南京大學

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

UNIVERSITY OF BIRMINGHAM

NEURAL INFORMATION PROCESSING SYSTEMS

Multi-agent Dynamic Algorithm Configuration

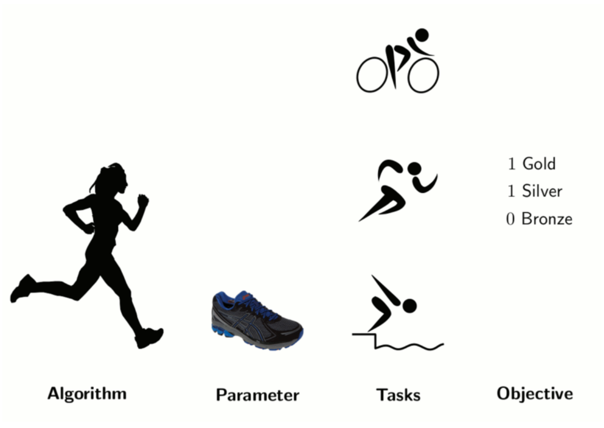
Ke Xue*, Jiacheng Xu*, Lei Yuan, Miqing Li,
Chao Qian†, Zongzhang Zhang, Yang Yu
(*Equal Contribution, †Corresponding author)

NeurIPS'22

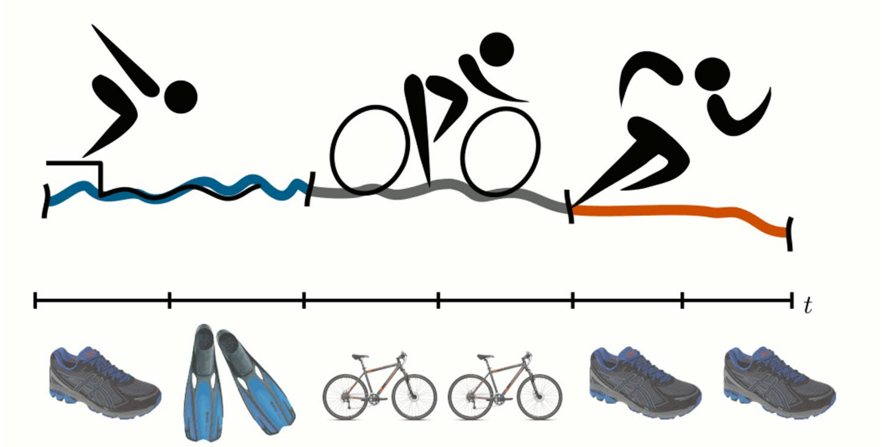
Email: {xuek, qianc}@lamda.nju.edu.cn
Nanjing University, China

Background

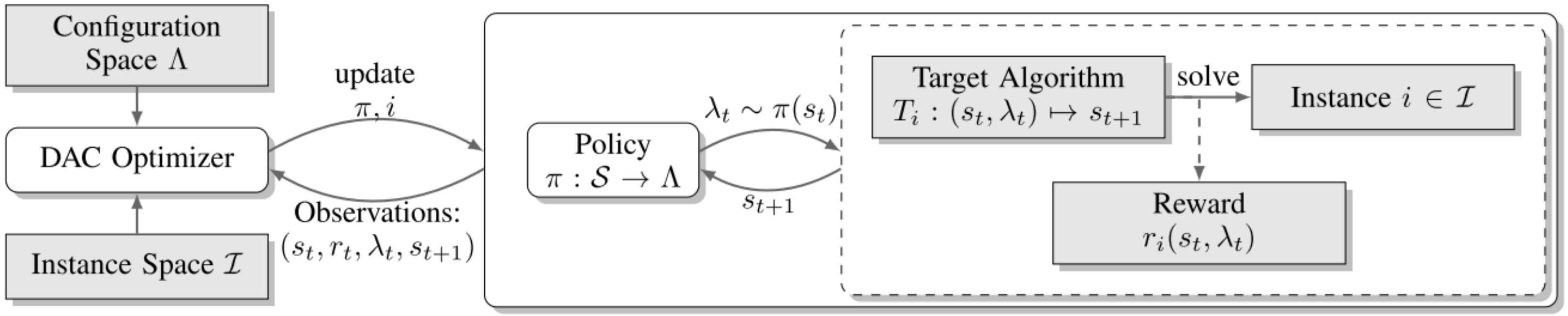
Dynamic algorithm configuration (DAC) is a new trend in Auto-ML.



Algorithm Configuration (AC)



Dynamic Algorithm Configuration (DAC)



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

[Eimer et al., IJCAI'21]

https://www.automl.org/automated-algorithm-design/dac/

Background

DAC has been found to outperform static methods on many tasks

- learning rate tuning of deep neural networks
- step-size adaptation of evolution strategies
- heuristic selection of AI planning

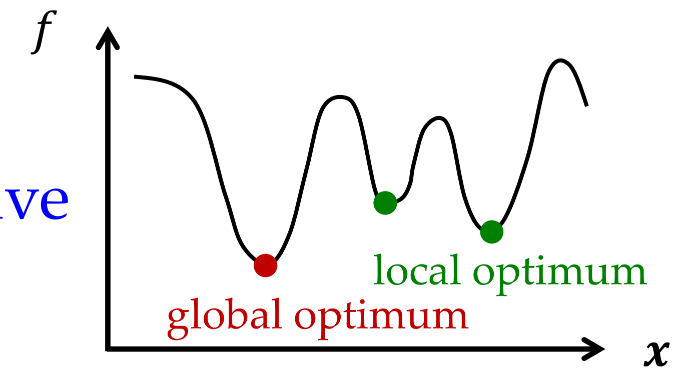
The task of DAC typically focuses on a single type of hyperparameter

However, due to the increasing complexity of real-world problem modeling, there are many algorithms whose performance rests on multiple types of hyperparameters.

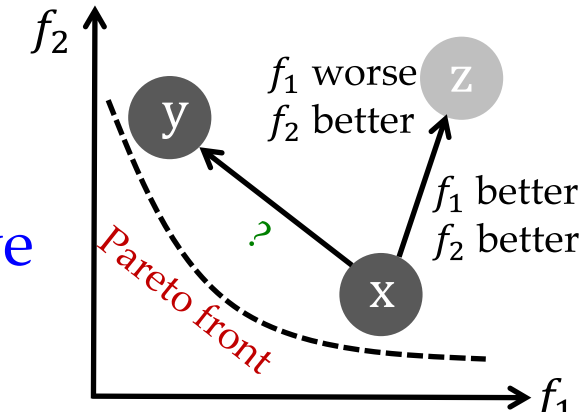
Background

Multi-objective optimization problems (MOPs)

Single-objective

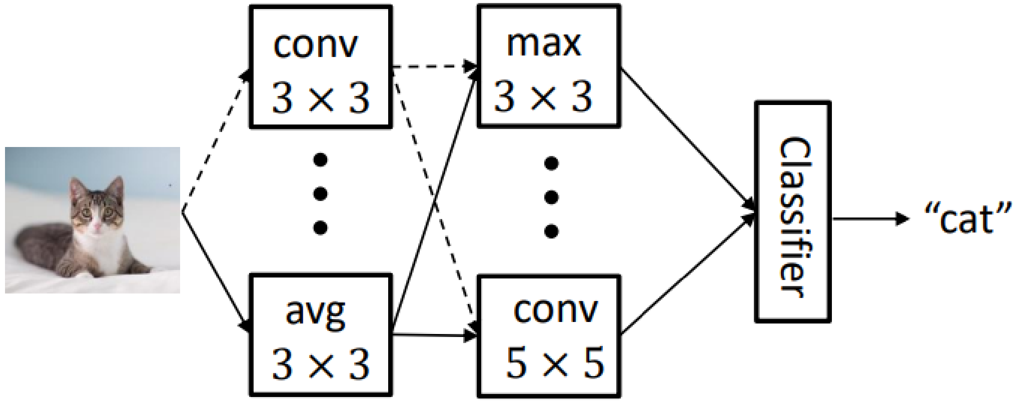


Multi-objective



Neural architecture search

- Max: accuracy
- Min: computation cost



Algorithm 1: MOEA/D

Parameters: Population size N, number T of iterations

Initialize a population $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ of solutions, and a corresponding set $W = \{\boldsymbol{w}^{(i)}\}_{i=1}^{N}$ of weight vectors ;
$t = 0$;
while $t < T$ **do**
  for $i = 1 : N$ **do**
    Randomly select parent solutions from the neighborhood of $\boldsymbol{w}^{(i)}$, denoted as $\Theta^{\boldsymbol{w}^{(i)}}$;
    Use crossover and mutation operators to generate an offspring solution $\boldsymbol{x}'^{(i)}$;
    Evaluate the offspring solution to obtain $\boldsymbol{F}(\boldsymbol{x}'^{(i)})$;
    Update the ideal point $\boldsymbol{z}^*$. That is, for any $j \in \{1, 2, \ldots, m\}$, if $f_j(\boldsymbol{x}'^{(i)}) < z_j^*$, then $z_j^* = f_j(\boldsymbol{x}'^{(i)})$;
    Update the corresponding solution of each sub-problem within $\Theta^{\boldsymbol{w}^{(i)}}$ by $\boldsymbol{x}'^{(i)}$. That is, for each $\boldsymbol{w}^{(j)} \in \Theta^{\boldsymbol{w}^{(i)}}$, if $g(\boldsymbol{x}'^{(i)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*) < g(\boldsymbol{x}^{(j)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*)$, then $\boldsymbol{x}^{(j)} = \boldsymbol{x}'^{(i)}$
  end
  $t = t + 1$
end

Complex and hard to tune

MA-DAC

How to dynamically adjust multiple types of configuration hyperparameters of complex algorithm such as MOEA/D?

We propose **MA-DAC**, modeling the configuration of a complex algorithm with multiple types of hyperparameters as a *cooperative multi-agent problem*, where **one agent works to handle one type of hyperparameter**.

MA-DAC

We consider a common-payoff, fully cooperative multi-agent setting

Different agents have different actions (**heterogeneous**)

Algorithm 1: MOEA/D

Parameters: Population size N, number T of iterations

Initialize a population $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ of solutions, and a corresponding set $W = \{\boldsymbol{w}^{(i)}\}_{i=1}^{N}$ of weight vectors ;

$t = 0$;

while $t < T$ **do**

    for $i = 1 : N$ **do**

        Randomly select parent solutions from the neighborhood of $\boldsymbol{w}^{(i)}$, denoted as $\Theta^{\boldsymbol{w}^{(i)}}$;

        Use crossover and mutation operators to generate an offspring solution $\boldsymbol{x}'^{(i)}$;

        Evaluate the offspring solution to obtain $\boldsymbol{F}(\boldsymbol{x}'^{(i)})$;

        Update the ideal point $\boldsymbol{z}^*$. That is, for any $j \in \{1, 2, \ldots, m\}$, if $f_j(\boldsymbol{x}'^{(i)}) < \boldsymbol{z}^*_j$, then $\boldsymbol{z}^*_j = f_j(\boldsymbol{x}'^{(i)})$;

        Update the corresponding solution of each sub-problem within $\Theta^{\boldsymbol{w}^{(i)}}$ by $\boldsymbol{x}'^{(i)}$. That is, for each $\boldsymbol{w}^{(j)} \in \Theta^{\boldsymbol{w}^{(i)}}$, if $g(\boldsymbol{x}'^{(i)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*) < g(\boldsymbol{x}^{(j)} \mid \boldsymbol{w}^{(j)}, \boldsymbol{z}^*)$, then $\boldsymbol{x}^{(j)} = \boldsymbol{x}'^{(i)}$

    end

    $t = t + 1$

end

MA-DAC

We analyze the formulation of contextual MMDP

- State
- Action
- Reward
- Transition

MA-DAC

We analyze the formulation of contextual MMDP

- State
- Action
- Reward
- Transition

Multi-agent RL for Multi-objective optimization (MaMo) benchmark

Benchmark	Heterogeneous	# of agents	Stochastic	Application scenarios
Matrix Games [5]	×	2	Low	Game
MPE [20]	×	2-3	Low	Game
MAgent [42]	×	2-1000	Low	Game
SMAC [28]	✓	2-30	Low	Game
Active Voltage Control [38]	×	3-38	Low	Control
MaMo (Ours)	✓	2-4	High	Optimization

Experiment

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?

Experimental results show the superior performance of MA-DAC

Contribution

1) To the best of our knowledge, MA-DAC is the **first one** to address dynamic configuration of algorithms with multiple types of hyperparameters.
2) The contextual MMDP formulation of MA-DAC is analyzed, and experimental results show that the presented formulation works well and has good **generalization** ability.
3) The instantiation of configuring MOEA/D in this work can be used as a benchmark problem for MARL.
    1) The **heterogeneity** of MOEA/D's hyperparameters and the **stochasticity** of its search can promote the research of the MARL algorithms.
    2) Besides, the learned policies are **useful** for multi-objective optimization, which will facilitate the application of MARL.

Our code is available at
https://github.com/lamda-bbo/madac

Thank you!

✉ **xuek@lamda.nju.edu.cn**