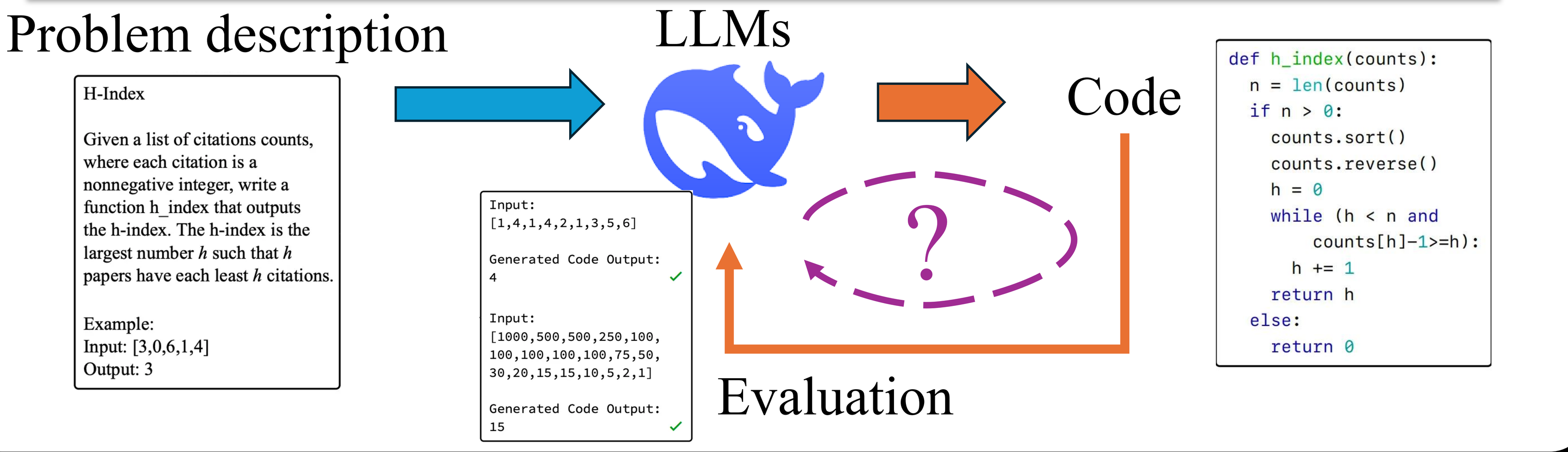
Nested-Refinement Metamorphosis: Reflective Evolution for Efficient Optimization of Networking Problems

Shuhan Guo, Nan Yin, James T. Kwok, Quanming Yao
THU & HKUST
Interested in future works?
Contact: Quanming Yao (qyaoaa@tsinghua.edu.cn) & Nan Yin (yinnan8911@gmail.com)

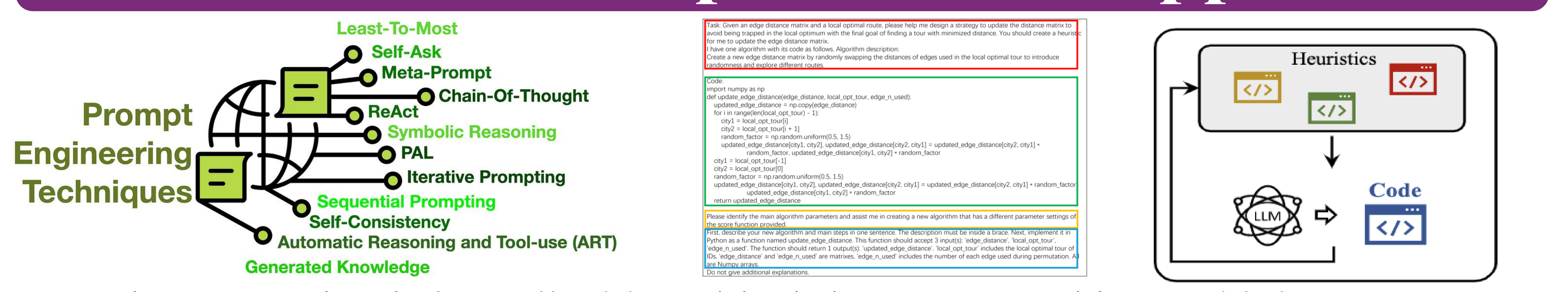
ACL 2025 VIENNA
JULY 27 - AUGUST 1

1. LLM for Algorithm Design

How to Obtain Higher-Quality Algorithms Faster?



2. Current Code Optimization Approaches



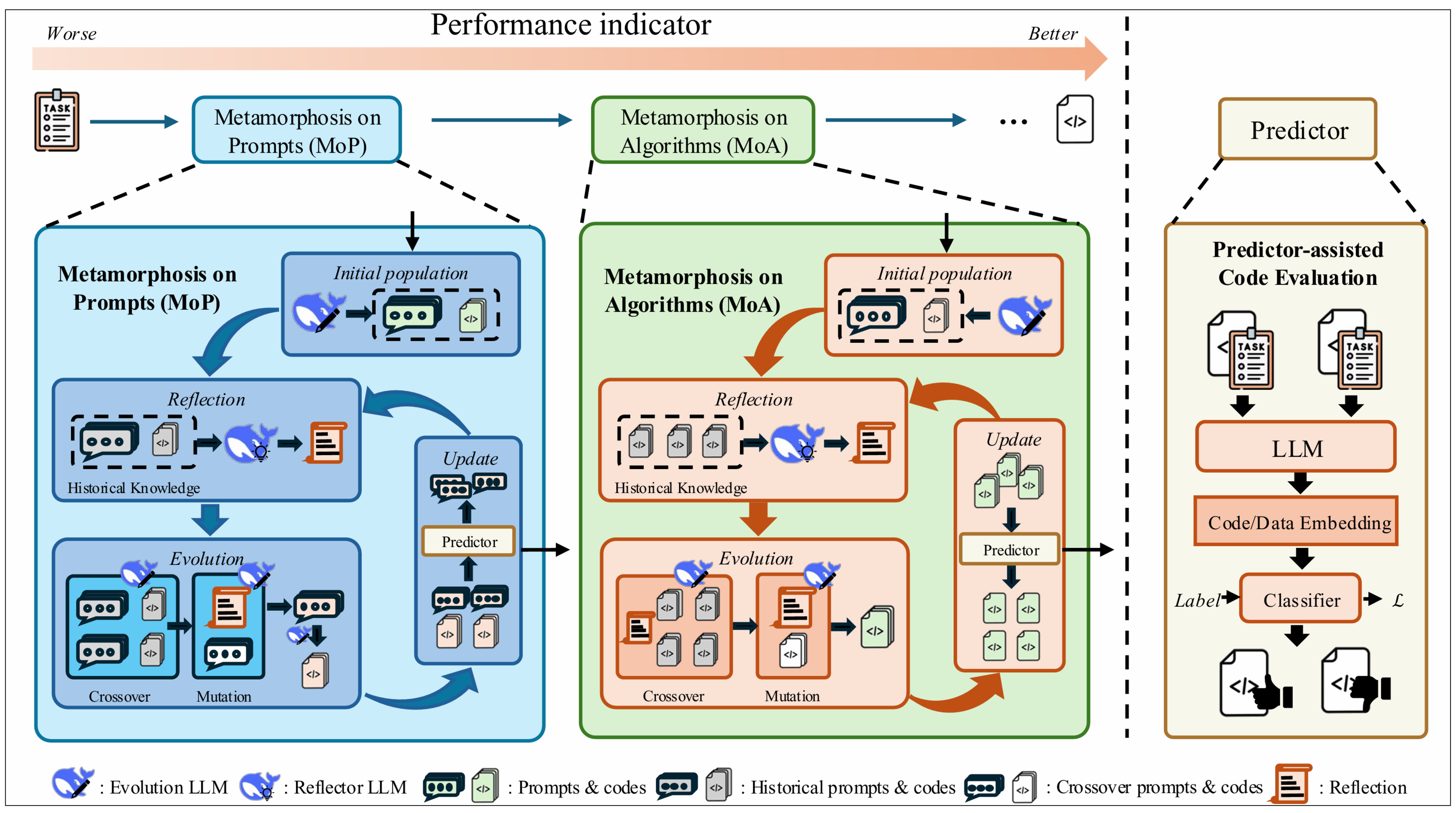
- x Continuous tuning is inapplicable to black-box LLM and incurs high costs.
- x Discrete prompt design with predefined templates fails to address algorithmic flaws.
- x Exclusive code evolutions relying on genetic-based refinement lack task prompt optimization and exhaustive evaluation, causing high computational costs and frequent requirement misalignment. ✓ Need better guidance & lower cost

3. Methodology

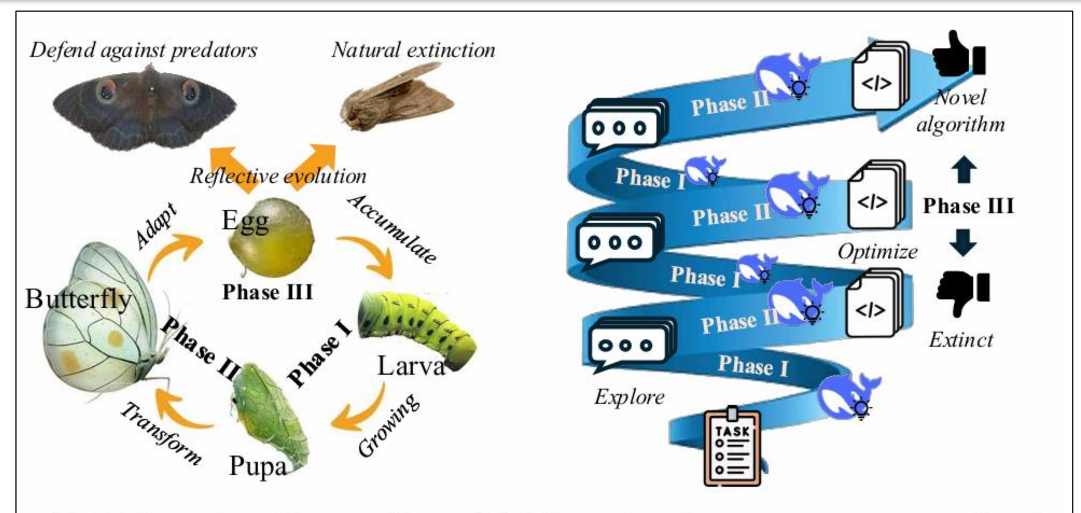
NeRM achieves efficient algorithm optimization through a three-phases bionic-inspired nested framework, integrating iterative **MoP**/**MoA** refinement with **predictor**-assisted candidate filtering.

- ☐ Phase I - **Metamorphosis on Prompts**. → **Refine the prompts**
- ☐ Phase II - **Metamorphosis on Algorithms**. → **Refine the codes guided by the evolved prompt**
- ☐ Phase III - **Nested evolution**. → **Predictor assisted code evaluation, filtering low-quality algorithms early**

feedback



3.1 Bionic-Inspired Three-Stage Strategy



(a) Lifecycle of butterfly. (b) Nested-refinement metamorphosis.

3.2 Nested-Refinement Metamorphosis

- ☐ MoP iteratively refines task descriptions to align with problem constraints.
- ☐ MoA evolves algorithms via reflection-driven adaptation.
- ☐ "Prompt->k*Code->Prompt->k*Code" manner, each provide feedback to others.

3.3 Predictor-Assisted Code Evaluation

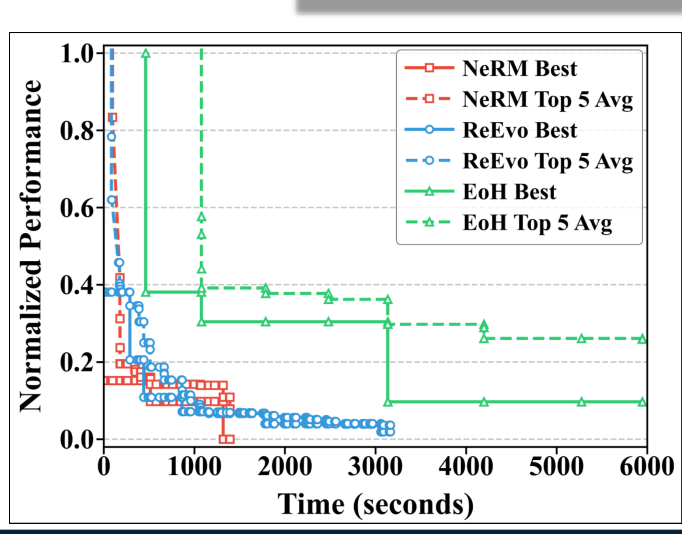
- ☐ Embeds code-data pairs via LLM into shared space
- ☐ MLP ranks candidates to prune low-quality trials
- ☐ Online fine-tuning aligns with target problem constraints
- ☐ Skips actual execution of poor codes

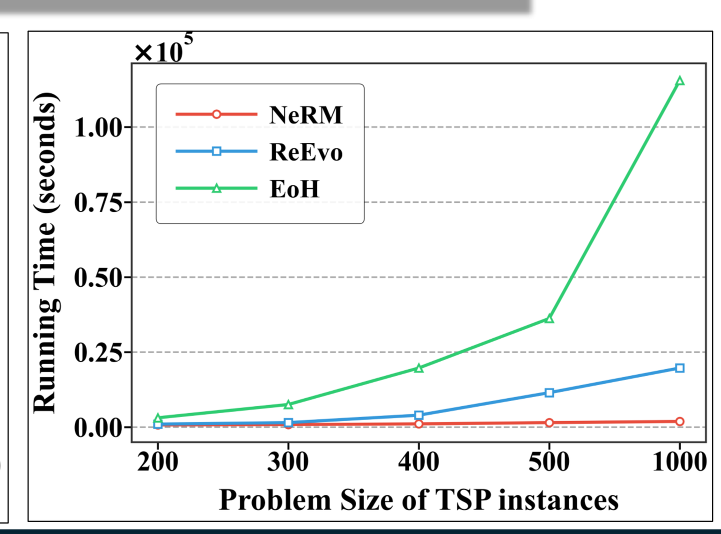
4. Experiments

4.1 High Performance

Method	TSP100	TSP200	TSP500	TSP1000
GLS	$0.66_{\pm 0.00}$	$2.70_{\pm 0.00}$	$3.86_{\pm 0.00}$	$4.22_{\pm 0.00}$
KGLS	$0.01_{\pm 0.00}$	$0.28_{\pm 0.00}$	$0.92_{\pm 0.00}$	$1.55_{\pm 0.00}$
EBGLS	$0.44_{\pm 0.00}$	$1.36_{\pm 0.00}$	$2.65_{\pm 0.00}$	$4.09_{\pm 0.00}$
EEPE	$0.00_{\pm 0.00}$	$0.24_{\pm 0.04}$	$0.81_{\pm 0.07}$	$1.40_{\pm 0.17}$
CoAPT	$0.01_{\pm 0.00}$	$0.24_{\pm 0.23}$	$0.93_{\pm 0.99}$	$1.58_{\pm 1.64}$
APR	$0.01_{\pm 0.00}$	$0.24_{\pm 0.24}$	$0.80_{\pm 0.78}$	$1.46_{\pm 1.39}$
EoH	$0.04_{\pm 0.01}$	$0.36_{\pm 0.05}$	$1.48_{\pm 0.13}$	$2.26_{\pm 0.09}$
ReEvo	$0.00_{\pm 0.00}$	$0.23_{\pm 0.24}$	$0.85_{\pm 0.87}$	$1.42_{\pm 1.44}$
NeRM	$\mathbf{0.00}_{\pm 0.00}$	$\mathbf{0.19}_{\pm 0.04}$	$\mathbf{0.77}_{\pm 0.04}$	$\mathbf{1.31}_{\pm 0.08}$

4.2 Efficient Evolution





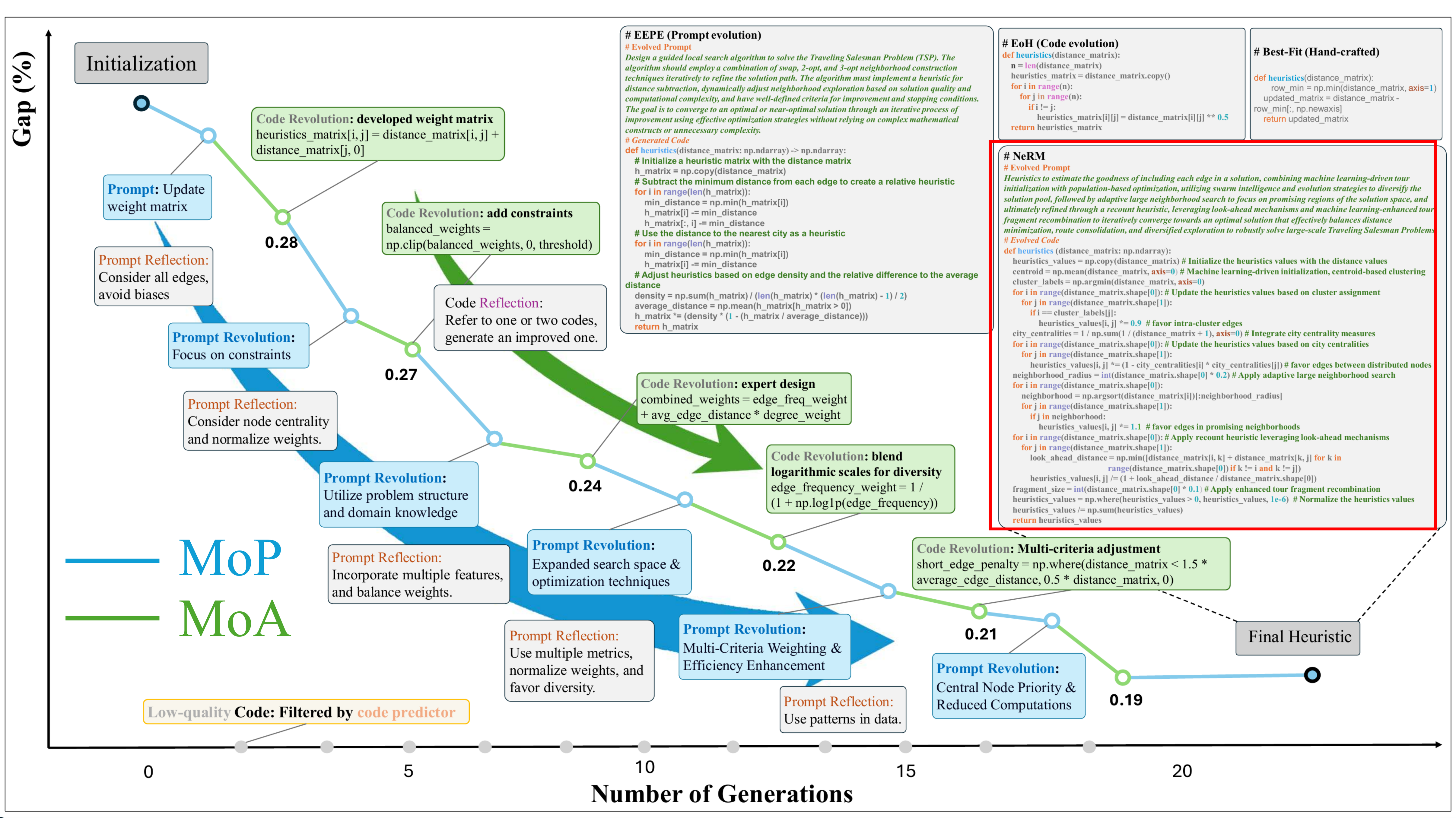
4.3 Effectiveness of Submodules

Method	Gap (%)	Time (s)
w/o predictor	$\mathbf{0.18}_{\pm 0.02}$	104
w/o reflection	$0.22_{\pm 0.01}$	63
w/o revolution	$0.24_{\pm 0.04}$	62
w/o MoP	$0.34_{\pm 0.16}$	80
w/o MoA	$0.23_{\pm 0.04}$	72
NeRM	$0.19_{\pm 0.04}$	83

Stronger Performance & Higher Efficiency

Find out more in our paper…

5. Examples



6. Want more?

NeRM achieves:

Fast and continuous optimization

☐More Accurate Task Description

☐More Complex Algorithm Design

Paper

Code