

When Optimization Meets Large Language Models

Xianchao Xiu

Department of Automation



Peking University, October 30, 2026

Joint work with [Jianhao Li](#) (SHU), [Xinjie Wang](#) (SUEP), [Pengfei Zhang](#) (SHU) and others

Outline

Introduction

Modeling Optimization

Low-Rank Optimization

Policy Optimization

Conclusion

Introduction

► Large language models (in Chinese)

对于提出的联邦 PCA 模型，请设计 ADMM 算法，提供代码，并分析收敛性和复杂度

```
\begin{equation}\label{eq:proposed_method}
\begin{array}{c}
\displaystyle \min_{\{W_i\}, \{S_i\}, V} \quad \sum_{i=1}^d \left\| (W_i - S_i) \right\|_F^2 \\
\text{s.t.} \quad W_i \in \mathbb{R}^{n \times p}, S_i \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{n \times p} \\
\text{and } W_i^T W_i = I_k, V^T V = I_p
\end{array}
\end{equation}
```

联邦鲁棒 PCA：ADMM 完整设计 + Python 可运行代码 + 收敛性分析 + 计算复杂度

整体框架概述

符号约定

- d : 客户端数量; 第 i 方数据 $X_i \in \mathbb{R}^{n_i \times p}$, n_i : 每方样本数, p : 特征维度
 - $W_i \in \mathbb{R}^{p \times k}$: 客户端局部投影矩阵, 满足 Stiefel 正交约束 $W_i^T W_i = I_k$
 - $V \in \mathbb{R}^{p \times k}$: 全局共享公共投影矩阵 (联邦耦合变量)
 - $S_i \in \mathbb{R}^{n_i \times p}$: 稀疏噪声 / 异常矩阵, l_1 正则去除偏差
 - $\|W_i\|_{2,1}$: 行稀疏正则, 让投影基具备行稀疏性
 - $\alpha > 0$: 稀疏权重; $\beta > 0$: 行稀疏权重; $\rho > 0$: ADMM 增广拉格朗日惩罚系数
- 约束: $\forall i, W_i = V, W_i^T W_i = I$

原目标:

$$\min_{\{W_i, S_i\}, V} \sum_{i=1}^d \left\{ \left\| (I - W_i W_i^T)(X_i - S_i) \right\|_F^2 + \alpha \|S_i\|_1 + \beta \|W_i\|_{2,1} \right\}, \quad \text{s.t. } W_i = V, W_i^T W_i = I$$

一、ADMM 算法推导

1. 增广拉格朗日函数

引入对偶乘子 $\Lambda_i \in \mathbb{R}^{p \times k}$, 增广拉格朗日:

$$\mathcal{L}_\rho(\{W_i, S_i\}, V, \{\Lambda_i\}) = \sum_{i=1}^d \left[\left\| (I - W_i W_i^T)(X_i - S_i) \right\|_F^2 + \langle \Lambda_i, W_i - V \rangle + \frac{\rho}{2} \|W_i - V\|^2 \right]$$

ADMM 四块交替迭代:

- 客户端并行更新 S_i
 - 客户端并行求解带 Stiefel 约束的 W_i
 - 服务器聚合更新全局 V
 - 更新对偶变量 Λ_i
- #### 2. 子问题闭式 / 数值解

(1) S_i 更新 (逐元素软阈值)

固定其余变量, 令 $Y_i = X_i - S_i$, $P_i = I - W_i W_i^T$ 幂等投影阵:

$$\min_{S_i} \|P_i(X_i - S_i)\|_F^2 + \alpha \|S_i\|_1$$

闭式解:

$$S_i = S_o(X_i - W_i W_i^T X_i)$$

软阈值算子: $S_\tau(z) = \text{sign}(z) \max(|z| - \tau, 0)$

```
python ^
import numpy as np

# ===== 辅助工具函数 =====
def soft_threshold(x, tau):
    """L1软阈值"""
    return np.sign(x) * np.maximum(np.abs(x) - tau, 0.0)

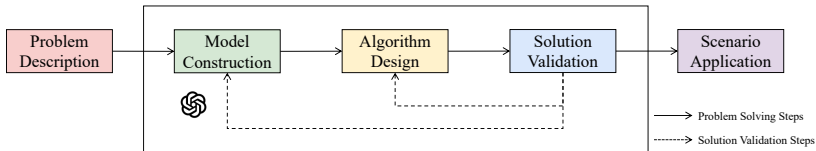
def row_2l_prox(W, beta):
    """L2,1行近端算子: 每行L2范数收缩"""
    p, k = W.shape
    for r in range(p):
        norm_r = np.linalg.norm(W[r, :], ord=2)
        if norm_r > 1e-8:
            scale = max(1 - beta / norm_r, 0.0)
            W[r, :] *= scale
    return W

def stiefel_retraction(W):
    """SVD回缩至Stiefel流形 W^T W = I"""
    U, S, Vt = np.linalg.svd(W, full_matrices=False)
    return U @ Vt

def stiefel_pgq(C, V_ref, Lambda, rho, beta, lr=1e-2, inner_iter=10):
    """
    Stiefel流形近端梯度下降求解W
    min -tr(W^T C) + 0.5*rho||W - (V_ref - Lambda/rho)||_F^2
    """
    p, k = C.shape[0], V_ref.shape[1]
    W = np.random.randn(p, k)
    W = stiefel_retraction(W)
    target = V_ref - Lambda / rho

    for _ in range(inner_iter):
        # 欧氏梯度
        grad = -C + rho * (W - target)
        W = W - lr * grad
        # L2,1近端
        W = row_2l_prox(W, beta * lr)
        # 流形回缩正交化
        W = stiefel_retraction(W)
    return W
```

► LLM for Optimization & Optimization for LLM



Llm4fs: Leveraging large language models for feature selection

J Li, X Xiu - 2025 China Automation Congress (CAC), 2025 - ieeexplore.ieee.org

Recent advances in large language models (LLMs) have provided new opportunities for decision-making, particularly in the task of automated feature selection. In this paper, we comprehensively evaluate LLM-based feature selection methods, covering the state-of-the-art [DeepSeek-R1](#), GPT-o3-mini, and GPT-4.5. Then, we propose a novel hybrid strategy called LLM4FS that integrates LLMs with traditional data-driven methods. Specifically, we feed data samples into LLMs, and directly call traditional data-driven techniques such as

☆ 保存 引用 被引用次数: 8 相关文章

LLM-Driven Reasoning for Constraint-Aware Feature Selection in Industrial Systems

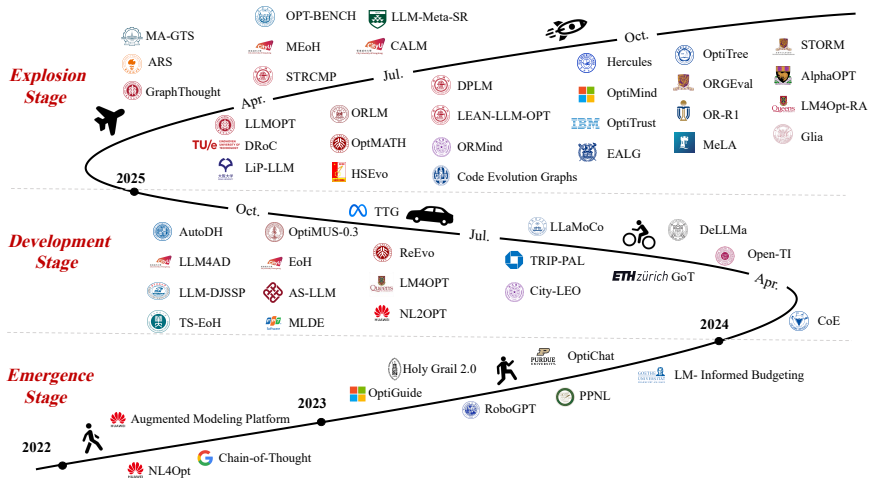
Yuhang Zhou, Zhuokai Zhao, Ke Li, Spilios Evmorfos, Gökalp Demirci, Mingyi Wang, Qiao Liu Wang, Serena Li, Weiwei Li, Tingting Wang, Mingze Gao, Gedi Zhou, Abhishek Kumar, Xiangj Lizhu Zhang, Jiayi Liu

Meta AI

Feature selection is a crucial step in large-scale industrial machine learning systems, directly affecting model accuracy, efficiency, and maintainability. Traditional feature selection methods rely on domain knowledge and statistical heuristics, making them difficult to apply in production environments where data and statistical heuristics, making them difficult to apply in production environments where data are limited and multiple operational constraints must be satisfied. To address this, we propose **Model Feature Agent (MOFA)**, a large language model-driven framework that performs e

Optimization

- Xiu-Li-Fan et al, Large Language Models for Operations Research: A Comprehensive Survey, [submitted to JORSC](#)



Advances

- ▶ Ramamonjison-Yu-Li et al, NL4Opt Competition: Formulating Optimization Problems Based on Their Natural Language Descriptions, [NeurIPS](#), 2022
- ▶ Yang-Wang-Lu et al, Large Language Models as Optimizers, [ICLR](#), 2024
- ▶ AhmadiTeshnizi-Gao-Udell, OptiMUS: Scalable Optimization Modeling with (MI)LP Solvers and Large Language Models, [ICML](#), 2024
- ▶ Romera-Paredes-Barekatain et al, Mathematical Discoveries from Program Search with Large Language Models, [Nature](#), 2024
- ▶ Huang-Tang-Hu et al, ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling, [OR](#), 2025
- ▶ Liu-Wang-Cai et al, OptiTree: Hierarchical Thoughts Generation with Tree Search for LLM Optimization Modeling, [NeurIPS](#), 2025
- ▶ Zeng-Yang-Lei et al, A Universal Framework for Vehicle Routing Problems with Large Language Models, [JORSC](#), 2026
- ▶ He-Li-Li et al, ReasFlow: Assisting Reasoning-Centric Scientific Discovery in Applied Mathematics via a Knowledge-Based Multi-Agent System, [arXiv:2607.14178](#)

Outline

Introduction

Modeling Optimization

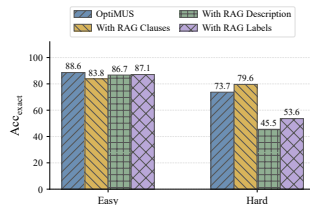
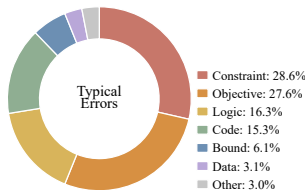
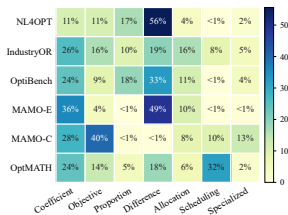
Low-Rank Optimization

Policy Optimization

Conclusion

Motivation

- ▶ Edge-Trinh-Cheng et al, From Local to Global: A GraphRAG Approach to Query-Focused Summarization [arXiv:2404.16130](https://arxiv.org/abs/2404.16130)
- ▶ Gutiérrez-Shu-Gu et al, HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models, [NeurIPS](#), 2024
- ▶ Guo-Xia-Yu et al, LightRAG: Simple and Fast Retrieval-Augmented Generation, [EMNLP](#), 2025
- ▶ Zhu-Huang-Wang et al, Graph-Based Approaches and Functionalities in Retrieval-Augmented Generation: A Comprehensive Survey, [ACM Computing Surveys](#), 2026



Example

- **Problem Type:** Vehicle routing problem (VRP)
- **Variant:** Capacitated vehicle routing problem (CVRP)
- **Description:** Design vehicle delivery routes to serve all customers with minimal travel cost under vehicle capacity constraints
- **Math Model:**

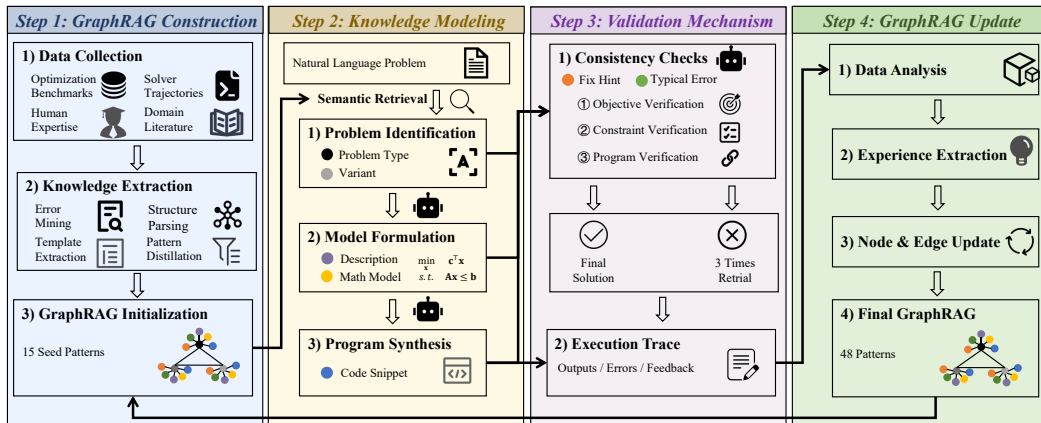
$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijk} \\ \text{s.t.} \quad & \sum_{k \in K} \sum_{j \in V} x_{ijk} = 1, \quad \forall i \in C \\ & \sum_{i \in C} d_i \sum_{j \in V} x_{ijk} \leq Q, \quad \forall k \in K \\ & x_{ijk} \in \{0, 1\}, \quad \forall i, j \in V, k \in K \end{aligned}$$

- **Code Snippet:**

```
x = addVars(V, V, K, vtype=GRB.BINARY)
min sum(c[i,j] * x[i,j,k] for i in V for j in V for k in K)
addConstrs(each customer visited once), addConstrs(vehicle capacity limits)
```
- **Typical Error:** Missing customer coverage, ignoring vehicle capacity, disconnected subtours
- **Fix Hint:** Visit each customer once, add capacity constraints, eliminate subtours

Workflow

► GraphRAG-enhanced agentic workflow



Experiments

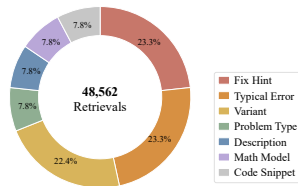
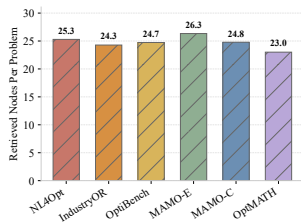
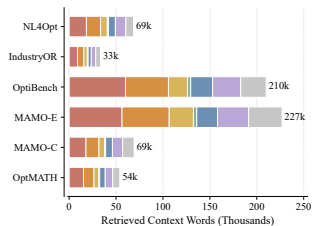
- ▶ Compared methods
 - ▶ GPT-5.1: 2025-11-13
 - ▶ Gemini-3.1-Pro: 2026-2-19
 - ▶ DeepSeek-V3.2: 2025-12-1
 - ▶ Qwen3.6-Plus: 2026-4-2
 - ▶ CoE: Xiao-Zhang-Wu et al, ICLR, 2024
 - ▶ OptiMUS: AhmadiTeshnizi-Gao-Brunborg et al, ICML, 2024
 - ▶ OptiTree: Liu-Wang-Cai, NeurIPS, 2025
 - ▶ Lean-LLM-OPT: Liang-Lu-Mao et al, arXiv:2601.09635
- ▶ Implementation details
 - ▶ GPT-5.1 for modeling, Gemini-3.1-Pro for validation
 - ▶ Evaluation metrics

$$\text{Acc}_{\text{exact}} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i), \quad \text{Acc}_{5\%} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}\left(\frac{|\hat{y}_i - y_i|}{|y_i| + \eta} \leq 0.05\right)$$

Results

► Accuracy comparisons

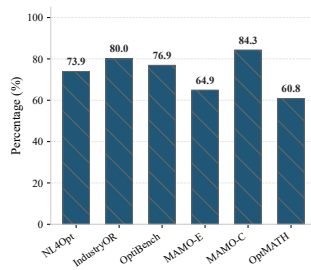
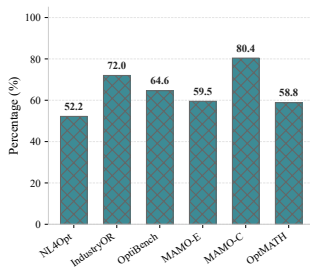
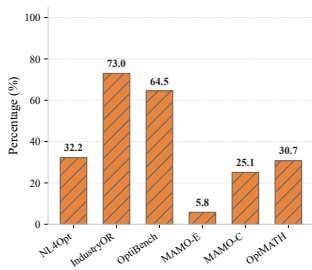
Methods	NL4OPT	IndustryOR	OptiBench	MAMO-E	MAMO-C	OptMATH
GPT-5.1	72.9%/78.0%	43.0%/51.0%	58.0%/65.8%	81.6%/85.8%	47.8%/58.1%	27.7%/30.7%
Gemini-3.1-Pro	73.4%/78.5%	42.0%/50.0%	58.2%/66.0%	82.1%/86.1%	46.8%/57.1%	27.1%/30.1%
DeepSeek-V3.2	72.4%/77.6%	44.0%/50.0%	57.4%/65.1%	81.0%/85.0%	48.3%/57.6%	26.5%/30.1%
Qwen3.6-Plus	69.6%/74.8%	42.0%/49.0%	55.4%/62.6%	78.0%/81.8%	46.3%/55.7%	27.1%/28.9%
CoE	77.1%/83.6%	51.0%/61.0%	65.5%/73.2%	93.5%/93.8%	67.0%/70.3%	40.4%/43.4%
OptiMUS	86.9%/93.9%	54.0%/60.0%	63.0%/72.1%	87.8%/89.0%	52.2%/70.4%	38.5%/41.3%
OptiTree	84.1%/92.5%	64.0%/71.0%	62.5%/71.9%	90.7%/91.2%	65.4%/72.8%	31.3%/33.7%
Lean-LLM-OPT	86.5%/94.9%	55.0%/63.0%	61.8%/70.3%	93.8%/94.2%	57.6%/70.0%	44.5%/45.2%
OptGraph (Ours)	86.9%/95.3%	71.0%/81.0%	66.0%/78.2%	93.9%/94.4%	76.4%/89.2%	58.4%/60.8%



Ablations

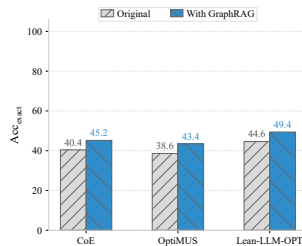
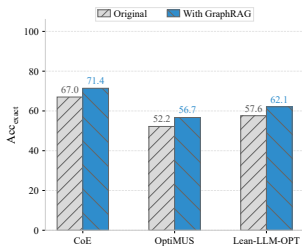
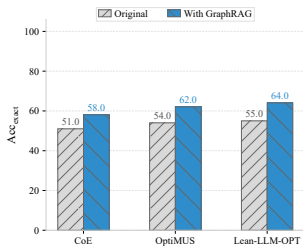
► Are GraphRAG & Validation important?

GraphRAG	Validation	NL4OPT	IndustryOR	OptiBench	MAMO-E	MAMO-C	OptMATH
×	×	67.8%/92.5%	47.0%/52.0%	60.8%/67.6%	85.8%/89.1%	52.2%/63.5%	36.1%/41.0%
×	✓	83.6%/93.9%	52.0%/57.0%	63.0%/69.3%	91.7%/92.8%	54.2%/67.9%	42.2%/44.6%
✓	×	84.6%/94.4%	66.0%/74.0%	64.5%/74.5%	91.9%/93.0%	68.0%/79.8%	51.8%/54.8%
✓	✓	86.9%/95.3%	71.0%/81.0%	66.0%/78.2%	93.9%/94.4%	76.4%/89.2%	58.4%/60.8%



Discussions

► Plug-and-play effects on IndustryOR, MAMO-C, and OptMATH

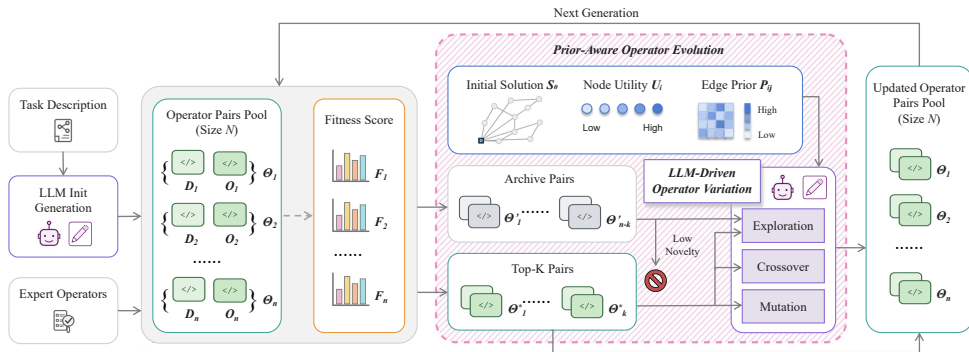


► Computational costs in average

Methods	Tokens (k)	Time (s)	Acc_{exact}	$Acc_{5\%}$
CoE	1.3	79.1	65.8%	70.9%
OptiMUS	15.9	111.1	63.7%	71.1%
OptiTree	4.6	14.3	66.3%	72.2%
Lean-LLM-OPT	1.5	17.1	66.5%	72.9%
OptGraph (Ours)	9.7	93.9	75.4%	83.2%

Remarks

- ▶ How to reduce the computational cost via lightweight RAG?
- ▶ How to develop local optimization agents with small parameter scale?
- ▶ Shen-Sun-Xiu, UNICO: Unified LLM-Driven Neural Combinatorial Optimization for Vehicle Routing Problems, [CAC](#), 2026



Outline

Introduction

Modeling Optimization

Low-Rank Optimization

Policy Optimization

Conclusion

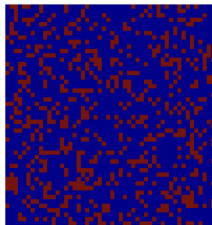
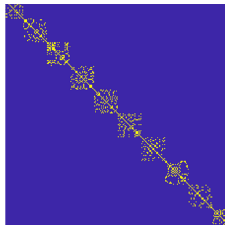
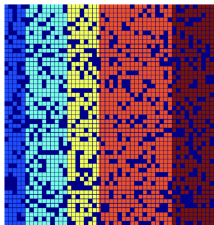
LRR

- Liu-Lin-Yan et al, Robust Recovery of Subspace Structures by Low-Rank Representation, *IEEE TPAMI*, 2013

$$\begin{aligned} \min_{Z, E} \quad & \text{rank}(Z) + \lambda \|E\|_0 \\ \text{s.t.} \quad & X = XZ + E \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} \min_{Z, E} \quad & \|Z\|_* + \lambda \|E\|_1 \\ \text{s.t.} \quad & X = XZ + E \end{aligned}$$



- Broad applications: clustering, classification, denoising, detection, fusion
- Related to low-rank regularization, low-rank decomposition, low-rank adaptation (LoRA)

Motivation

- Chen-Wang-Xiu et al, Rethinking Low-Rank Representation with Deep Semi-Smooth Newton, [PRCV](#), 2026

$$\min_{Z, E} \text{rank}(Z) + \lambda \|E\|_0$$

$$\text{s.t. } X = XZ + E$$

$\Rightarrow$

$$\min_{Z, E} \mathcal{R}(Z) + \lambda \mathcal{S}(E)$$

$$\text{s.t. } X = XZ + E$$

Methods	Datasets								Average	
	AR	FRDUE	MNIST	USPS	GLOMA	ISOLET	MSTAR	TDT2	ACC	Time
IDR (TKDE'23)	72.64%	95.31%	58.61%	79.80%	66.00%	67.65%	84.50%	84.10%	76.08%	225.80
TS1-LLRR (KBS'24)	76.46%	88.64%	49.65%	79.48%	61.60%	65.64%	87.61%	84.51%	74.20%	265.10
JSSC (ESWA'25)	78.09%	91.31%	46.78%	69.22%	50.00%	66.61%	65.99%	59.37%	65.92%	358.60
ISC (TPAMI'26)	54.42%	88.19%	44.02%	65.30%	60.00%	81.02%	43.13%	51.75%	60.98%	118.50
DSC-Net (NeurIPS'17)	45.19%	82.53%	66.07%	90.00%	70.00%	41.41%	82.53%	12.45%	61.27%	23.23
PRO-DSC (ICLR'25)	63.75%	96.09%	45.39%	67.60%	59.40%	52.00%	66.41%	88.08%	67.34%	0.03
DCSSC (Neuro'25)	83.68%	80.46%	87.25%	90.30%	84.00%	86.00%	91.85%	78.35%	85.24%	0.85
LRRSSN-Net	89.95%	96.02%	93.43%	99.40%	98.00%	97.72%	99.72%	96.40%	96.33%	0.21

- How to avoid k -means in clustering, i.e., end-to-end clustering?
- What are the advantages of large language models for clustering?

Model

- Incomplete multi-view clustering (IMVC)

$$\begin{aligned} \min_{Z, E} \quad & \mathcal{R}(Z) + \lambda \mathcal{S}(E) \\ \text{s.t.} \quad & X = XZ + E \end{aligned} \quad \Rightarrow \quad \begin{aligned} \min_{Z_v, \tilde{E}_v} \quad & \sum_{v=1}^V \mathcal{R}(Z_v) + \lambda \mathcal{S}(\tilde{E}_v) \\ \text{s.t.} \quad & \mathcal{P}_{\Omega}(\tilde{X}_v) = \mathcal{P}_{\Omega}(\tilde{X}_v Z_v + \tilde{E}_v) \end{aligned}$$

- Denote $X_v = \mathcal{P}_{\Omega}(\tilde{X}_v)$, $E_v = \mathcal{P}_{\Omega}(\tilde{E}_v)$, introduce $Z_v = J_v$. For the v -th view, it derives

$$\begin{aligned} \min_{J_v, E_v, Z_v} \quad & \mathcal{R}(J_v) + \lambda \mathcal{S}(E_v) \\ \text{s.t.} \quad & X_v = X_v Z_v + E_v, \quad Z_v = J_v \end{aligned}$$

which has the augmented Lagrangian function

$$\begin{aligned} \mathcal{L}_{\beta}(J_v, E_v, Z_v, Y_1, Y_2) = & \mathcal{R}(J_v) + \lambda \mathcal{S}(E_v) + \langle Y_1, Z_v - J_v \rangle \\ & + \frac{\beta}{2} \|Z_v - J_v\|_F^2 + \langle Y_2, X_v - X_v Z_v - E_v \rangle + \frac{\beta}{2} \|X_v - X_v Z_v - E_v\|_F^2 \end{aligned}$$

Algorithm

- Update variables according to

$$\begin{cases} (J_v, E_v, Z_v) \leftarrow \arg \min_{J_v, E_v, Z_v} \mathcal{L}_\beta(J_v, E_v, Z_v, Y_1, Y_2) \\ Y_1 \leftarrow Y_1 + \beta(Z_v - J_v) \\ Y_2 \leftarrow Y_2 + \beta(X_v - X_v Z_v - E_v) \end{cases}$$

- First-order optimality conditions

$$\begin{cases} 0 \in \partial \mathcal{R}(J_v) - Y_1 + \beta(J_v - Z_v) \\ 0 \in \lambda \partial \mathcal{S}(E_v) - Y_2 - \beta(X_v - X_v Z_v - E_v) \\ 0 = Y_1 + \beta(Z_v - J_v) - X_v^\top Y_2 - \beta X_v^\top (X_v - X_v Z_v - E_v) \end{cases}$$

which derives

$$J_v = \text{Prox}_{1/\beta, \mathcal{R}}(Z_v + Y_1/\beta), \quad E_v = \text{Prox}_{\lambda/\beta, \mathcal{S}}(X_v - X_v Z_v + Y_2/\beta)$$

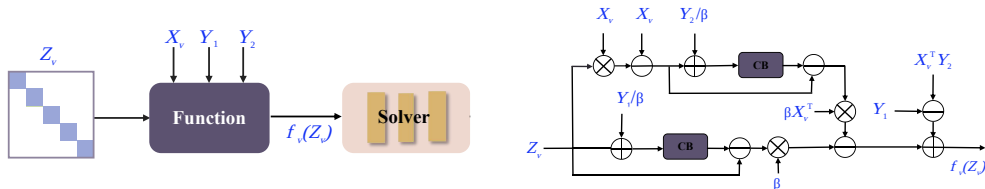
- Denote the nonlinear equation

$$f_v(Z_v) = Y_1 + \beta(Z_v - \text{Prox}_{1/\beta, \mathcal{R}}(Z_v + Y_1/\beta)) - X_v^\top Y_2 \\ - \beta X_v^\top (X_v - X_v Z_v - \text{Prox}_{\lambda/\beta, \mathcal{S}}(X_v - X_v Z_v + Y_2/\beta))$$

$$\Downarrow$$

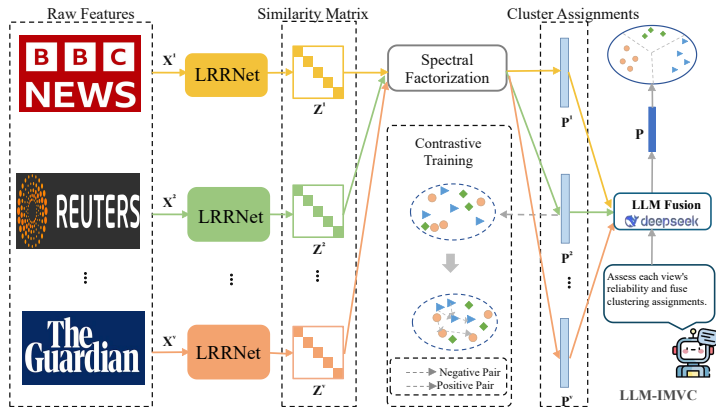
$$\begin{cases} g_v(Z_v) \Delta Z_v = -f_v(Z_v) \\ Z_v \leftarrow Z_v + \eta \Delta Z_v \end{cases}$$

- Detailed structure: NS (Newton solver), CB (convolutional block)



LLM-IMVC

► Overall architecture (to be improved)



► Contrastive learning $\Rightarrow$ end-to-end clustering

► Large language models $\Rightarrow$ adaptive clustering assignments

Experiments

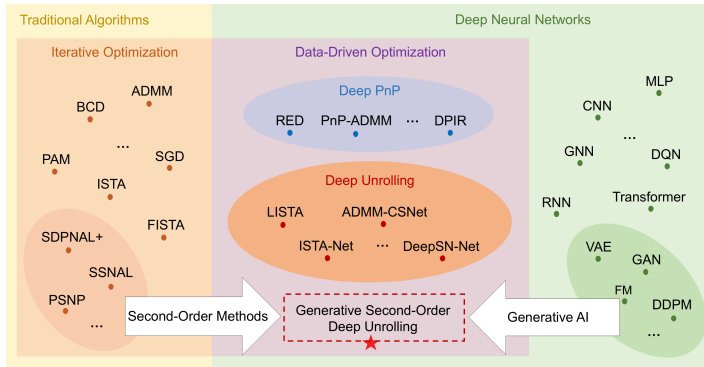
- ▶ Implementation details
 - ▶ Learnable parameters: $\mathcal{R}(J)$, $\mathcal{S}(E)$, λ , β
 - ▶ Version: DeepSeek-V4
 - ▶ Metrics: ACC ($\uparrow$)
- ▶ Clustering performance

Methods	HandWrite	BDGP	MSRC	COIL20	Caltech5V	USPS	Average
LMVSC (AAAI'20)	67.1%	55.6%	35.2%	69.2%	46.1%	56.0%	54.9%
SFMC (TPAMI'22)	75.7%	37.8%	60.0%	74.8%	46.8%	98.9%	65.7%
FPMVS-CAG (TIP'22)	82.3%	55.6%	80.5%	60.2%	80.9%	78.3%	73.0%
FDAGF (AAAI'23)	82.3%	52.5%	73.9%	74.0%	81.3%	56.1%	70.0%
RCAGL (TKDE'24)	79.4%	51.8%	77.6%	65.1%	83.6%	60.9%	69.7%
ALPC (AAAI'25)	63.2%	92.2%	64.3%	76.4%	46.1%	50.2%	65.4%
TLRLF4 (TPAMI'25)	76.4%	92.7%	—	76.3%	75.5%	99.2%	84.0%
LLM-IMVC	99.9%	100.0%	88.1%	77.2%	100.0%	99.9%	94.2%

- ▶ More numerical results are provided!

Remarks

- ▶ What is the connection between learned solvers and true Newton directions?
- ▶ More about Transformers
 - ▶ Yang-Huang-Wipf, Transformers from An Optimization Perspective, [NeurIPS](#), 2022
 - ▶ Yu-Buchanan-Pai et al, White-box Transformers via Sparse Rate Reduction, [NeurIPS](#), 2023
 - ▶ Tai-Liu-Li et al, A Mathematical Explanation of Transformers, [SIIMS](#), 2026



Outline

Introduction

Modeling Optimization

Low-Rank Optimization

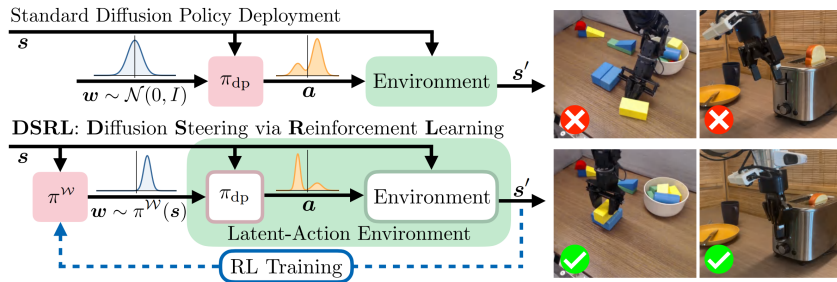
Policy Optimization

Conclusion

- ▶ Cen-Cheng-Chen et al, Fast Global Convergence of Natural Policy Gradient Methods with Entropy Regularization, [OR](#), 2022
- ▶ Li-Wei-Chi et al, Softmax Policy Gradient Methods Can Take Exponential Time to Converge, [MP](#), 2023
- ▶ Li-Gupta-Yu et al, Approximate Newton Policy Gradient Algorithms, [SISC](#), 2023
- ▶ Müller-Cayci-Montúfar, Fisher–Rao Gradient Flows of Linear Programs and State-Action Natural Policy Gradients, [SIOPT](#), 2025
- ▶ Zeng-Hong-Garcia, Structural Estimation of Markov Decision Processes in High Dimensional State Space with Finite-Time Guarantees, [OR](#), 2025
- ▶ Li-Wu-Lan, Stochastic First-Order Methods for Average-Reward Markov Decision Processes, [MOR](#), 2025
- ▶ Li-Lan, Policy Mirror Descent Inherently Explores Action Space, [SIOPT](#), 2025
- ▶ Ju-Lan, Strongly-Polynomial Time and Validation Analysis of Policy Gradient Methods, [MP](#), 2026
- ▶ Ju-Lan, Policy Optimization over General State and Action Spaces, [SIOPT](#), 2026

Motivation

- ▶ Wagenmaker-Nakamoto-Zhang et al, Steering Your Diffusion Policy with Latent Space Reinforcement Learning, [CoRL](#), 2025 (Google citation: 145)



- ▶ How to improve the action representation?
- ▶ How to develop a lightweight residual modulation mechanism?

Methodology

- How to improve the action representation?

$$(z_t, u_t) \sim \pi_\theta(\cdot \mid o_t), \quad f_t = \mathcal{A}_\omega(u_t)$$

$\Downarrow$

$$a_t = G_\phi(o_t, z_t; f_t)$$

- How to develop a lightweight residual modulation mechanism?

$$H_{t,\kappa}^{(l)} = \mathcal{T}_\phi^{(l)}(\tilde{H}_{t,\kappa}^{(l-1)}; \mathbf{e}_t, \kappa), \quad \delta_t = \lambda_{\text{inj}} \mathcal{B}(f_t)$$

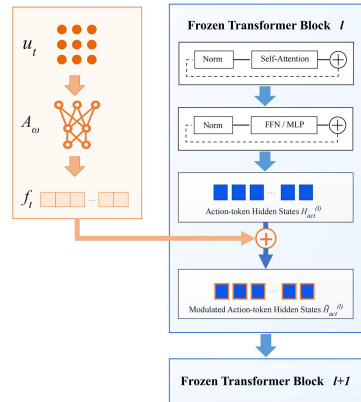
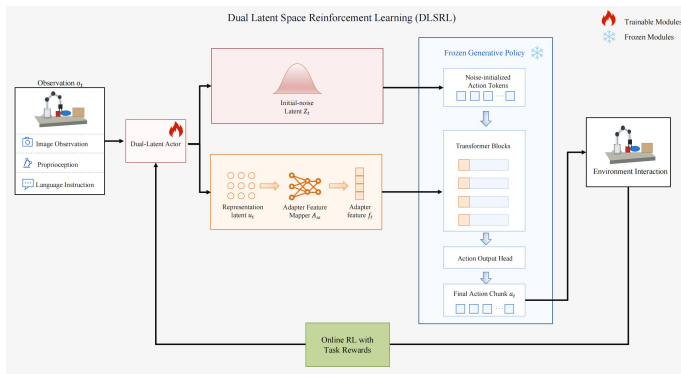
$\Downarrow$

$$\tilde{H}_{t,\kappa}^{(l)} = H_{t,\kappa}^{(l)} + \delta_t$$

- Latent-space optimization

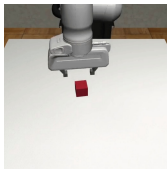
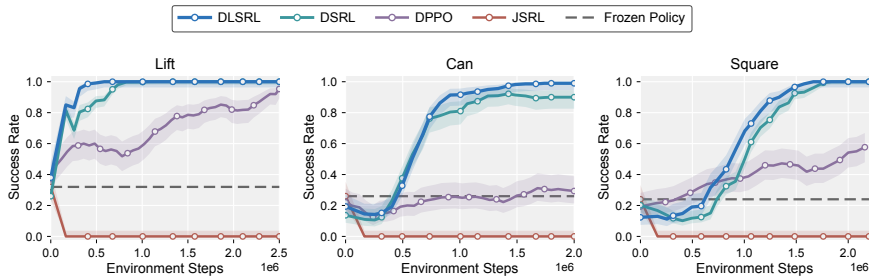
$$\mathcal{L}_{\text{actor}}(\theta, \omega) = E_{o \sim \mathcal{D}, (z, u) \sim \pi_\theta(\cdot \mid o)} \left[\alpha \log \pi_\theta^z(z \mid o) - Q_\nu^{\text{lat}}(o, \xi) \right] + \mathcal{R}_{\text{aux}}$$

► Overall architecture (to be improved)



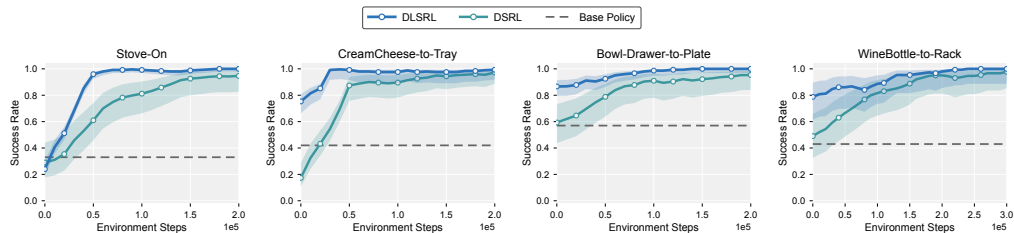
Experiments

► Diffusion-denoising policies on RoboMimic



Experiments

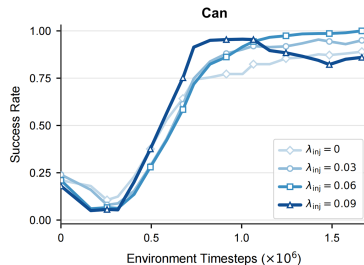
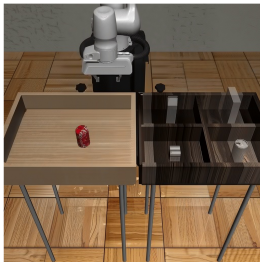
► Flow-matching VLA models on LIBERO



Tasks	DLSRL ↓	DSRL ↓	Reduction
Stove-On	65.90 ± 1.12	88.34 ± 16.42	25.4%
CreamCheese-to-Tray	157.20 ± 5.77	203.44 ± 12.58	22.7%
Bowl-Drawer-to-Plate	97.06 ± 0.22	125.80 ± 29.18	22.8%
WineBottle-to-Rack	95.96 ± 0.65	113.26 ± 8.97	15.3%
Macro Average	104.03	132.71	21.6%

Remarks

- ▶ How to develop fast and robust policy optimization?
- ▶ How to choose λ_{inj} by learning approaches?



- ▶ More experiments on real-world robots!

Outline

Introduction

Modeling Optimization

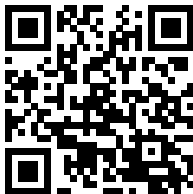
Low-Rank Optimization

Policy Optimization

Conclusion

References

- ▶ Xiu-Li-Chen et al, OptGraph: Large Language Models Enhanced Evolutionary Optimization Via Graph Retrieval-Augmented Generation, [submitted to ICLR](#)
- ▶ Chen-Wang-Xiu et al, LLM-IMVC: Large Language Model-Guided Deep Unrolling for Incomplete Multi-View Clustering, [working in process](#)
- ▶ Zhang-Xiu-Liu, Toward Efficient Robotic Control with Dual Latent Space Reinforcement Learning, [submitted to ICRA](#)



OptGraph



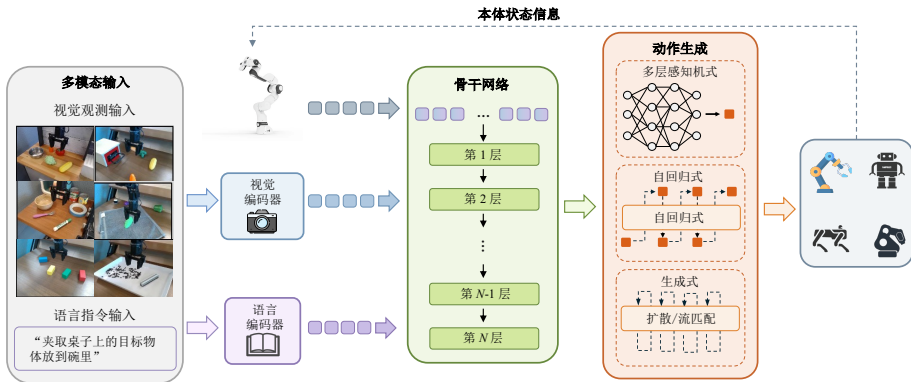
LLM-IMVC



DLSRL

Future Work

- Xiu-Xu-Zhu et al, Review of Compression Methods for Vision-Language-Action Models (in Chinese), [submitted to Acta Automatica Sinica](#)



Thank you for your attention!

xcxiu@shu.edu.cn