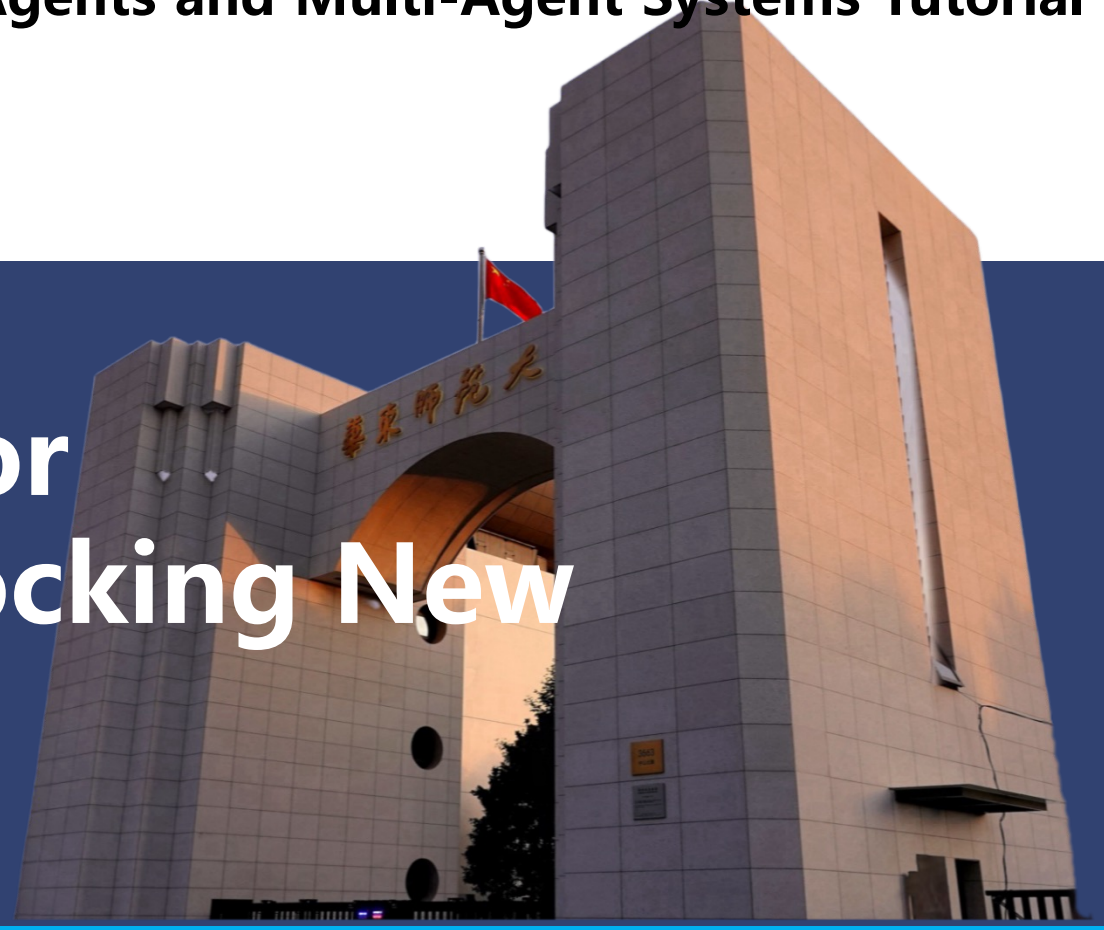
The 23rd International Conference on Autonomous Agents and Multi-Agent Systems Tutorial



Reinforcement Learning for Operations Research: Unlocking New Possibilities (Part I)

Xiangfeng Wang, Junjie Sheng (East China Normal University)
Wenhao Li (The Chinese University of Hong Kong, Shenzhen)
Contact email: xfwang@cs.ecnu.edu.cn

Operations Research Problems

Mathematical Programming | Combination Optimization | Graph Theory | Game

Artificial Intelligence

Adam、RMSprop algorithms; Large Foundation Model training

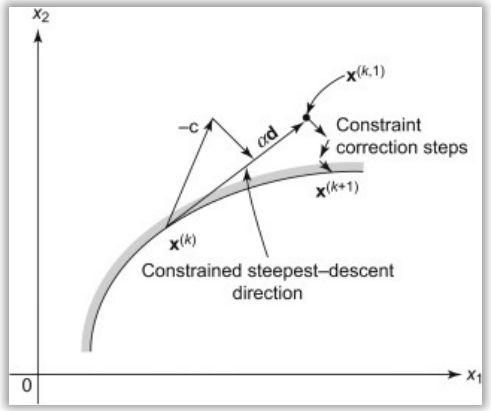
Advanced computing

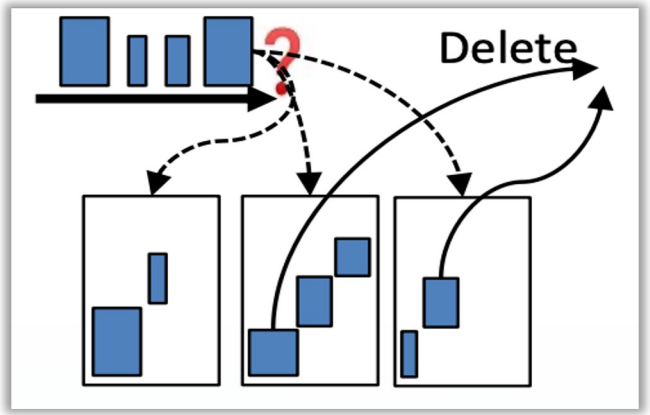
EDA layout and wiring; cloud computing planning and scheduling; SAT problems

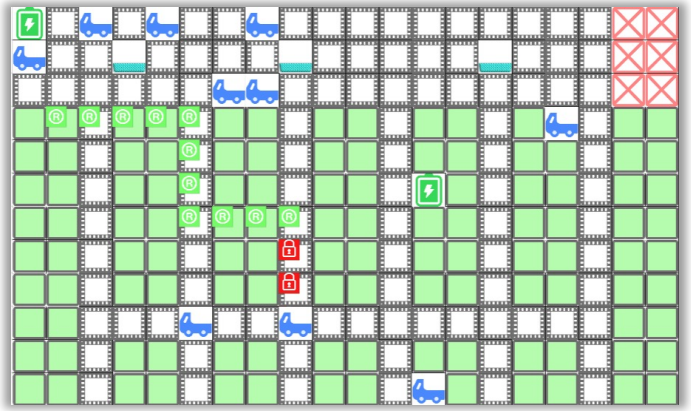
Path planning

Logistics/AGV/electromagnetic navigation path planning

Core: Searching





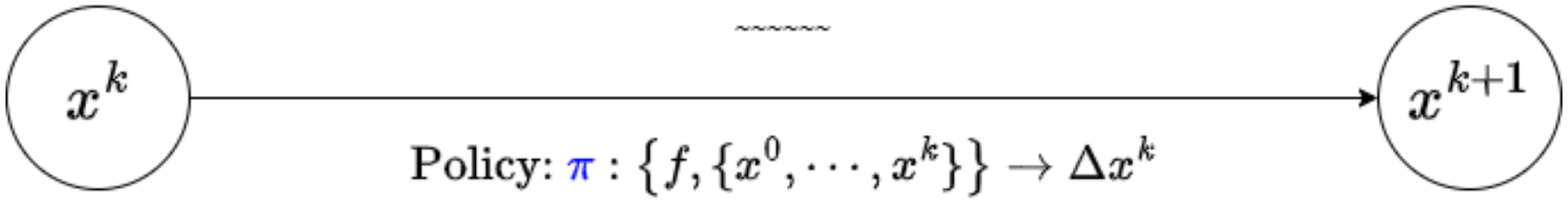


Mathematical Programming

First-order Methods	Second-order Methods	Primal-dual Methods	SGD-type Methods
Gradient descent Proximal gradient Nesterov acceleration Block coordinate-type	Newton method Quasi-Newton Semi-smooth Newton ……	ALM、ADMM Primal-dual hybrid gradient ……	SGD Mini-batch SGD SVRG、Adam、RMSprop ……

$$\text{Gradient Descent: } x^{k+1} = x^k - \gamma \nabla f(x^k)$$

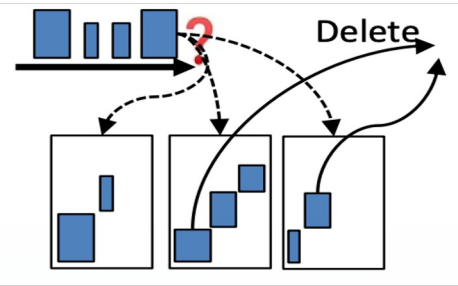
$$\text{Momentum: } x^{k+1} = x^k - \gamma \left[\sum_{\ell=0}^{k} \alpha_\ell \nabla f(x^\ell)\right]$$



K. Li, J. Malik, Learning to Optimize, ***ICLR***, 2017.

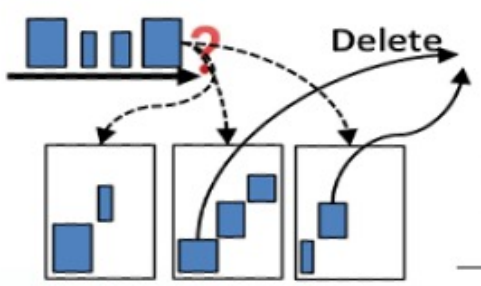
Scheduling Problem

System Scheduler Request



Vector Bin-packing Problem

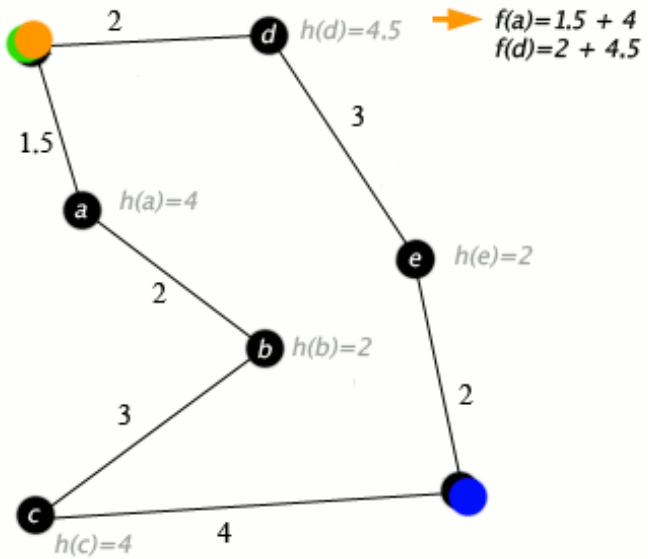
Combination Optimization	Heuristic methods
➔ Batch scheduling / Optimal solution ➔ Solver (Gurobi, COPT, etc.)	➔ Online scheduling ➔ Low complexity

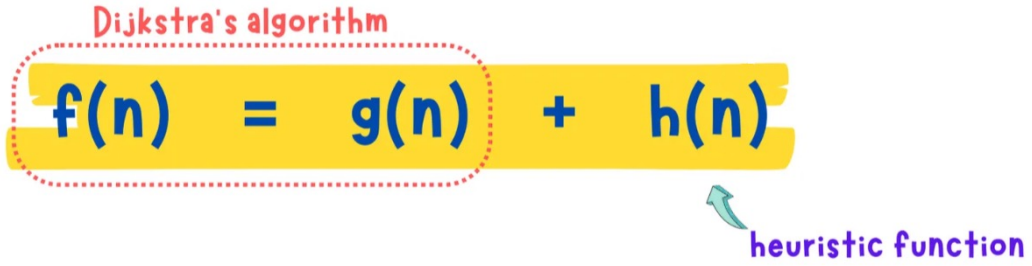


J Sheng, and etc., Learning to Schedule Multi-NUMA Virtual Machines via Reinforcement Learning, ***Pattern Recognition*** 121, 108254, 2022.

Graph Theory Problem

Graph Structure	Vertex Problems	Edge Problems	Traversing/Sorting
Maximum flow	Maximum independent set	Shortest path	DFS problem
The isomorphism	Minimum control set	Minimum cut	BFS problem
The connectivity	Hamiltonian circle	Minimum spanning tree	Topological sorting
.....	Minimum point coverage		.....





A-star for shortest path

Operations Research Methods

There is no universal method that can be considered as the "best" solution to all problems.

Solvers	
Commercial	Gurobi, COPT, CPLEX, MOSEK, Lingo
Open source	SCIP, Google OR-Tools, SDPT3
Specialized	Metis、 LeafVein

Data-driven

Strcuture-driven

Problem-specific

……

Mathematical Programming

Constant step-size

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma \nabla f(\mathbf{x}^k)$$

Barzilai-Borwein step-size

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{(\mathbf{x}^k - \mathbf{x}^{k-1})^{\mathrm{T}}(\mathbf{x}^k - \mathbf{x}^{k-1})}{(\mathbf{x}^k - \mathbf{x}^{k-1})^{\mathrm{T}}(\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1})} \nabla f(\mathbf{x}^k)$$

Adaptive step-size（Adam）

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{\eta}{\sqrt{v_k} + \varepsilon} m_k,\ m_k = \beta_1 m_{k-1} + (1-\beta_1)\nabla f_{i(k)}(\mathbf{x}^k),\ v_k = \beta_2 v_{k-1} + (1-\beta_2)\|\nabla f_{i(k)}(\mathbf{x}^k)\|^2$$

Based on professional experience, enable optimization methods to quickly "iterate" and evolve

Mathematical Programming

Benchmarking Suite

- Test Problems
  - 2D Test Problems
  - Quadratic Test Problems
  - MNIST Test Problems
  - Fashion-MNIST Test Problems
  - CIFAR-10 Test Problems
  - CIFAR-100 Test Problems
  - SVHN Test Problems
  - ImageNet Test Problems
    - ImageNet VGG16
    - ImageNet VGG19
    - ImageNet Inception v3
  - Tolstoi Test Problems

State

Baseline Algorithms

SGD Momentum Adam

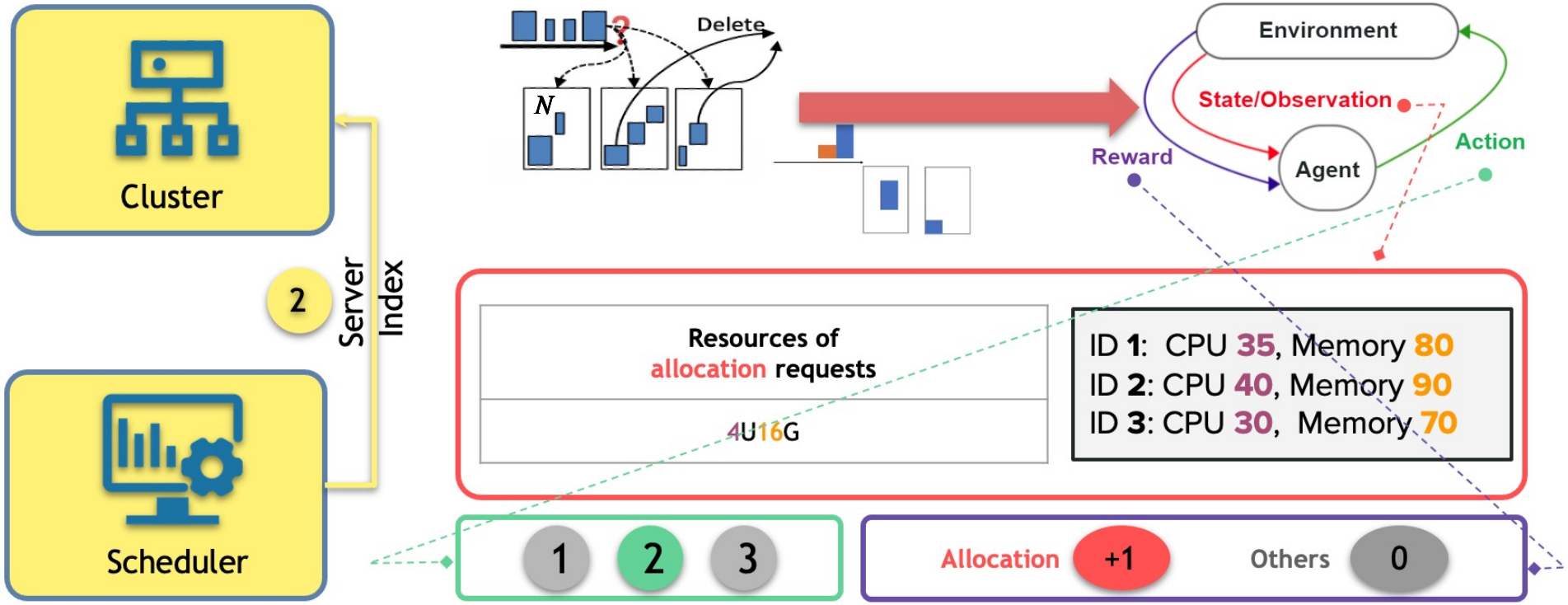
Design “new” method

Action

Environment

Scheduling

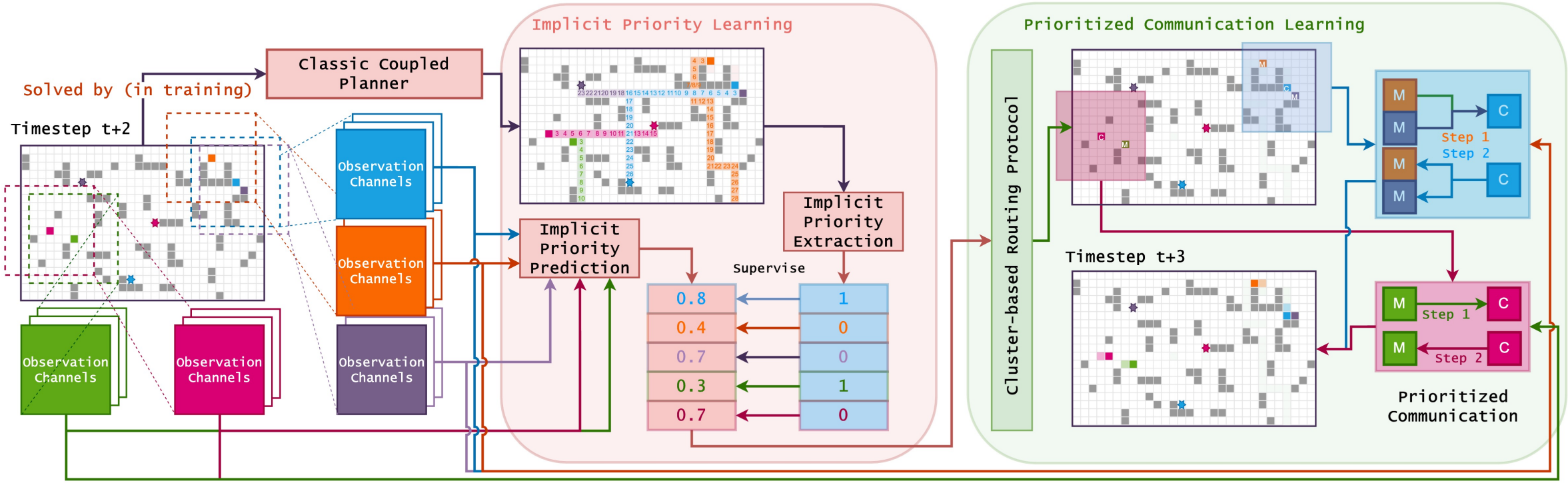
- ✓ State (Observation) Space: N servers with resources(CPU/Memory) and creation requests
- ✓ Action Space: server numbers {1, 2, …, N}
- ✓ State Transition: Transitioning from the current state s to the next state s` after placing a request
- ✓ Reward Setting: Successful placement +1, unsuccessful placement 0 (Is this simple reward setup appropriate?)



J Sheng, and etc., Learning to Schedule Multi-NUMA Virtual Machines via Reinforcement Learning, ***Pattern Recognition*** 121, 108254, 2022.

Multi-Agent Pathfinding

- ✓ State (Observation) Space: Obstacle and AGV positions, goal positions, or directions
- ✓ Action Space: Up, down, left, right, and stay in place.
- ✓ State Transition: Transitioning from the current state *s* to the next state *s`* after all AGVs move
- ✓ Reward Setting: Move -0.3, collision -2.0, no movement (on/off goal) 0.0/-0.5, finish episode +20.0



W. Li, H. Chen, B. Jin, W. Tan, H. Zha, and X. Wang. Multi-Agent Path Finding with Prioritized Communication Learning. ***ICRA*** 2022.

Large Foundation Model

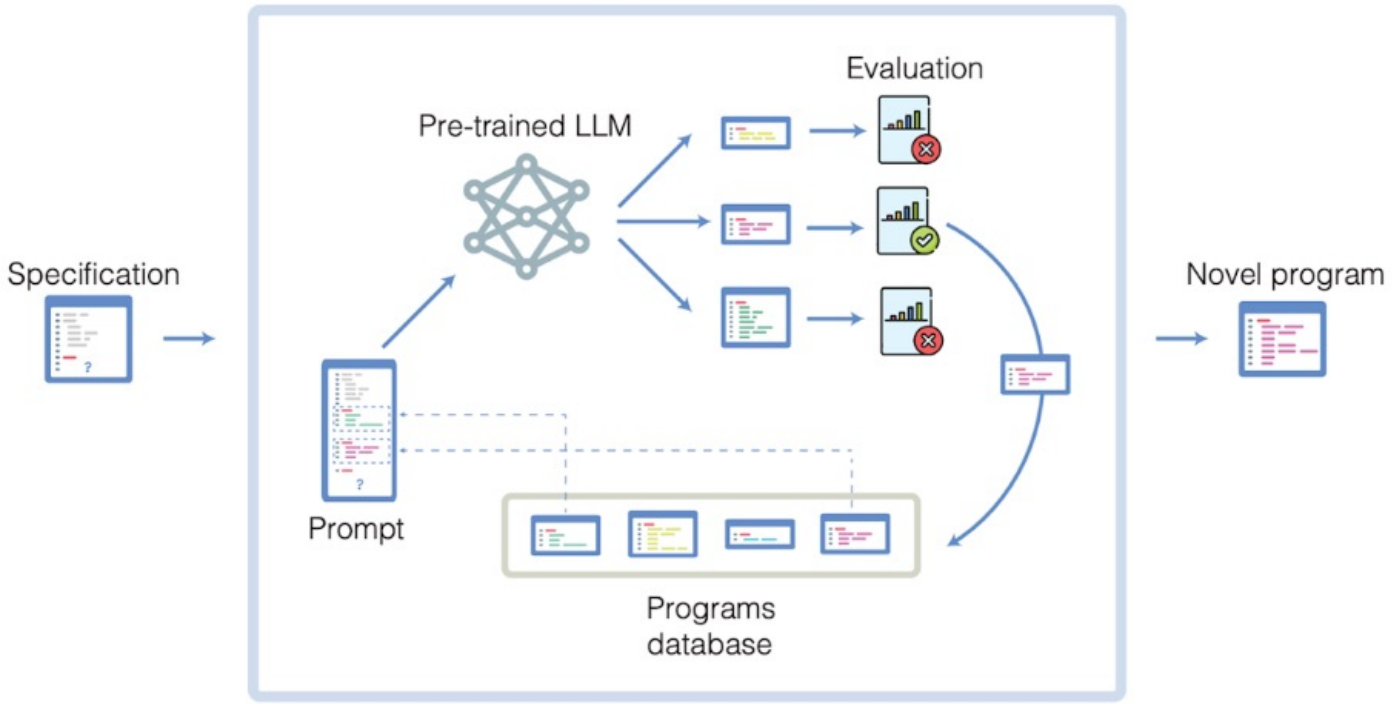
Large Language Models (LLMs)	
LLMs	ChatGPT(OpenAI) / LLaMA(Meta) / Gemma(Google)
	Kimi(Moonshot) / Qwen(Alibaba)
Multimodal	Sora(OpenAI) / Gemini(Google) / GPT-5?

Do you need skills such as dialogue or text generation?

What abilities are needed for a large model?
(Representation ability? Reasoning ability?)

LLM-based RL for OR

Reinforcement Learning



Funsearch【*Nature* 2024】【Deepmind】

Romera-Paredes, B., Barekatain, M., Novikov, A. *et al.* Mathematical discoveries from program search with large language models. *Nature* **625**, 468–475, 2024.

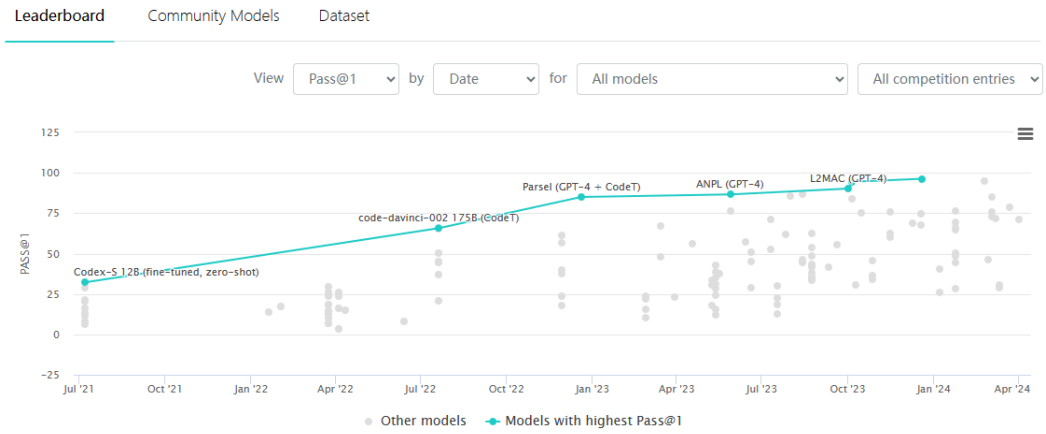
LLM-based RL for OR

Code! A special language with logic!

```
def priority(bin, item):
    """
    Returns priority with which we want to add item to each bin.
    Args:
        bin (int): Total available CPU resources.
        item (int): Item need to be placed.

    Returns:
        int: The total score for current bin.

    """
    score = -(bin[0] - item[0])
    return score
```

Code LLMs	
OpenAI	OpenAI Codex / GPT4
Google	CodeGemma 7B/2B
ZhipuAI	CodeGeeX (GLM) 13B
High-flyer	DeepSeek Coder 33B

Code Generation on HumanEval

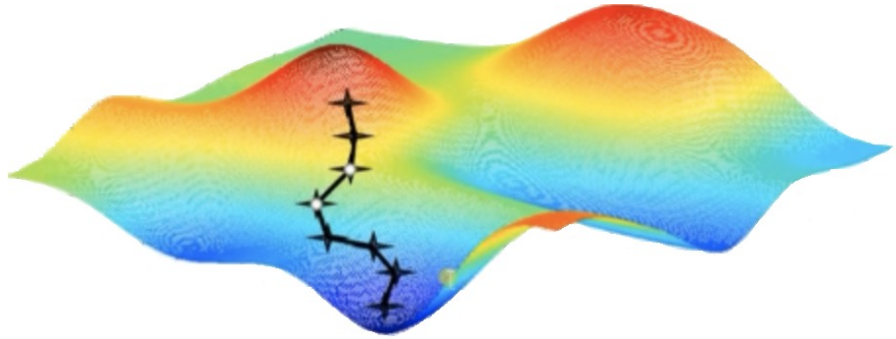


https://paperswithcode.com/sota/code-generation-on-humaneval

LLM-based RL for OR – VM Scheduling

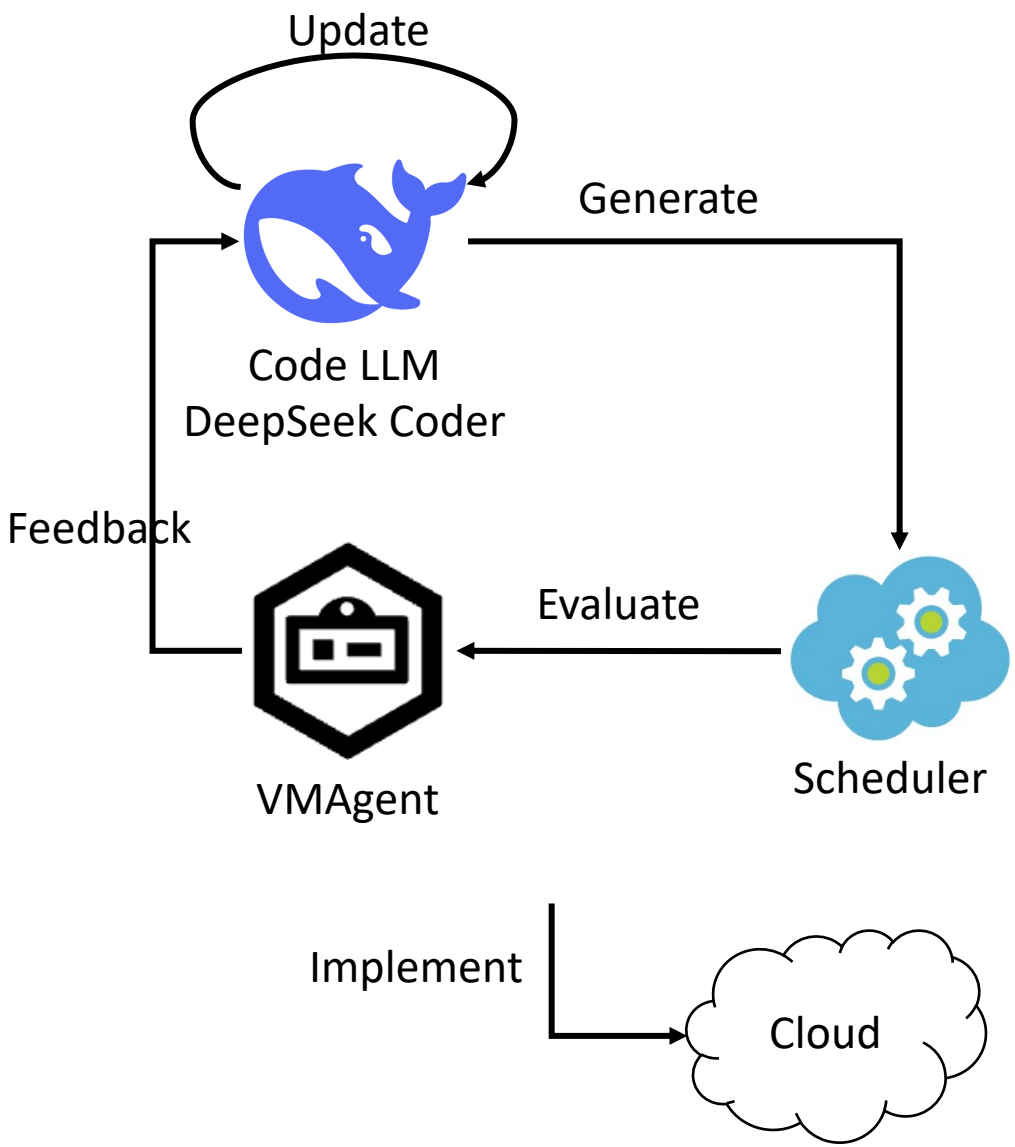
```
def priority(bin, item):
    """
    Returns priority with which we want to add item
to each bin.
    Args:
        bin (int): Total available CPU resources.
        item (int): Item need to be placed.

    Returns:
        int: The total score for current bin.

    """
    score = -(bin[0] - item[0])
    return score
```

Mapping



Given a method → Code LLMs → Evolution to optimal method

LLM-based RL for OR – VM Scheduling



Prompt

Given the existing priority_v0 function, please generate an optimized version named priority_v*. This new version should be more complex and efficient, incorporating multiple conditional logic and loops as necessary. The function should calculate priorities for items to be added to bins, considering the item size and bin capacities.

```
def priority(bin, item):
    score = - (bin[0] - item[0])
    return score
```

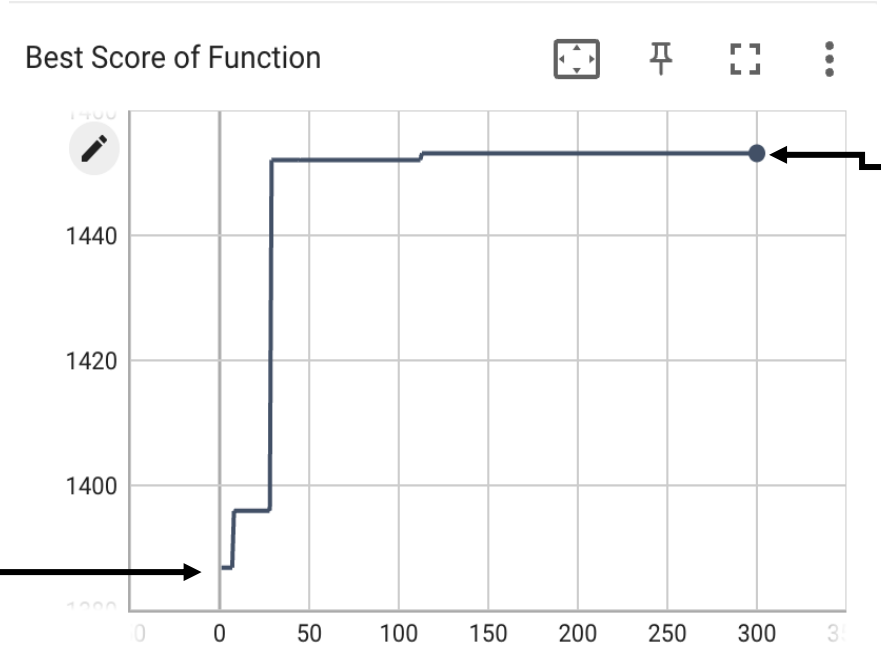
BestFit

```
def priority(bin, item):
    score = 0
    for i in range(len(bin)):
        diff = bin[i] - item[i]
        score += diff if diff > 0  else bin[i] + item[i]
        score *= 2 if diff < -5 else 1
        score += np.sin(score) if score > 100 else 0
    …
    return score
```

BestFit*

LLM-based RL for OR – VM Scheduling

```
def priority(bin, item):
    """
    Returns priority with which we want to add item to each bin.
    Args:
        bin (int): Total available CPU resources.
        item (int): Item need to be placed.

    Returns:
        int: The total score for current bin.

    """
    score = - (bin[0] - item[0])
    return score
```



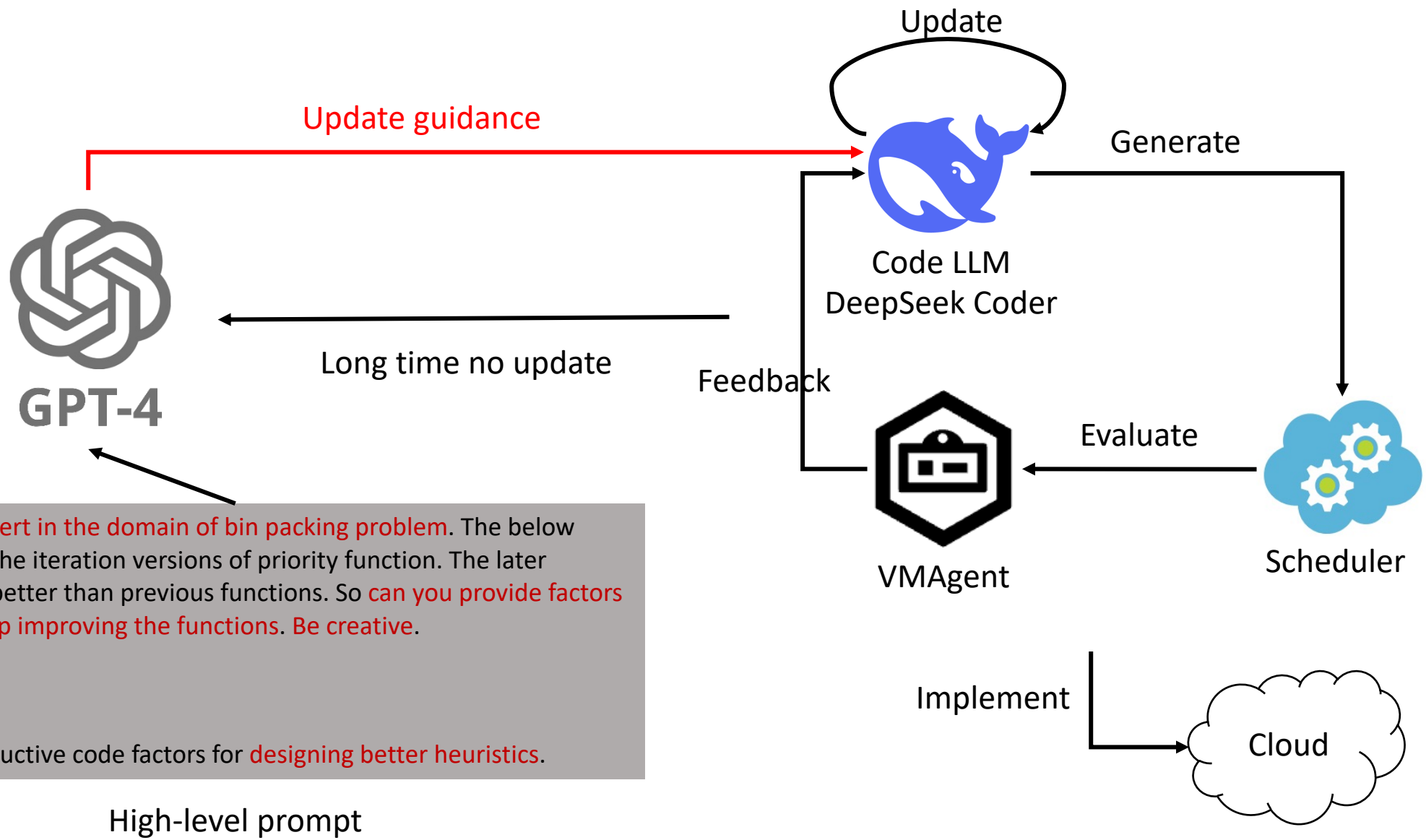

```
def priority(bin, item):
    score = 0
    for i in range(len(bin)):
        diff = bin[i] - item[i]
        score += diff if diff > 0 else bin[i] + item[i]
        score *= 2 if diff < -5 else 1
        score += np.sin(score) if score > 100 else 0
    score = -abs(score + 10)
    score += 5 if score % 2 == 0 else -5
    score += sum(range(100))
    score += item[-1]
    return score
```

Advantage
- Efficiency
- Emergence
- Generalization
- Universality

Scenarios	Size	BestFit	BestFit*	Training
Testing	30	566.9	**572.1333**	
Training	50	1386.8	**1453.133**	**A100 80G**
Testing	100	3811.5	**3823.633**	**300 epochs**
Testing	150	6256.266	**6306.133**	**7~8 hours**
Testing	200	8366.466	**8447.9**	

Shortcoming
- Interpretability?
- Prompt design?

LLM-based RL for OR – VM Scheduling

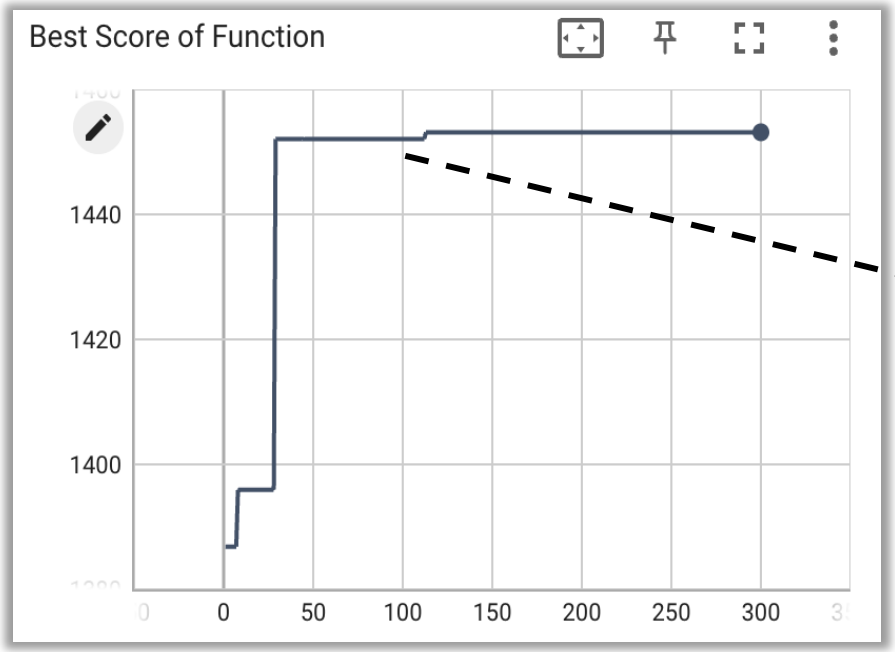


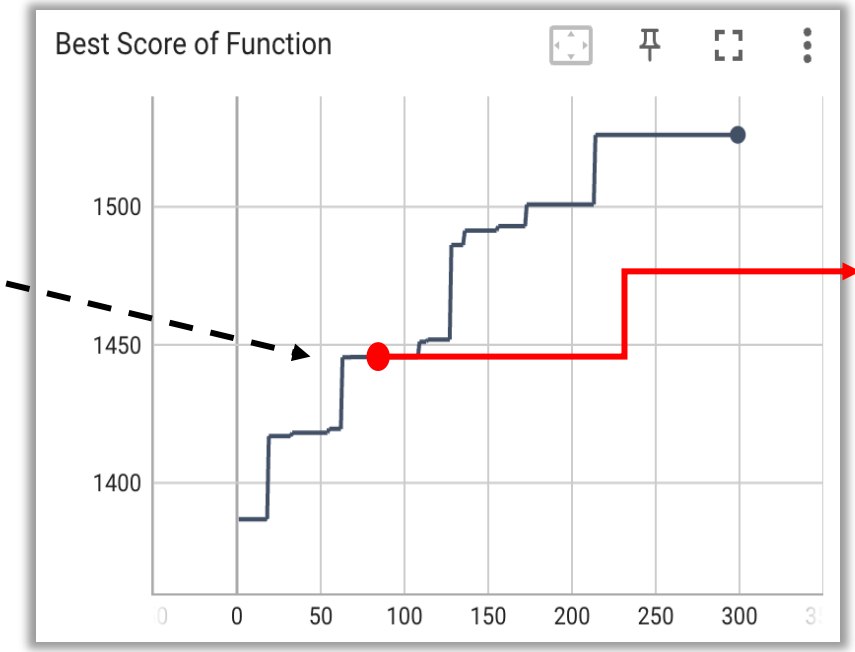
You are an expert in the domain of bin packing problem. The below functions are the iteration versions of priority function. The later functions are better than previous functions. So can you provide factors which may help improving the functions. Be creative.
priority_1()
priority_2()

… Write constructive code factors for designing better heuristics.

High-level prompt

LLM-based RL for OR – VM Scheduling





1. Normalize score

```
for score in scores:
    score /= max_possible_score
```

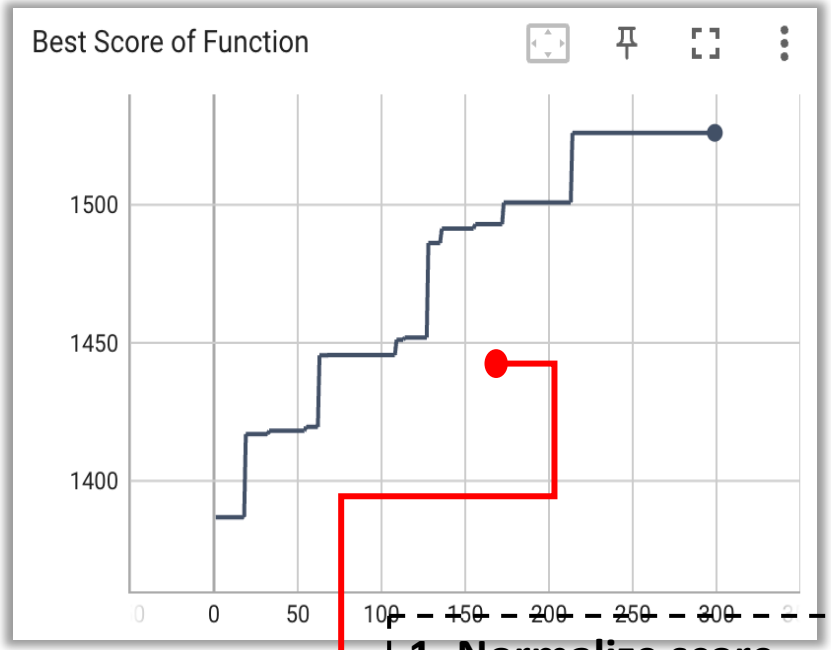
2. Exponential Bonus:

```
for score in scores:
    scores.append(np.exp(score))
total_score = sum(scores)
```

Update guidance

Size	BestFit	BestFit*	BestFit**	Traing
30	566.9	572.1333	**572.6333**	
50	1386.8	1453.133	**1526.066**	**A100 80G**
100	3811.5	3823.633	**3874.966**	**300 epochs**
150	6256.266	6306.133	**6318.166**	**7~8 hours**
200	8366.466	**8447.9**	8403.366	

LLM-based RL for OR – VM Scheduling



1. Normalize score

```
for score in scores:
    score /= max_possible_score
```

2. Exponential Bonus:

```
for score in scores:
    scores.append(np.exp(score))
total_score = sum(scores)
```

Update guidance

Empty Bin Bonus: If the bin is empty, add a predefined bonus:

$$\text{score}_{\text{empty}} = \text{EMPTY\_BIN\_BONUS} \times (\text{bin} == 0)$$

Proportion Bonus: Add a bonus if the proportion of item to bin sum is below a threshold:

$$\text{score}_{\text{proportion}} = \text{PROPORTION\_BONUS} \times (\frac{\sum \text{item}}{\sum \text{bin}} < \text{THRESHOLD\_1})$$

Proximity Bonus: Add a bonus if the bin is close to reaching maximum capacity:

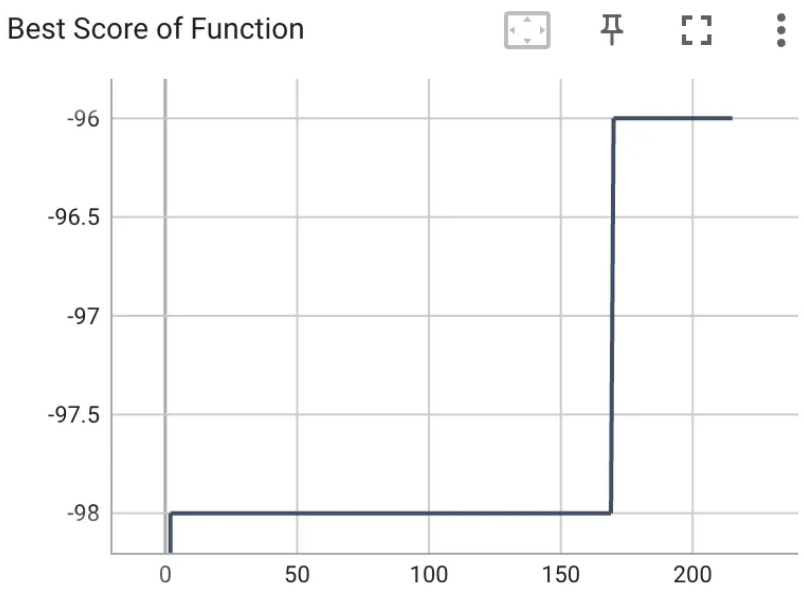
$$\text{score}_{\text{proximity}} = \text{PROXIMITY\_BONUS} \times (|\sum \text{bin} - \text{MAX\_BIN\_CAPAC}$$

Half of Bonus: Add a bonus if any bin is less than half full:

$$\text{score}_{\text{half}} = \text{HALF\_OF\_BONUS} \times (\text{any}(\text{bin} < \frac{\text{MAX\_BIN\_CAPACITY}}{2}))$$

The final priority function that considers multiple factors

LLM-based RL for OR – Adam method optimization



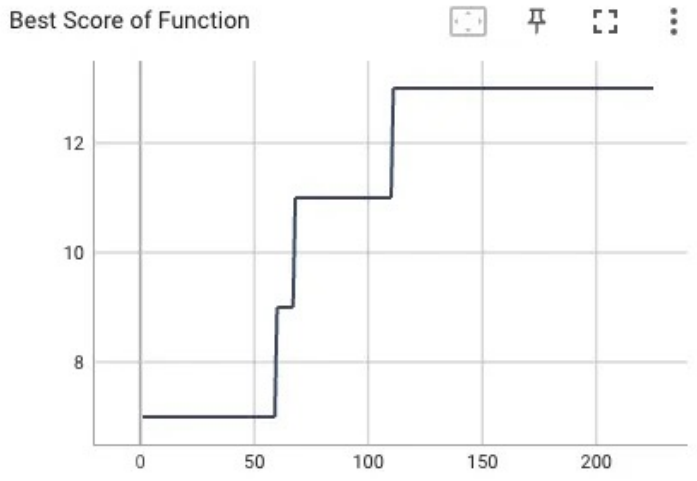
Adam	Adam*
132	96

Epoch numbers to achieve 80% accuracy
MLP model / MNIST dataset

```
def adan(...):
    bias_correction1 = 1 / (1 - beta1 ** step)
    bias_correction2 = 1 / (1 - beta2 ** step)

    # Update the exponential moving averages of the gradients and squared gradients
    for param, grad, exp_avg, exp_avg_sq in zip(parameters, grads, exp_avgs,
exp_avg_sqs):
        if grad is None:
            continue
        exp_avg.mul_(beta1).add_(1 - beta1, grad)
        exp_avg_sq.mul_(beta2).addcmul_(1 - beta2, grad, grad)

    # Adjust learning rate
    lr_adjusted = lr * (bias_correction2 ** 0.5) / bias_correction1

    # Update the parameters
    for param, exp_avg, exp_avg_sq in zip(parameters, exp_avgs, exp_avg_sqs):
        param.addcdiv_(exp_avg, exp_avg_sq.sqrt().add_(eps), value=-lr_adjusted)

    return parameter
```

Adam* (Python)

LLM-based RL for OR – SAT optimization



100s Timeout

Timeout	EasySAT	SAT*
100s	45/400	53/400
200s	53/400	61/400
300s	56/400	**76**/400

```
void priority(xxxx) {
   const double max_activity = 1e100, activity_factor = 1e-100,
decay_factor = 0.9, decay_threshold = 1e-7;
   double& var_activity = activity[var];

var_activity += coeff;
   if (var_activity >= max_activity)
   {
      var_activity = activity_factor * var_activity;
      for (int i = 0; i < vars; ++i)
      {   activity[i] = activity_factor * activity[i];    }
      var_inc = activity_factor * var_inc;
   }
   if (activity[var] < decay_threshold)
   {
      activity[var] = 0.0;
      var_inc = decay_factor * var_inc;
      for(int i = 0; i < vars; ++i)
      {   activity[i] = decay_factor * activity[i];   }
   }
   // Update the variable in the heap
   vsids.update(var);
```

SAT* (C++)

RL for OR

Paradigm

Expert-RL

LLM-based RL

GPT-4

LLM for LLM-based RL

Universality

MP algorithms/Scheduling/SAT/etc.

Theory

Provable ? Evaluation and Analysis of Algorithm Evolution Capability (Tool?)

References

- ✓ Romera-Paredes, B., Barekatain, M., Novikov, A. et al. Mathematical discoveries from program search with large language models. ***Nature*** 625, 468–475, 2024. **Deepmind**
- ✓ L. Lehnert, S. Sukhbaatar, P. Mcvay, M. Rabbat, Y. Tian. Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping, ***arXiv:2402.14083***, 2024. **Meta**
- ✓ S. Liu, C. Chen, X. Qu, K. Tang, Y.-S. Ong. Large Language Models as Evolutionary Optimizers, ***arXiv:2310.19046***, 2023.
- ✓ Z. Fan, B. Ghaddar, X. Wang, L. Xing, Y. Zhang, Z. Zhou. Artificial Intelligence for Operations Research: Revolutionizing the Operations Research Process, ***arXiv:2401.03244***, 2024. **Huawei**
- ✓ J. Sheng, Z. Huang, C. Shen, W. Li, Y. Hua, B. Jin, H. Zha, X. Wang. Can Language Agents Approach the Performance of RL? An Empirical Study On OpenAI Gym, openreview.net/pdf?id=F0q880yOgY, 2023.
- ✓ S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, D. Schuurmans. Foundation models for decision making: Problems, methods, and opportunities, ***arXiv:2303.04129***, 2023. **Deepmind/BAIR**
- ✓ Sherry Yang, Jacob Walker, Jack Parker-Holder, Yilun Du, Jake Bruce, Andre Barreto, Pieter Abbeel, Dale Schuurmans. Video as the New Language for Real-World Decision Making, ***arXiv:2402.17139***, 2024. **Deepmind/BAIR**
- ✓ ……