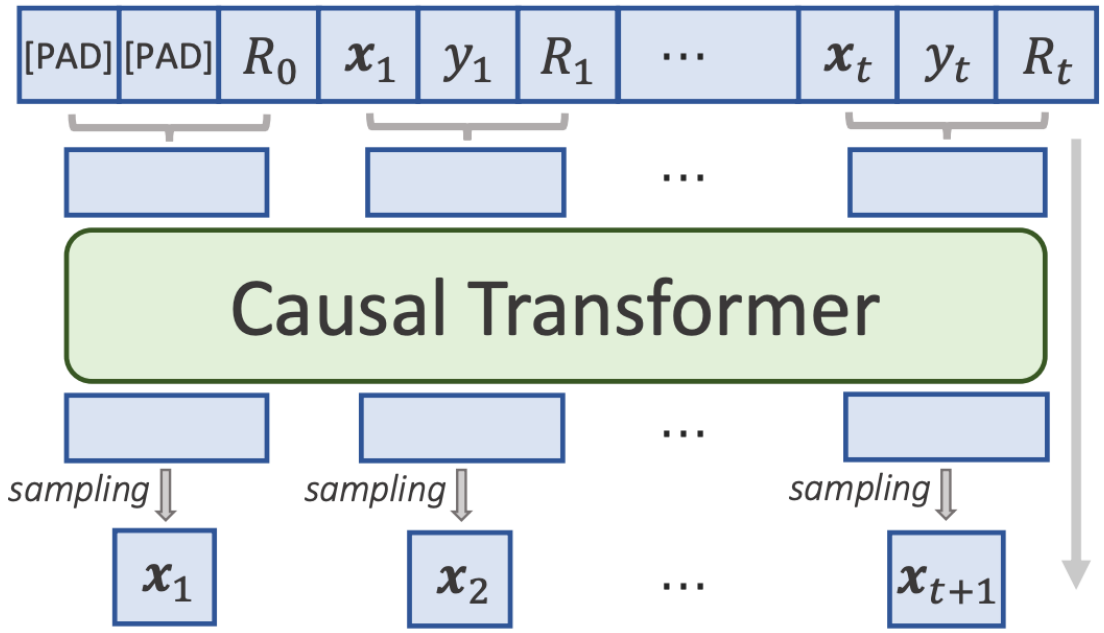
Augment histories by *regret-to-go* (RTG)

$$R_t = \sum_{t'=t+1}^{T} (y^* - y_{t'})$$



[Song et al., IJCAI'25]

Reinforced In-context BBO



(b) Training and Inference

$$\mathcal{L}_{RIBBO}(\theta) = -E_{\hat{h}_T \sim \mathcal{D}_{i,j}}\left[\sum_{t=1}^{T} \log M_\theta(x_t|\hat{h}_{t-1})\right]$$

Augmented Histories

$$R_t = \sum_{t'=t+1}^{T}(y^* - y_{t'})$$

Bring identifiability of algorithms and help generate user-desired algorithms automatically

Naïve RTG update strategy for inference $R_t = R_{t-1} - (y^* - y_t)$ Fall below 0

[Song et al., IJCAI'25]

Reinforced In-context BBO

Hindsight Regret Relabeling (HRR) for inference

Algorithm 1 Model Inference with HRR

Input: trained model $\mathcal{M}_{\boldsymbol{\theta}}$, budget T, optimum value y^*

Process:

1: Initialize context $\hat{\boldsymbol{h}}_0 = \{(\boldsymbol{x}_0, y_0, R_0)\}$, where $\boldsymbol{x}_0$ and y_0 are placeholders for padding and $R_0 = 0$;
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Generate the next query point $\boldsymbol{x}_t \sim \mathcal{M}_{\boldsymbol{\theta}}(\cdot|\hat{\boldsymbol{h}}_{t-1})$;
4: Evaluate $\boldsymbol{x}_t$ to obtain $y_t = f(\boldsymbol{x}_t)$;
5: Calculate the instantaneous regret $r = y^* - y_t$;
6: Relabel $R_i \leftarrow R_i + r$, for each $(\boldsymbol{x}_i, y_i, R_i)$ in $\hat{\boldsymbol{h}}_{t-1}$;
7: $\hat{\boldsymbol{h}}_t = \hat{\boldsymbol{h}}_{t-1} \cup \{(\boldsymbol{x}_t, y_t, 0)\}$;
8: **end for**

The immediate RTG is set as 0 to generate the most advantageous solutions

Previous RTG tokens are updated by adding the current regret

$$R_i = \sum_{t'=i+1}^{t}(y^* - y_{t'}) \qquad R_t = \sum_{t'=t+1}^{T}(y^* - y_{t'}) = 0 \Rightarrow R_i = \sum_{t'=i+1}^{T}(y^* - y_{t'})$$

[Song et al., IJCAI'25]

Experimental Settings

Behavior algorithms

- Heuristic search, e.g., random search, shuffled grid search, hill climbing
- Evolutionary algorithms, e.g., regularized evolution, eagle strategy, CMA-ES
- Bayesian optimization, e.g., GP-EI

Benchmarks

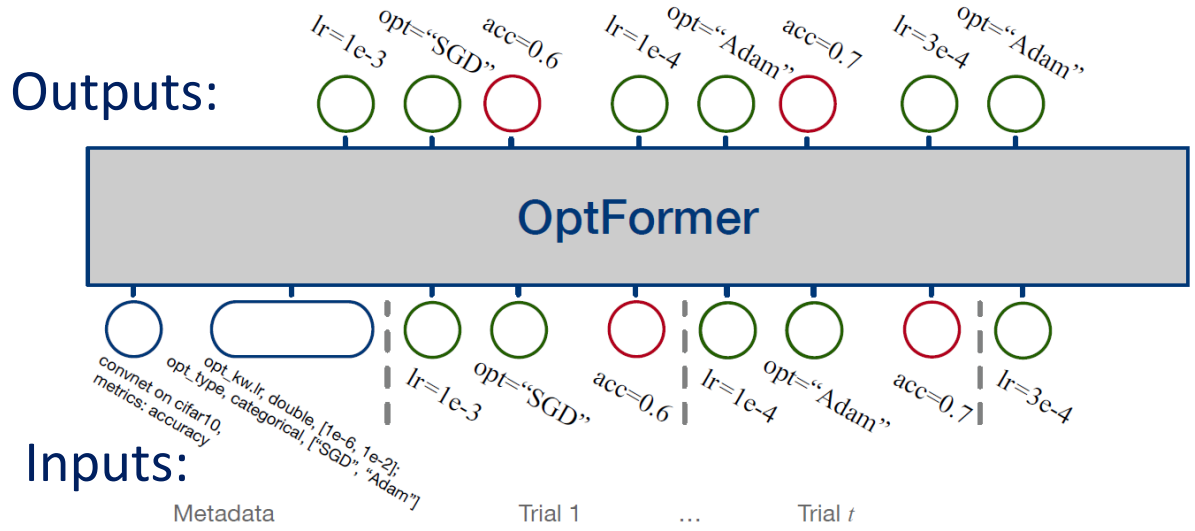
- BBOB functions [Elhara et al., 2019]
- HPO [Arango et al., 2021]
- Robot control problems [Wang et al., 2018]

A series of transformation are used to construct training and test data sets for BBOB and robot control problems, and a training and test split is provided by the authors for HPO

Experimental Settings

[Google, NeurIPS'22] makes the first attempt on learning end-to-end black-box optimizers with text-based Transformers

- Convert the metadata (description of the problem and algorithm) into text
- Convert the historical optimization trajectory of classical algorithms into text
- Train the OPTFormer to learn the converted trajectories from datasets



Training

$$\mathcal{L}(\theta; m, h) = \sum_n \log P_\theta(h^{(n)} | m, h^{(1:n-1)})$$

Inference

$$\pi(x_t | m, h_{t-1}) = \prod_{d=1}^{D} p_\theta(x_t^d | m, h_{t-1}, x_t^{(1:d-1)})$$

Experimental Settings

Baselines

- OptFormer [Google, NeurIPS'22]: Imitate the behavior of algorithms with an identifier of algorithm

Cannot select proper algorithms automatically

- Behavior cloning [Bain & Sammut, 1995]: Imitate the behavior of algorithms by optimizing

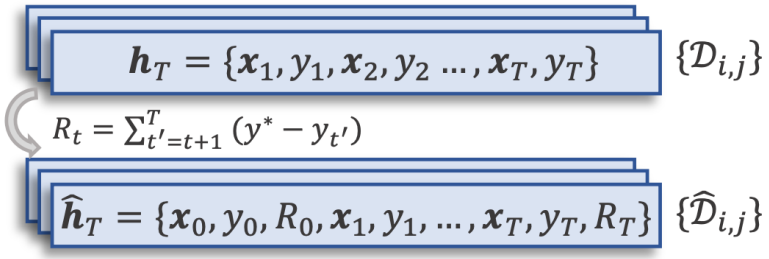
$$\mathcal{L}_{BC}(\theta) = -E_{h_T \sim \mathcal{D}_{i,j}} \left[\sum_{t=1}^{T} \log M_\theta(x_t | h_{t-1}) \right]$$

Average performance of behavior algorithms

Our method RIBBO:

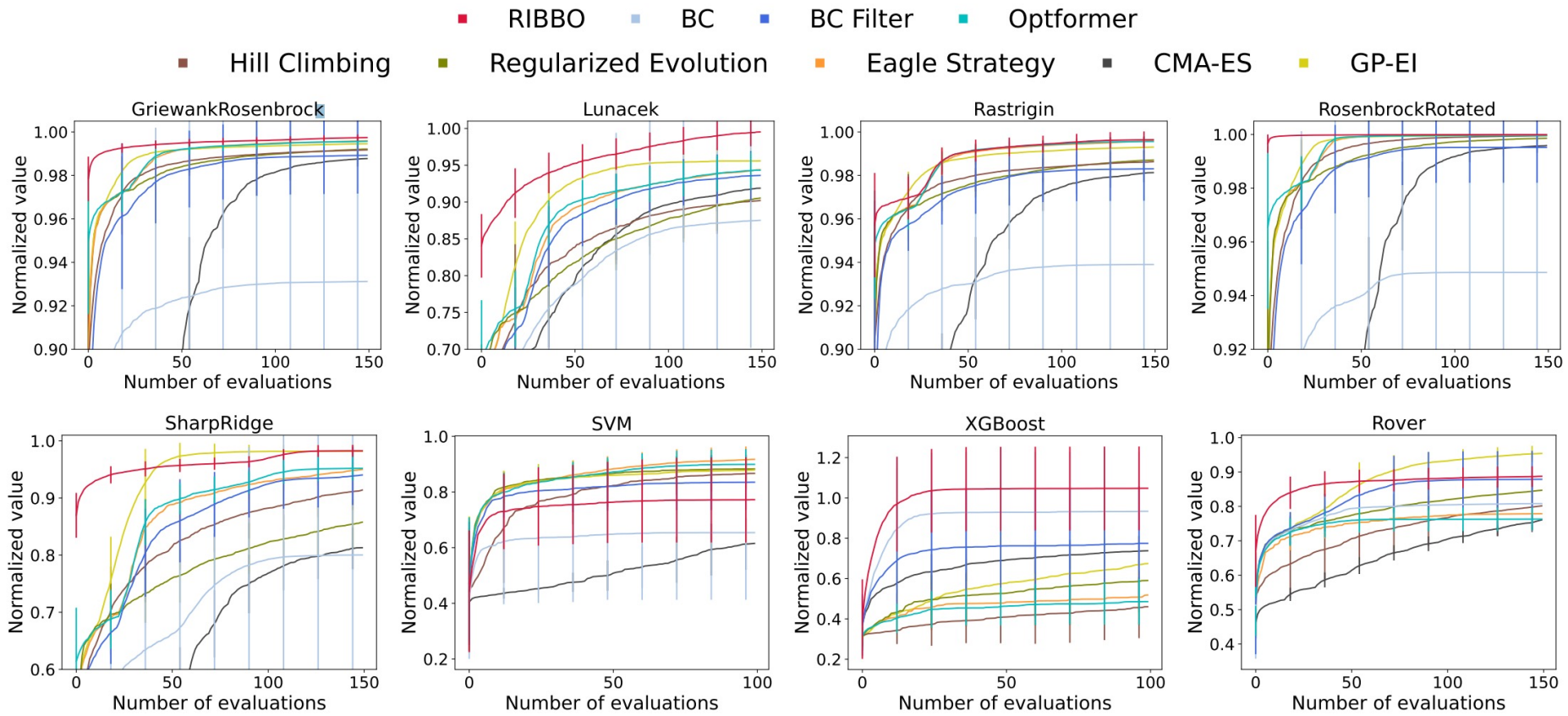
Augment histories by *regret-to-go*

$$R_t = \sum_{t'=t+1}^{T} (y^* - y_{t'})$$



Generate user-desired algorithms automatically

Experimental Results

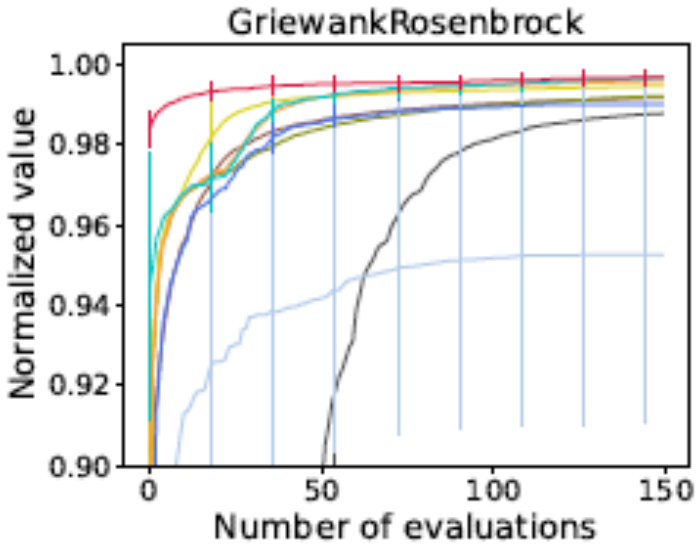


RIBBO outperforms the behavior algorithms and baselines

[Song et al., IJCAI'25]

Experimental Results

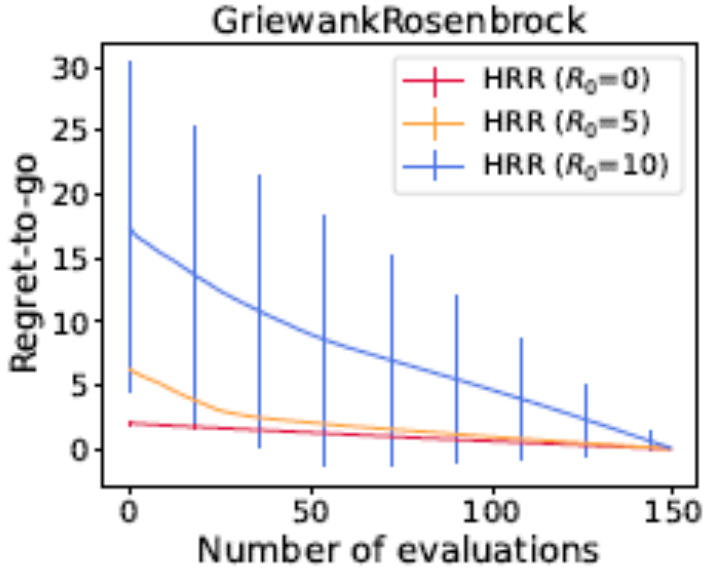
Cross-distribution generalization



RIBBO can generalize to unseen function distributions

train on 4 other function distributions and test on GriewankRosenbrock, which has different properties

Influence of initial RTG token



By incorporating RTG tokens into the optimization histories, RIBBO can automatically generate user-desired optimization trajectories

[Song et al., IJCAI'25]

Learning to Optimize

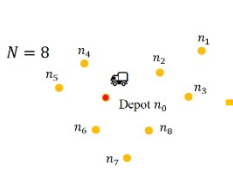
- Can we learn to configure optimization algorithms dynamically?
  - Multi-agent dynamic algorithm configuration [Xue et al., NeurIPS'22 Spotlight]
- Can we learn to construct optimization algorithms directly?
  - Pretrained seq-to-seq model for black-box optimization [Song et al., IJCAI'25]
- Can we learn to select proper optimization algorithms automatically?

End-to-end Learning for Combinatorial Optimization

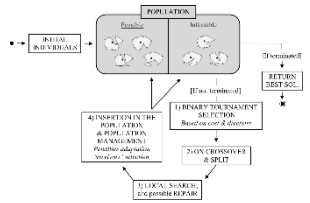
Traditional heuristic algorithms

Expert knowledge

Handcrafted

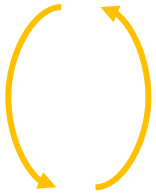


Instance



Algorithm

Low efficiency

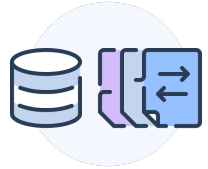


Iterations

Solutions

Learning to construct solutions

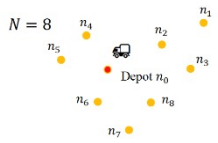
- An end-to-end learning way



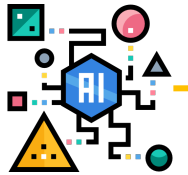
Offline datasets

Data-driven

Learning algorithms



Unseen Instance



Model

Extend

Partial Solutions

High efficiency

Complete Solutions

Generalization

"Approximate solutions can be found by leveraging problem special structure"

[Bengio et al., EJOR'21]

Vehicle Routing Problems (VRPs)

VRPs: a kind of canonical NP-hard combinatorial optimization problems, with broad applications in logistics and transportation

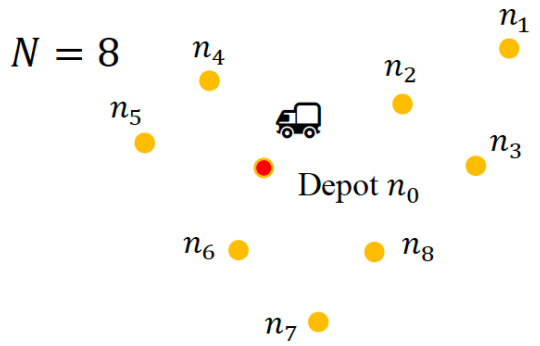
The goal: plan the routes and assign them to vehicles, aiming to minimize the total length while satisfying the demands of nodes and the limited capacity of vehicles

d_1=3

d_2=5

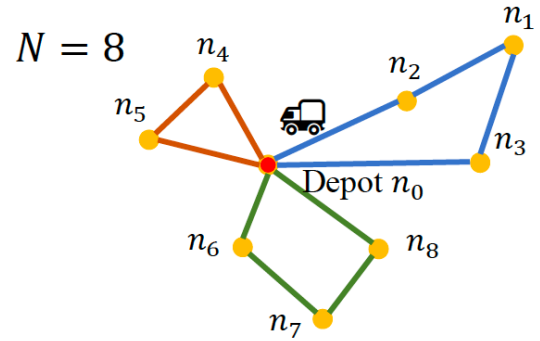
d_3=2

Capacity=10



Vehicle capacity: Q. Problem scale: N.
Node info: $\{n_i = (x_i, y_i, d_i), i = 1,2,..,N\}$.
Coordinates Demand

(a)



$\{n_0, n_2, n_1, n_3, n_0, n_4, n_5, n_0, n_6, n_7, n_8, n_0\}$

Route 1 Route 2 Route 3

(b)

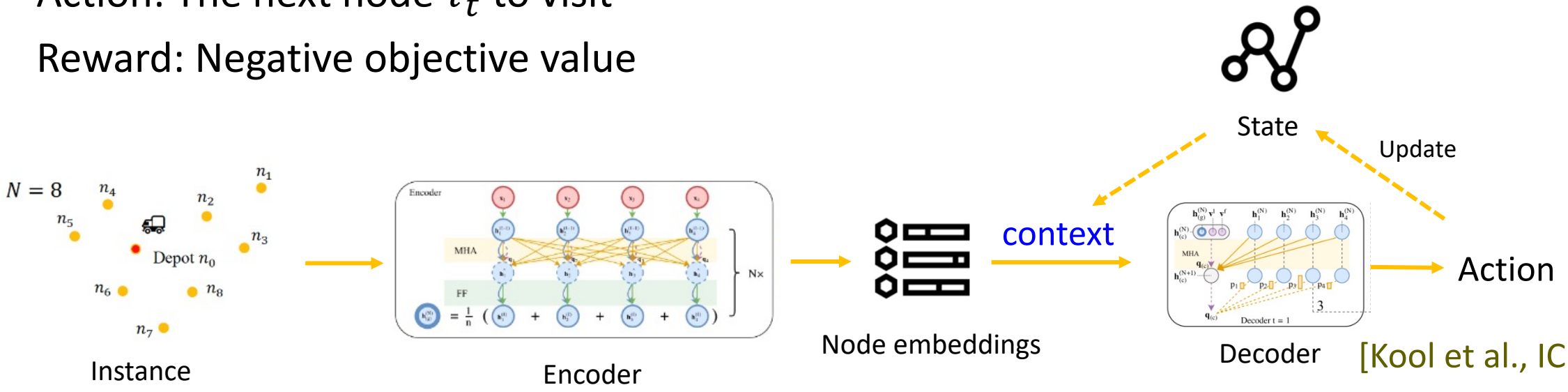
Learning to Construct Solutions for VRPs

A sequential solution can be naturally constructed in an autoregressive way

$$P(\boldsymbol{\tau}) = \Pi_t P(\tau_t \mid I, \{\tau_0, \tau_1, \ldots, \tau_{t-1}\})$$ [Vinyals et al., NeurIPS'15]

A common paradigm: Formulated as a Markov decision process and solved by RL

- State: The partial solution $\{\tau_0, \tau_1, \ldots, \tau_{t-1}\}$, always reduced to $\{\tau_0, \tau_{t-1}\}$
- Action: The next node τ_t to visit
- Reward: Negative objective value



Generalization Issues

Previous methods are usually trained on synthetic instances with specified distributions

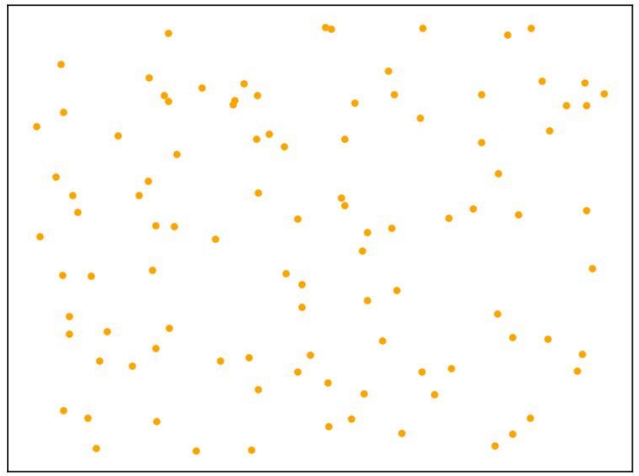
Training problem instances [Kool et al., ICLR'19]

- Small-scale ($N \leq 200$)
- Uniform node distribution

Real-world problem instances

- Large-scale with several thousand nodes
- Complex and unknown node distribution

Training problem instances



Small-scale and uniformly distributed

Hard to generalize

Real-world problem instances

Large-scale and complex

Generalization Issues

Training on TSP problems with 100 nodes and uniform distribution

Table 2: Experiment results on TSP

Method	TSP20 Len.	TSP20 Gap	TSP20 Time	TSP50 Len.	TSP50 Gap	TSP50 Time	TSP100 Len.	TSP100 Gap	TSP100 Time
Concorde	3.83	-	(5m)	5.69	-	(13m)	7.76	-	(1h)
LKH3	3.83	0.00%	(42s)	5.69	0.00%	(6m)	7.76	0.00%	(25m)
Gurobi	3.83	0.00%	(7s)	5.69	0.00%	(2m)	7.76	0.00%	(17m)
OR Tools	3.86	0.94%	(1m)	5.85	2.87%	(5m)	8.06	3.86%	(23m)
Farthest Insertion	3.92	2.36%	(1s)	6.00	5.53%	(2s)	8.35	7.59%	(7s)
GCN [9], beam search	3.83	0.01%	(12m)	5.69	0.01%	(18m)	7.87	1.39%	(40m)
Improv. [11], {5000}	3.83	0.00%	(1h)	5.70	0.20%	(1h)	7.87	1.42%	(2h)
Improv. [12], {2000}	3.83	0.00%	(15m)	5.70	0.12%	(29m)	7.83	0.87%	(41m)
AM [10], greedy	3.84	0.19%	($\ll$1s)	5.76	1.21%	(1s)	8.03	3.51%	(2s)
AM [10], sampling	3.83	0.07%	(1m)	5.71	0.39%	(5m)	7.92	1.98%	(22m)
POMO, single trajec.	3.83	0.12%	($\ll$1s)	5.73	0.64%	(1s)	7.84	1.07%	(2s)
POMO, no augment.	3.83	0.04%	($\ll$1s)	5.70	0.21%	(2s)	7.80	0.46%	(11s)
POMO, $\times$8 augment.	3.83	0.00%	(3s)	5.69	0.03%	(16s)	7.77	0.14%	(1m)

In-distribution: Gap < 1%

[Kwon et al., NeurIPS'20]

Cross-Size and Distribution Generalization (1K ins.)

Method	$(300, R)$ Obj. (Gap)	$(300, R)$ Time	$(300, E)$ Obj. (Gap)	$(300, E)$ Time	$(500, R)$ Obj. (Gap)	$(500, R)$ Time	$(500, E)$ Obj. (Gap)	$(500, E)$ Time	$(1000, R)$ Obj. (Gap)	$(1000, R)$ Time	$(1000, E)$ Obj. (Gap)	$(1000, E)$ Time
Concorde	9.79 (0.00%)	1.2m	9.48 (0.00%)	1.5m	12.39 (0.00%)	5.0m	11.73 (0.00%)	5.8m	17.09 (0.00%)	0.7h	15.66 (0.00%)	0.9h
LKH3	9.79 (0.00%)	6.0m	9.48 (0.00%)	6.8m	12.39 (0.00%)	11.8m	11.73 (0.00%)	13.8m	17.09 (0.00%)	0.4h	15.66 (0.00%)	0.5h
POMO	10.23 (4.43%)	1.5m	9.88 (4.20%)	1.5m	13.63 (10.00%)	6.0m	12.89 (9.88%)	6.0m	20.74 (21.38%)	0.8h	18.94 (20.97%)	0.8h
AMDKD-POMO	10.35 (5.69%)	1.5m	10.06 (6.15%)	1.5m	13.74 (10.85%)	6.0m	13.08 (11.52%)	6.0m	20.73 (21.25%)	0.8h	19.08 (21.85%)	0.8h

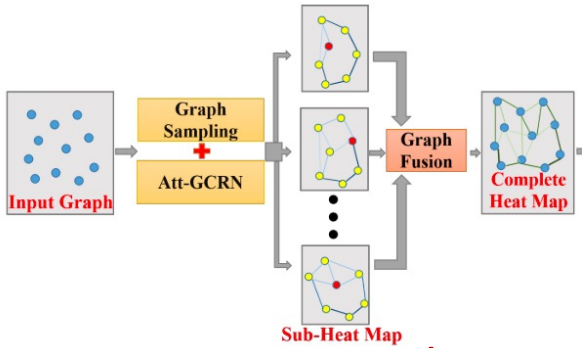
Cross-distribution and scale: Gap > 10%

[Zhou et al., ICML'23]

Recent Works on Improving Generalization

Divide and conquer

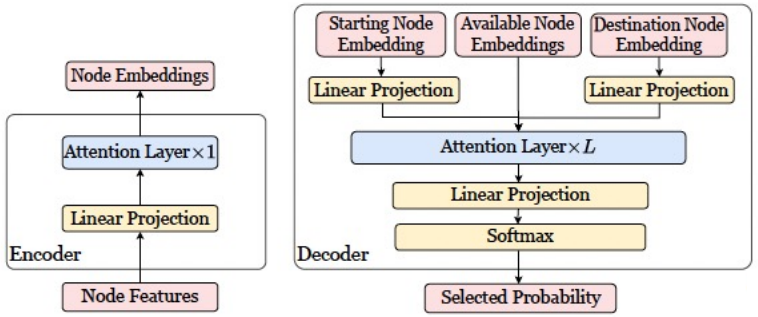
[Fu et al., AAAI'21]



Cross-scale

Heavy decoder

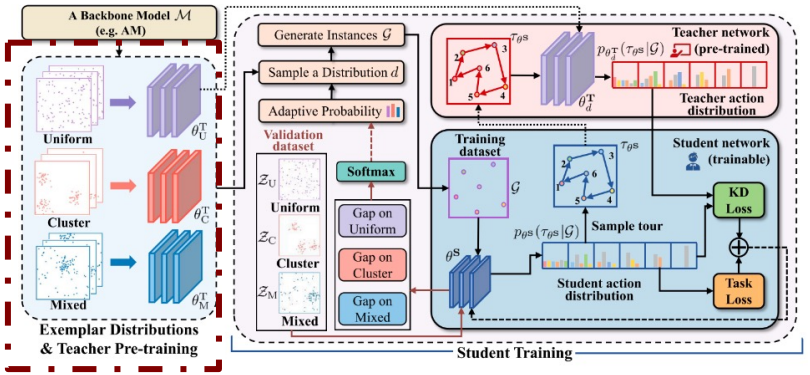
[Luo et al., NeurIPS'23]



Cross-scale

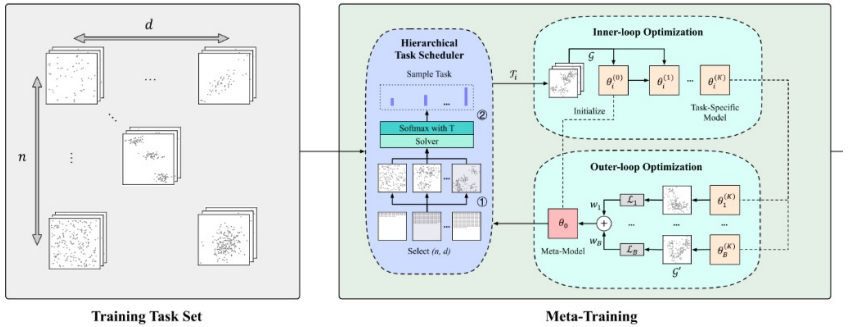
Knowledge distillation

[Bi et al., NeurIPS'22]



Cross-distribution

Meta learning [Zhou et al., ICML'23]



Cross-scale and Cross-distribution

Different from utilizing advanced learning methods,

We are to design

transferable features of the problem itself

Local Policy

Local topological features have great potential to be transferable

- Invariant to different problem scales
- More local similarity between varying node distributions

The optimal action (i.e., next node) is often included in local neighbor nodes

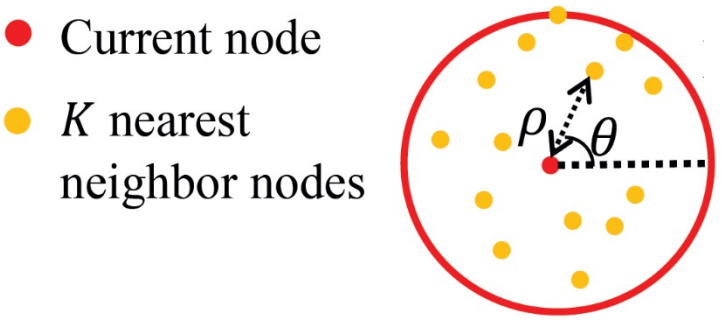
State space ⇩ Action space

Local subgraph

- K nearest neighbors of the current node and itself

$$\{(\rho_i, \theta_i) | i = 0,1, \dots, K)\}$$

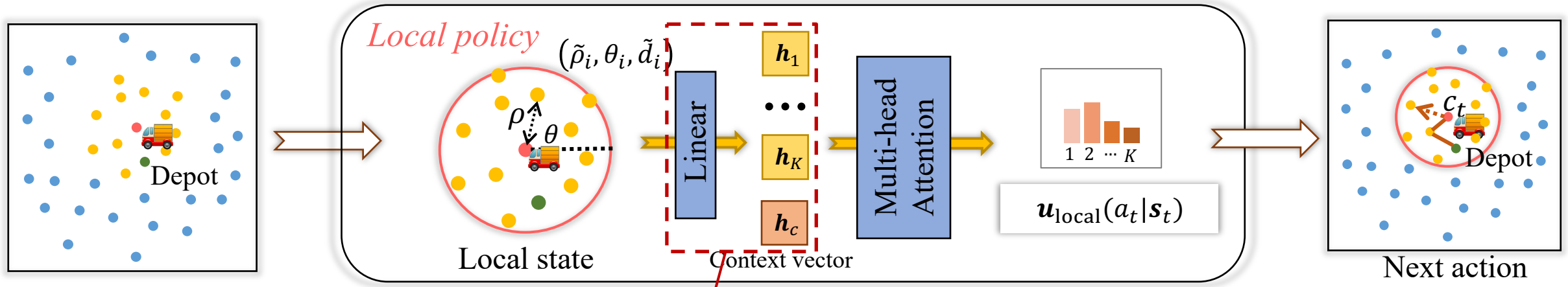
- Normalize to a unit ball: $\rho_i \leftarrow \rho_i / \rho_{\max}$



[Gao et al., IJCAI'24]

Local Policy

Attention-based architecture



Linear embedding layer

$$\boldsymbol{h}_i = W\boldsymbol{x}_i + b + \mathrm{PE}(\boldsymbol{x}_i), i \in \{1, \ldots, K\}$$

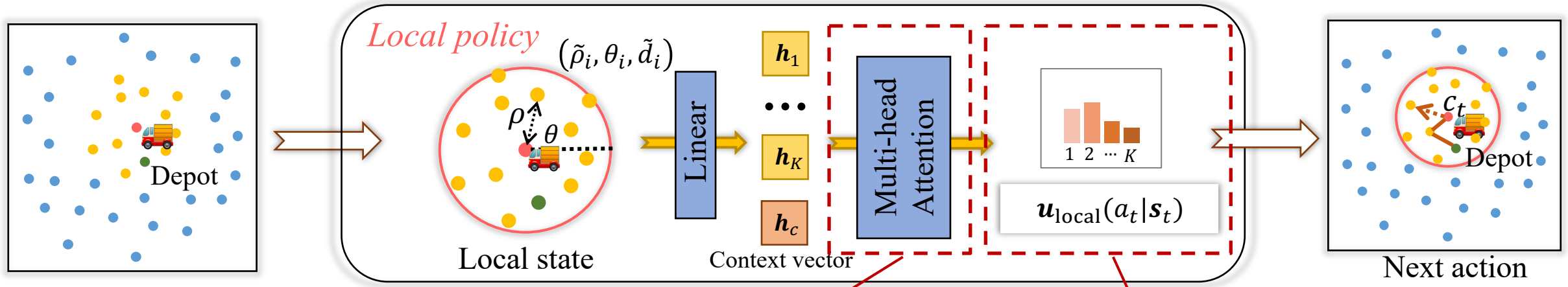
Positional encoding according to the distance ordering

$\boldsymbol{h}_c$: Learnable context vector for representing the current node

[Gao et al., IJCAI'24]

Local Policy

Attention-based architecture



Multi-head attention

$$\boldsymbol{k}_i = W^k \boldsymbol{h}_i, \boldsymbol{v}_i = W^v \boldsymbol{h}_i, \boldsymbol{q} = W^q \boldsymbol{h}_c,$$
$$\tilde{\boldsymbol{h}}_c = \mathrm{MHA}([\boldsymbol{k}_1, \dots, \boldsymbol{k}_K], [\boldsymbol{v}_1, \dots, \boldsymbol{v}_K], \boldsymbol{q})$$

Aggregating the neighborhood information to the context vector

Action scores

$$u^i_{\text{local}} = \begin{cases} \dfrac{\tilde{\boldsymbol{h}}_c^T \boldsymbol{h}_i}{\sqrt{d_e}}, & \text{if } n_i \in \mathcal{N}(c_t), \\ 0, & \text{otherwise.} \end{cases}$$

[Gao et al., IJCAI'24]

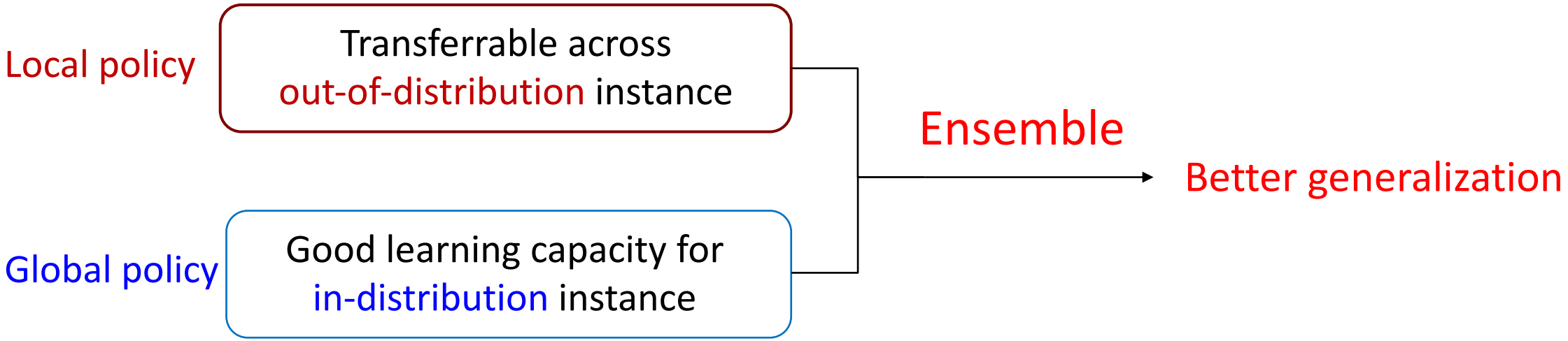
Ensemble of Local and Global Policies

Local policy

Reduces the state and action spaces to a local subgraph

Global policy, e.g., POMO [Kwon et al., NeurIPS'20]

Any typical constructive models that learn from the global information of complete VRP ins



[Gao et al., IJCAI'24]

Training of the Ensemble

Ensemble strategy

- Addition aggregation: The action score

$$\boldsymbol{u}_{\text{ens}} = \boldsymbol{u}_{\text{global}} + \boldsymbol{u}_{\text{local}}$$

$$\boldsymbol{u}^{i}_{\text{masked}} = \begin{cases} \text{C} \cdot \tanh\left(\boldsymbol{u}^{i}_{\text{ens}}\right), & \text{if action } a_i \text{ is valid.} \\ -\text{inf}, & \text{otherwise} \end{cases}$$

$$\boldsymbol{\pi}_{\text{ens}} = \text{softmax}(\boldsymbol{u}_{\text{masked}})$$

We use REINFORCE algorithm with shared baseline [Kwon et al., NeurIPS'20]

- Shared baseline: Average reward of multiple rollouts

$$\nabla L = \frac{1}{N \cdot B} \sum_{i=1}^{B} \sum_{j=1}^{N} \left(R_{i,j} - \frac{1}{N} \sum_{j=1}^{N} R_{i,j}\right) \nabla \log \boldsymbol{\pi}_{\text{ens}}(\tau_i, j)$$

[Gao et al., IJCAI'24]

Experimental Settings

Training data generator

- Node coordinates and demands are sampled from uniform distribution
- Fixed problem scale and vehicle capacity: $N = 100$ and $Q = 50$

Test on diverse problem instances with large scales and complex distributions

Baselines

- **POMO-based**: POMO [Kwon et al., NeurIPS'20], Sym-POMO [Kim et al., NeurIPS'22]
- **Cross-distribution methods**: AMDKD [Bi et al., NeurIPS'22], and Omni [Zhou et al., ICML'23]
- **Cross-scale methods**: Pointerformer [Jin et al., AAAI'23], LEHD [Luo et al., NeurIPS'23], and BQ [Drakulic et al., NeurIPS'23]
- **Diffusion-based**: DIFUSCO [Sun & Yang, NeurIPS'23] and T2TCO [Li et al., NeurIPS'23]

Experimental Results

On CVRPLib Set-X [Uchoa et al., EJOR 2017]: Synthetic complex instances with $N \in [100,1000]$

Non-learning heuristics ←

State-of-the-art constructive method ←

Method	(0, 200]	(200, 1000]	Total	Time
LKH3	0.36%	1.18%	1.00%	16m
HGS	0.01%	0.13%	0.11%	16m
HGS (less time)	0.21%	1.28%	1.59%	10s
POMO	5.26%	11.82%	10.37%	0.80s
Sym-POMO	9.99%	27.09%	23.32%	0.87s
Omni-POMO	5.04%	6.95%	6.52%	0.75s
LEHD	11.11%	12.73%	12.25%	1.67s
BQ	10.60%	10.97%	10.89%	3.36s
ELG-POMO (Ours)	**4.51%**	**6.46%**	**6.03%**	1.90s

Our ELG-POMO outperforms the SOTA constructive method

[Gao et al., IJCAI'24]

Experimental Results

On CVRPLib Set-XXL [Arnold et al., COR 2019]: Real-world instances with several thousand nodes

Method	A1	A2	L1	L2
POMO	112.27%	159.22%	75.30%	78.16%
Sym-POMO	79.72%	179.55%	170.44%	179.55%
Omni-POMO	42.52%	48.59%	22.79%	60.39%
LEHD	18.90%	26.40%	14.04%	26.30%
BQ	11.21%	**15.02%**	13.27%	24.00%
ELG-POMO (Ours)	**10.70%**	17.69%	**10.77%**	**21.80%**

Our ELG-POMO even works well on real-world instances with several thousand nodes

Experimental Results

On TSPLib [Reinelt, 1991]: various instances with $N < 1002$

Method	(0, 200]	(200, 1002]	Total	Time
LKH3	0.00%	0.00%	0.00%	24s
POMO	3.07%	13.35%	7.45%	0.41s
Sym-POMO	2.88%	15.35%	8.29%	0.34s
Omni-POMO	1.74%	7.47%	4.16%	0.34s
Pointerformer	2.31%	11.47%	6.32%	0.24s
LEHD	2.03%	3.12%	2.50%	1.28s
BQ	1.62%	**2.39%**	**2.22%**	2.85s
DIFUSCO	1.84%	10.83%	5.77%	30.68s
T2TCO	1.87%	9.72%	5.30%	30.82s
ELG-POMO (Ours)	**1.12%**	5.90%	3.08%	0.63s

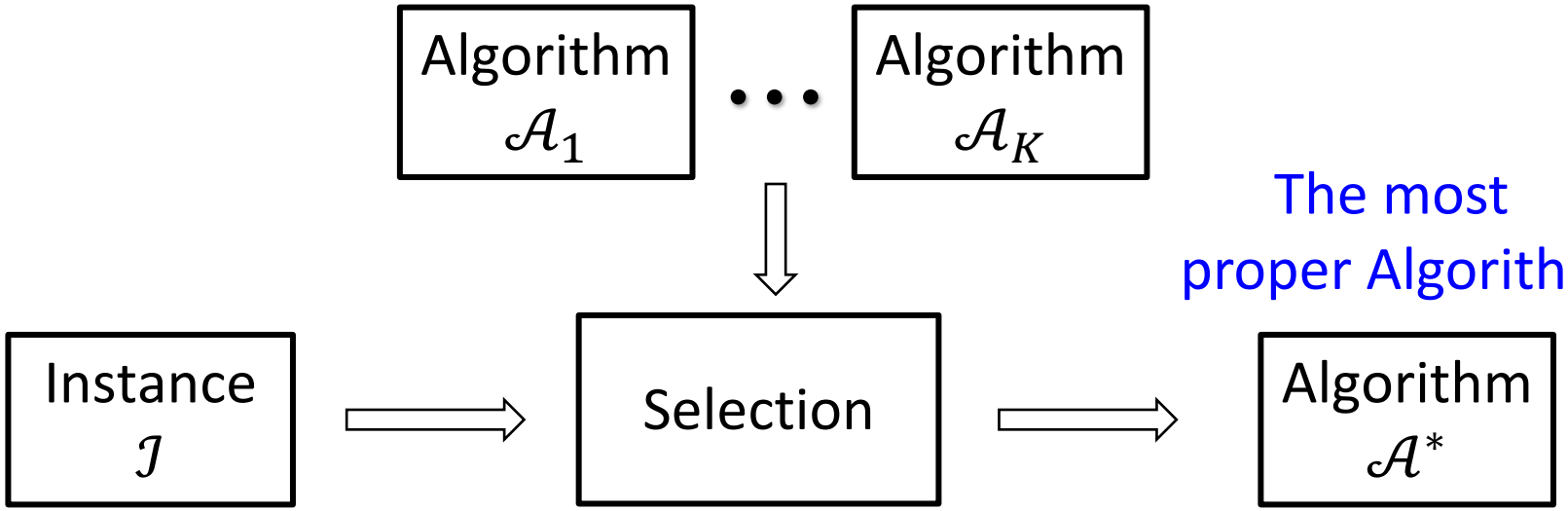
Heavy decoder can learn strong scale-invariant features

ELG-POMO performs the best on $N \in (0, 200]$, but worse than BQ and LEHD on $N > 200$

Learning to Optimize

- Can we learn to configure optimization algorithms dynamically?
  - Multi-agent dynamic algorithm configuration [Xue et al., NeurIPS'22 Spotlight]
- Can we learn to construct optimization algorithms directly?
  - Pretrained seq-to-seq model for black-box optimization [Song et al., IJCAI'25]
  - Generalizable neural solvers for vehicle routing problems [Gao et al., IJCAI'24]
- Can we learn to select proper optimization algorithms automatically?

Algorithm Selection



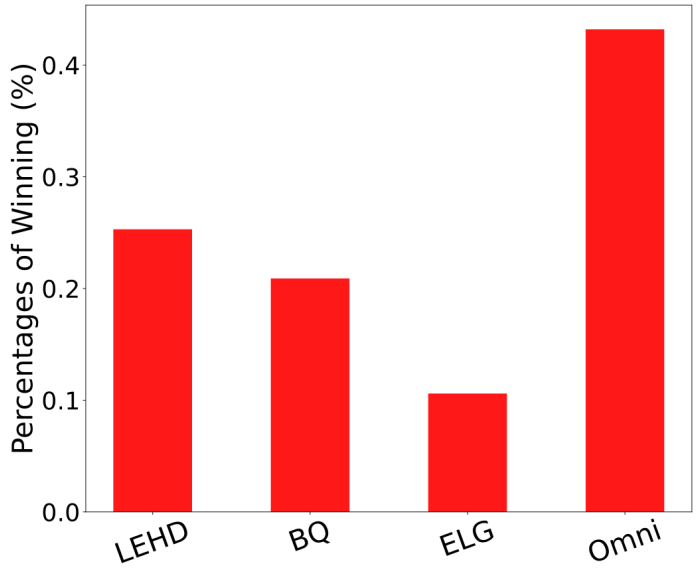
Algorithm selection has been applied in many fields, e.g., machine learning and black-box optimization

But it's new for neural combinatorial optimization!

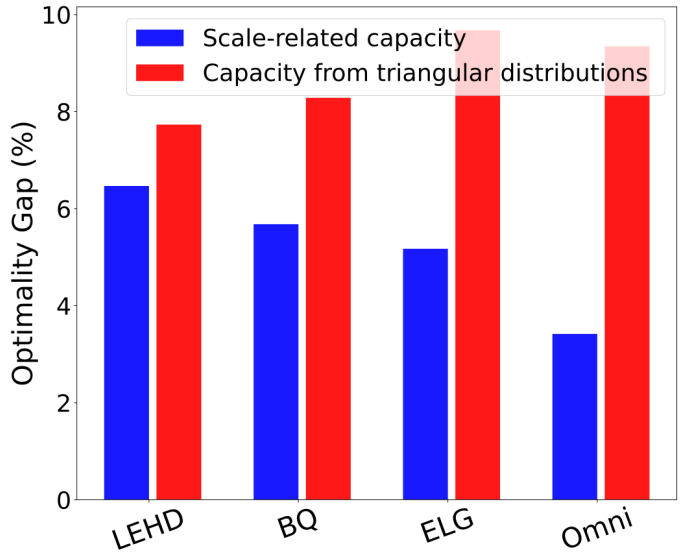
Diversity of Neural Combinatorial Optimization Solvers

Learning And Mining from DatA
http://www.lamda.nju.edu.cn

Different neural solvers demonstrate complementary performance at instance level



(a) Percentages of Winning



(b) Optimality Gap (Average)

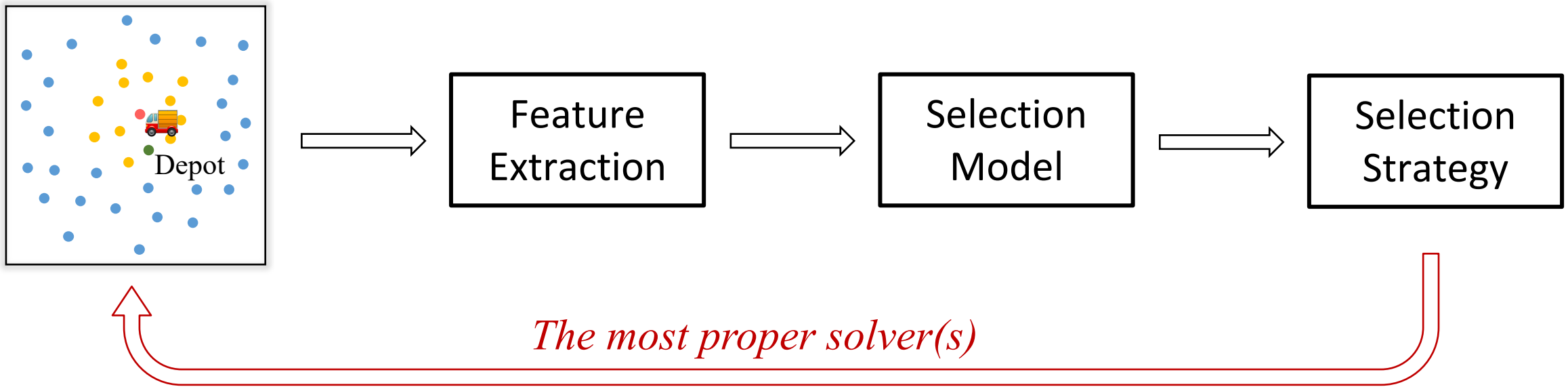
Different solvers win on different instances

The change of problem distribution almost reverses the rank of solvers

[Gao et al., ICML'25]

Neural Solver Selection for Combinatorial Optimization

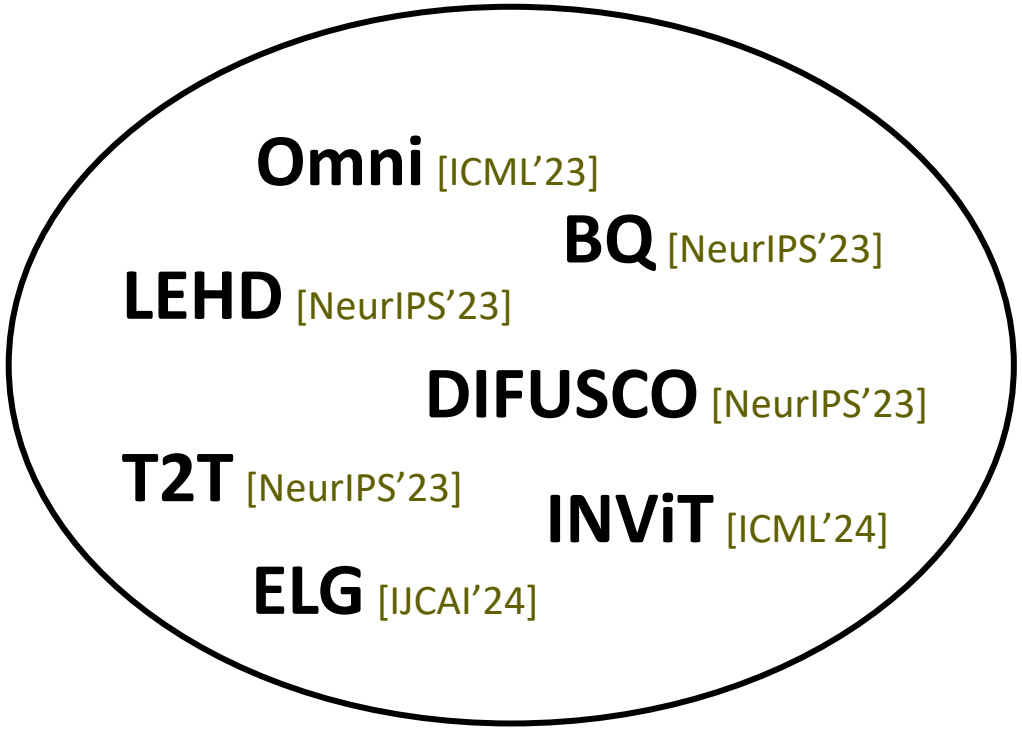
We introduce the idea of solver selection into the field of NCO for the first time



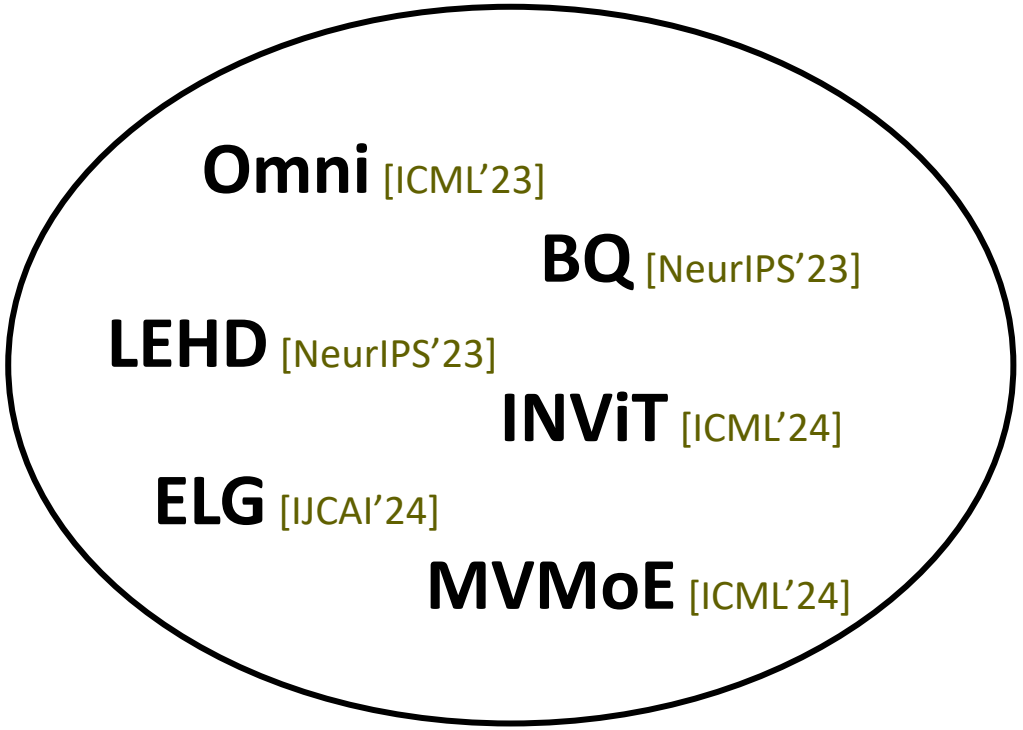
[Gao et al., ICML'25]

Application to TSP and CVRP

As an instantiation, we apply neural solver selection to two widely studied problems: Traveling Salesman Problem (TSP) and Capacitated Vehicle Routing Problem (CVRP)



Neural solver pool for TSP



Neural solver pool for CVRP

[Gao et al., ICML'25]

Application to TSP and CVRP

Feature extraction

A hierarchical graph encoder

- Adapt graph pooling to construct multi-level local views of problem instances
- Leverage the similarity of local views to boost OOD generalization

Selection model

A MLP to predict scores of neural solvers, trained by

- Classification: Treat the best solver as the ground truth label
- Ranking: Rank solvers according to their performance metrics

Selection strategies

- Greedy, top-k, rejection-based, and top-p selection

[Gao et al., ICML'25]

Experimental Results

Results on synthetic dataset with Gaussian distribution and $N \in [50,500]$

State-of-the-art individual solvers

Run all the solvers and select the best

Methods	TSP	
	Gap	Time
BQ (*3rd*)	3.00%	1.40s
T2T (*2nd*)	2.40%	1.58s
DIFUSCO (*1st*)	2.33%	1.45s
OPT	1.24%	8.93s
Selection by classification		
Greedy	1.94% (0.02%)	1.36s (0.01s)
Top-k ($k = 2$)	1.53% (0.01%)	2.52s (0.04s)
Rejection (20%)	1.81% (0.01%)	1.63s (0.01s)
Top-p ($p = 0.5$)	1.84% (0.03%)	1.55s (0.06s)
Selection by ranking		
Greedy	1.86% (0.01%)	1.33s (0.01s)
Top-k ($k = 2$)	**1.51% (0.02%)**	2.56s (0.03s)
Rejection (20%)	1.75% (0.02%)	1.63s (0.01s)
Top-p ($p = 0.5$)	1.68% (0.02%)	1.86s (0.07s)

Methods	CVRP	
	Gap	Time
LEHD (*3rd*)	7.37%	1.01s
BQ (*2nd*)	7.20%	1.59s
Omni (*1st*)	6.82%	0.24s
OPT	4.64%	4.38s
Selection by classification		
Greedy	5.35% (0.02%)	0.64s (0.01s)
Top-k ($k = 2$)	**4.81% (0.01%)**	1.87s (0.03s)
Rejection (20%)	5.19% (0.03%)	0.77s (0.01s)
Top-p ($p = 0.8$)	5.16% (0.03%)	0.87s (0.08s)
Selection by ranking		
Greedy	5.31% (0.01%)	0.62s (0.01s)
Top-k ($k = 2$)	4.82% (0.01%)	1.90s (0.04s)
Rejection (20%)	5.15% (0.02%)	0.74s (0.01s)
Top-p ($p = 0.8$)	4.99% (0.02%)	1.03s (0.03s)

Our method is close to OPT, but achieves significant speedup

Our method outperforms the best individual solvers on TSP and CVRP, and can even consume less time on TSP

[Gao et al., ICML'25]

Experimental Results

Generalization results on out-of-distribution datasets

- Training: Synthetic dataset with Gaussian distribution and $N \in [50,500]$
- Test: **TSPLIB/CVRPLIB** with more complex distributions and $N \in [50,1002]$

State-of-the-art single solvers

Run all the solvers and select the best

Methods	TSPLIB	
	Gap	Time
BQ (*3rd*)	3.04%	1.44s
DISFUCO (*2nd*)	2.13%	1.44s
T2T (*1st*)	1.95%	1.74s
OPT	0.89%	9.14s
Selection by classification		
Greedy	1.54% (0.05%)	1.33s (0.02s)
Top-k ($k = 2$)	1.22% (0.10%)	2.47s (0.02s)
Rejection (20%)	1.42% (0.11%)	1.54s (0.03s)
Top-p ($p = 0.5$)	1.49% (0.11%)	1.37s (0.02s)
Selection by ranking		
Greedy	1.33% (0.06%)	1.28s (0.03s)
Top-k ($k = 2$)	**1.07% (0.03%)**	2.48s (0.02s)
Rejection (20%)	1.26% (0.03%)	1.51s (0.04s)
Top-p ($p = 0.5$)	1.28% (0.04%)	1.46s (0.06s)

Methods	CVRPLIB Set-X	
	Gap	Time
BQ (*3rd*)	10.31%	2.60s
Omni (*2nd*)	6.21%	0.38s
ELG (*1st*)	6.10%	1.31s
OPT	5.10%	6.81s
Selection by classification		
Greedy	5.96% (0.12%)	1.06s (0.08s)
Top-k ($k = 2$)	5.44% (0.08%)	2.40s (0.25s)
Rejection (20%)	5.83% (0.12%)	1.31s (0.09s)
Top-p ($p = 0.8$)	5.79% (0.09%)	1.42s (0.17s)
Selection by ranking		
Greedy	5.76% (0.04%)	1.31s (0.10s)
Top-k ($k = 2$)	**5.39% (0.06%)**	2.56s (0.13s)
Rejection (20%)	5.63% (0.05%)	1.60s (0.08s)
Top-p ($p = 0.8$)	5.61% (0.03%)	1.72s (0.08s)

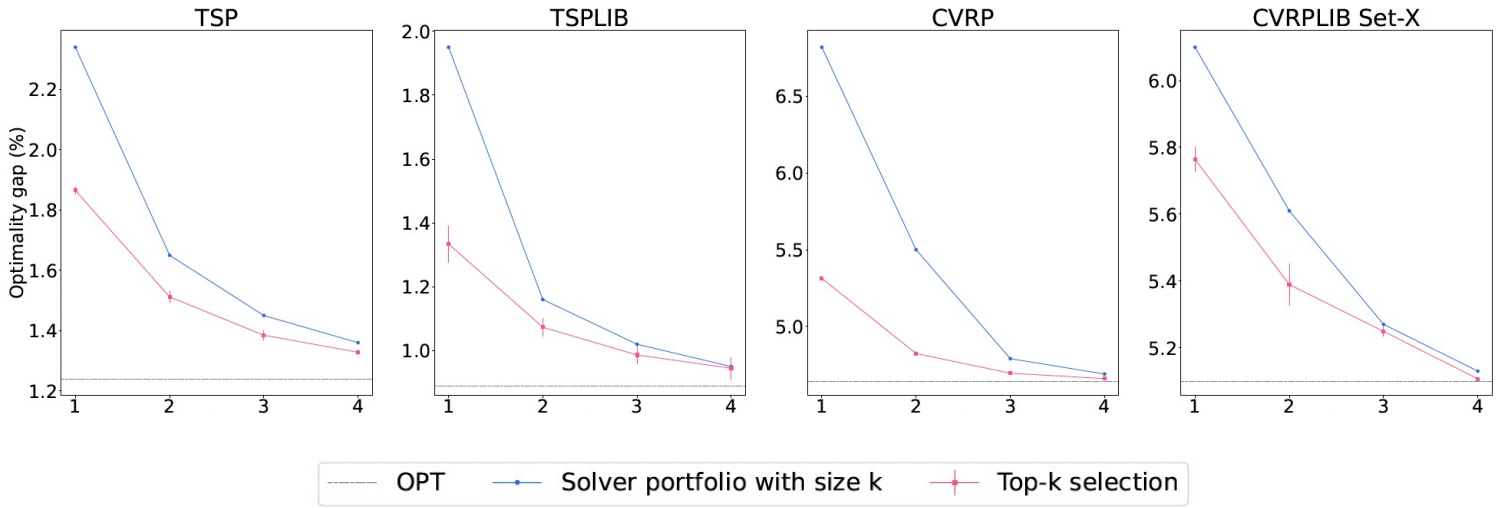
Our method is robust against the distribution shifts

[Gao et al., ICML'25]

Experimental Results

Comparison between top-k selection and solver portfolio with the same size

- Solver portfolio: The optimal solver subset obtained by enumeration



Our top-k selection consistently outperforms the size-k solver portfolio

We hope this work can open a new line for NCO. Many future works: Neural solver feature, runtime-aware selection, complementary solvers, etc.

[Gao et al., ICML'25]

Learning to Optimize

- Can we learn to configure optimization algorithms dynamically?
  - Multi-agent dynamic algorithm configuration [Xue et al., NeurIPS'22 Spotlight]
- Can we learn to construct optimization algorithms directly?
  - Pretrained seq-to-seq model for black-box optimization [Song et al., IJCAI'25]
  - Generalizable neural solvers for vehicle routing problems [Gao et al., IJCAI'24]
- Can we learn to select proper optimization algorithms automatically?
  - Neural solver selection for combinatorial optimization [Gao et al., ICML'25]

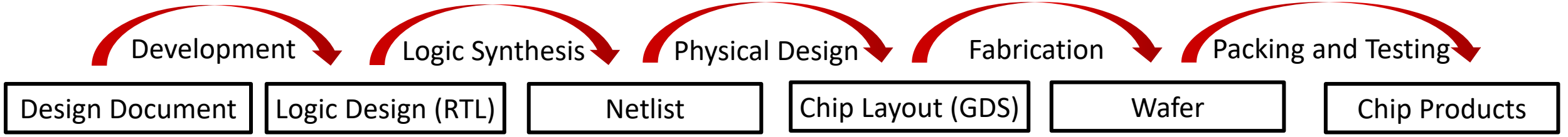
应用案例：芯片自动化设计

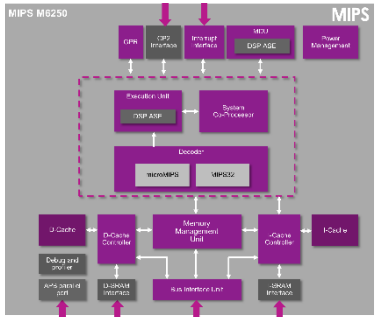
Function design and verification: Design the RTL and verify the functions. (Document -> RTL)

Logic synthesis: Mapping the RTL design into netlist. (RTL -> Netlist)

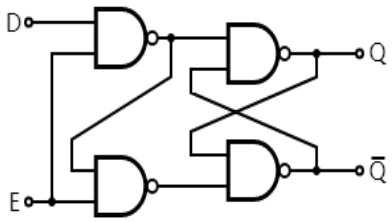
Physical design: Design the physical layout according to netlist by EDA tools. (Netlist -> GDS)

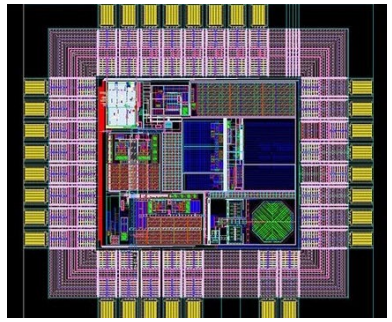
Chip manufacturing: Fabricate the chip from GDS layout by photolithography. (GDS -> Product)

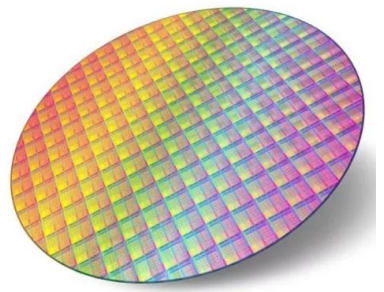


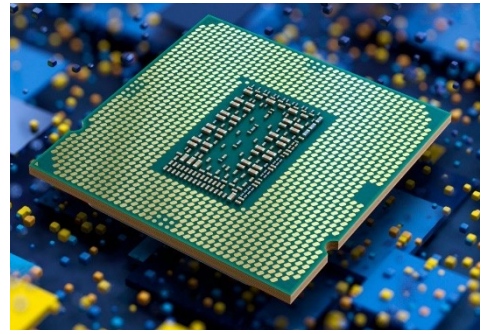



```
module alu32(Result, ALUOp, A, B, Zero) ;
  output [`ALULEN:0] Result;
  reg    [`ALULEN:0] Result;
  output Zero;
  reg    Zero;
  input  [2:0]  ALUOp;
  input  [`ALULEN:0] A, B;

  always @(A or B or ALUOp)
  begin
        case (ALUOp)
          3'b000: Result = A & B ;//and
          3'b001: Result = A | B ;//or
//       add your code here for addition, subtrac

        endcase

//      add your code here for Zero detect
  end
endmodule
```









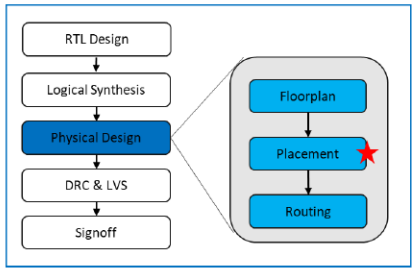
应用案例：芯片自动化设计

难题4：布局布线优化技术

出题组织：海思/诺亚方舟实验室　接口专家：许思源 xusiyuan520@huawei

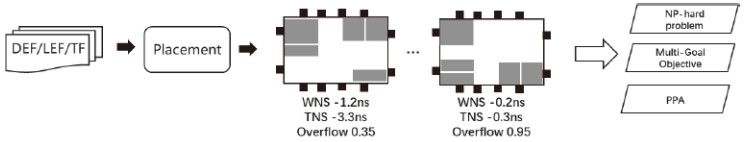
技术背景

芯片布局起着承上（逻辑综合）启下（布线）的作用，是现代超大规模集成电路设计物理设计流程中的一步，显著影响芯片设计的迭代周期。由于芯片布局被认为是NP-Complete的问题，因此如何快速生成高质量的芯片布局是布局问题的重要挑战。



Chip Design Flow　Auto Placement and Route (APR) Flow

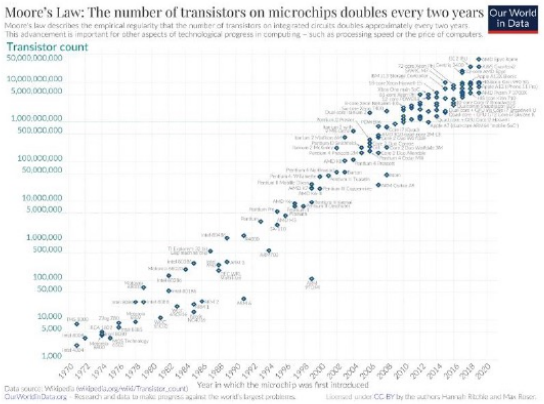
芯片布局问题可以描述为给定标准单元和宏单元的大小和连接关系，给出约束(如没有元件重叠)和优化目标(时序，拥塞，功耗，面积等)，基于特定策略确定每个单元的位置。



难题5：超高维空间多目标黑盒优化技术

出题组织：海思/诺亚方舟实验室　接口专家：胡守博 hushoubo@huawei.

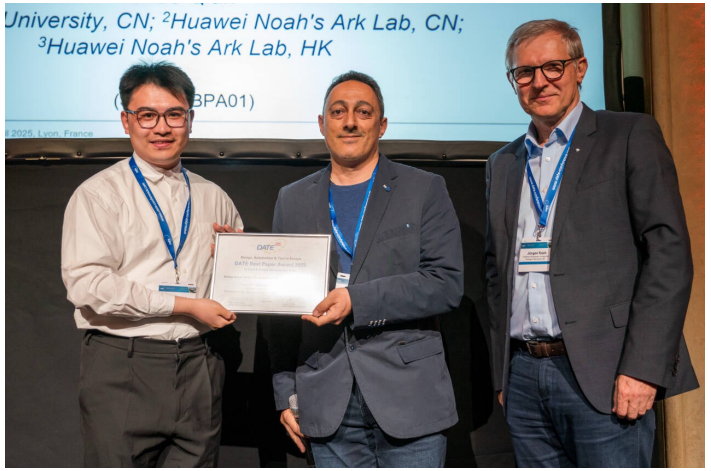
技术背景



[Reference: https://en.wikipedia.org/wiki/Moore%27s_law#/media/File:Moore's_Law_Transistor_Count_1970-2020.png]

- 随着芯片制程日益演进，芯片微架构设计空间呈现几何增长导致设计空间爆炸的问题，因此无法通过穷举的方式确定最优的架构空间参数配置。
- 传统的最优参数选取依靠架构师选取有潜力的参数配置进行仿真确定，但随着设计空间的膨胀，专家经验的局限性导致人工结果距最优配置差距越来越大。
- 现有自动化最优参数寻优方案主要依赖黑盒优化算法（如贝叶斯优化，MCTS 等），但实际任务中的超高维设计空间寻优仍未很好解决，是当前主要挑战之一。

应用案例：芯片元件布局



获 EDA 领域顶级国际会议 DATE’25

最佳论文奖

应用案例：芯片元件布局

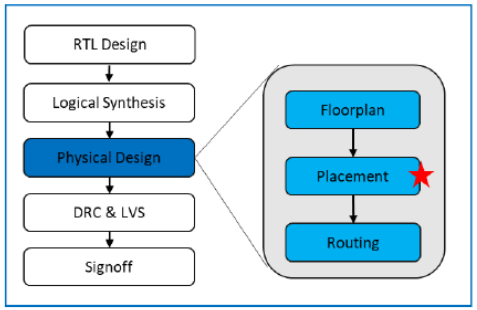
LAMDA
Learning And Mining from DatA
http://www.lamda.nju.edu.cn

难题4：布局布线优化技术

出题组织：海思/诺亚方舟实验室　　接口专家：许思源 xusiyuan520@huawei

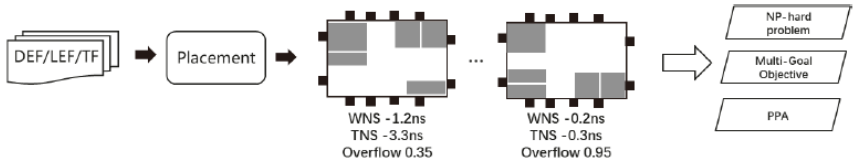
技术背景

芯片布局起着承上（逻辑综合）启下（布线）的作用，是现代超大规模集成电路设计物理设计流程中的一步，显著影响芯片设计的迭代周期。由于芯片布局被认为是NP-Complete的问题，因此如何快速生成高质量的芯片布局是布局问题的重要挑战。

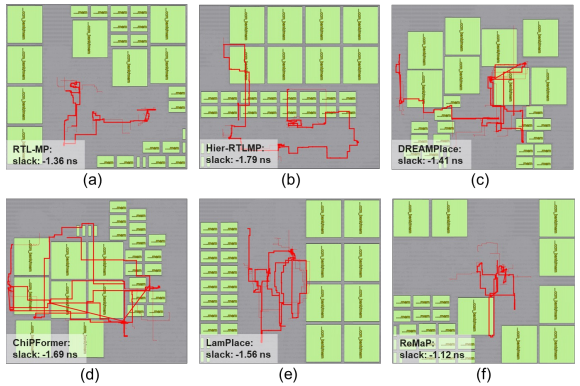


Chip Design Flow　　Auto Placement and Route (APR) Flow

芯片布局问题可以描述为给定标准单元和宏单元的大小和连接关系，给出约束(如没有元件重叠)和优化目标(时序，拥塞，功耗，面积等)，基于特定策略确定每个单元的位置。



在 6 个样例上对比SOTA方案平均提升 PPA 性能约 18.9%，平均提升拥塞指标约 46.3%，预测相关性（99%）



a) RTL-MP
b) Hier-RTLMP
c) DREAMPlace
d) ChiPFormer
e) LamPlace
f) ReMaP（钱超）

方案对比布局时序图

已集成进华为EDA工具

火花奖第115期

火花奖

难题期数	难题分类	难题	院校/部门	姓名
EDA专题第一期	EDA专题	布局布线优化技术	南京大学人工智能学院	钱超教授
算力会战第七期	算力会战	超长序列视频生成推理加速关键技术	浙江大学计算机科学与技术学院	李玺教授
算力会战第七期	算力会战	HPC应用骨架函数自动提取方法	浙江大学软件学院	常志豪研究员

应用案例：芯片寄存器寻优

对比算法为华为自研贝叶斯优化方法HeBO，曾获NeurIPS 2020黑箱优化比赛冠军

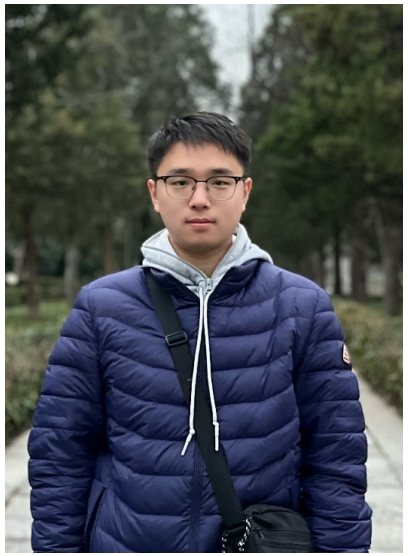
工业实际数据集

达到HeBO收敛目标值

任务	HeBO 收敛值	HeBO 运行轮数	提出方法 运行轮数	效率提升
mysql	393522	293	11	26.64倍
nginx	21768	198	4	49.50倍
redis	560446	175	49	3.57倍
unixbench	107	256	29	8.83倍

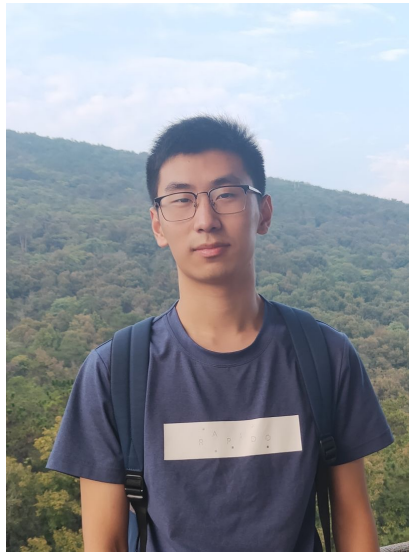
相较HeBO，优化效率平均提升22.14倍

Collaborators



Ke Xue
(PhD, 3rd year)

Lei Song
(Master, 3rd year)



Chengrui Gao
(PhD, 1st year)

Yunqi Shi
(Master, 2nd year)

Thank you!