
MOOSEAGENT: A LLM BASED MULTI-AGENT FRAMEWORK FOR AUTOMATING MOOSE SIMULATION

Tao Zhang, Zhenhai Liu, Yong Xin, Yongjun Jiao
State Key Laboratory of Advanced Nuclear Energy Technology
Nuclear Power Institute of China
Chengdu, China
taozhan22@gmail.com

ABSTRACT

The Finite Element Method (FEM) is widely used in engineering and scientific computing, but its pre-processing, solver configuration, and post-processing stages are often time-consuming and require specialized knowledge. This paper proposes an automated solution framework, MooseAgent, for the multi-physics simulation framework MOOSE, which combines large-scale pre-trained language models (LLMs) with a multi-agent system. The framework uses LLMs to understand user-described simulation requirements in natural language and employs task decomposition and multi-round iterative verification strategies to automatically generate MOOSE input files. To improve accuracy and reduce model hallucinations, the system builds and utilizes a vector database containing annotated MOOSE input cards and function documentation. We conducted experimental evaluations on several typical cases, including heat transfer, mechanics, phase field, and multi-physics coupling. The results show that MooseAgent can automate the MOOSE simulation process to a certain extent, especially demonstrating a high success rate when dealing with relatively simple single-physics problems. The main contribution of this research is the proposal of a multi-agent automated framework for MOOSE, which validates its potential in simplifying finite element simulation processes and lowering the user barrier, providing new ideas for the development of intelligent finite element simulation software. The code for the MooseAgent framework proposed in this paper has been open-sourced and is available at <https://github.com/taozhan18/MooseAgent>.

Keywords Large Language Models, Multi-agent System, LangGraph, Moose, Automated Simulation.

1 Introduction

The Finite Element Method (FEM) has extremely broad applications in the fields of engineering and scientific computation, such as structural mechanics, heat conduction, electromagnetic fields, and multi-physics problems involving fluid-solid coupling [6, 8, 5]. As a mature and highly modular multi-physics simulation framework, MOOSE (Multiphysics Object Oriented Simulation Environment) provides researchers and engineers with a flexible numerical solution interface, automatic differentiation capabilities, and a rich library of physical couplings. It has been widely used in multiple fields, including nuclear energy, materials science, and biomechanics. However, with the increasing scale of simulations and the growing complexity of coupled models, finite element simulations typically require a significant amount of human effort and time in pre-processing (geometry and mesh generation, material and equation settings), solver configuration, and post-processing visualization. Under the traditional workflow, researchers often need to master script writing, parameter configuration, and numerical algorithms simultaneously. Any minor errors, such as improper boundary condition settings, unreasonable mesh division, or incorrect material model calls, can lead to distorted numerical results or even solver failure, posing a high barrier to entry for beginners and interdisciplinary users.

In recent years, with the rapid development of Natural Language Processing (NLP) technologies and Large Language Models (LLMs), efforts have been made to explore how to leverage these general-purpose "agents" to assist and automate scientific computation processes [7, 1, 15, 11]. In the field of Computational Fluid Dynamics (CFD), some work has attempted to combine multi-agent systems with LLMs to achieve an integrated automation process from natural

language requirements to mesh generation, solver settings, and post-processing script writing [2, 10]. Experiments have shown that by adopting the approach of "task decomposition + multi-round iterative verification," the traditional manual workflow can be transformed into semi-automated or fully automated operations carried out collaboratively by agents. This significantly reduces the dependence on users' professional background in numerical analysis and improves usability and efficiency. However, there is still a lack of similar attempts in finite element software platforms, especially in the area of multi-physics coupling modeling. The MOOSE framework also suffers from cumbersome configuration and parameter tuning processes. If an agent system could be introduced into MOOSE and extended to cater to its multi-physics characteristics, it would bring new opportunities for users with non-specialist backgrounds and interdisciplinary research.

Based on this, this paper proposes an automated solution framework integrating large language models and multi-agent systems for the MOOSE platform. The framework leverages large language models to accurately understand users' simulation needs and aligns closely with user requirements through continuous feedback. By decomposing tasks and conducting multiple rounds of iterative verification, the framework not only enhances its flexibility in handling complex simulation tasks but also fully exploits the diverse functionalities of the MOOSE platform. Moreover, by constructing a MOOSE database and incorporating retrieval augmentation techniques, the framework effectively reduces model hallucination and improves result reliability. Through a series of numerical examples covering multiple fields, including heat conduction, mechanics, phase field, and multiphysics coupling, the framework has demonstrated significant advantages in terms of executability and accuracy, while also lowering the threshold for users' expertise in the finite element method (FEM) to a certain extent.

The main contributions of this paper are as follows:

- A multi-agent system for MOOSE is proposed, which uses large language models to automate the finite element solution process from natural language, including pre-processing (mesh, material, and equation settings), solver configuration, and post-processing.
- Multi-round iterative verification is combined with task decomposition to adapt to the diversity and complexity of simulation tasks, automatically review and correct each sub-task, and improve the executability and stability of the solution process.
- Systematic tests are conducted on several typical multi-physics cases to evaluate the success rate, numerical accuracy, and requirements for computational resources and human interaction of this method, providing a feasible approach for intelligent finite element simulation software.

2 Related Work

2.1 Multi-agent Framework

Multi-agent frameworks based on LLMs are becoming key tools for complex task automation and collaborative AI systems. These frameworks can be broadly categorized into four main types based on encapsulation and automation. First, frameworks like Langchain [12] and Langgraph [3] offer foundational components, but require developers to manually code and manage agent interactions and workflows. This provides high flexibility and control for customized collaboration, but demands more development effort. Second, frameworks such as CrewAI [3], AutoGen [13], and MetaGPT [4] offer higher-level encapsulation. Users mainly define task goals and agent prompts, while the framework handles inter-agent communication, task assignment, and collaboration using predefined roles and protocols, simplifying development. Third, frameworks like Dify, Bisheng, and Coze achieve an even higher abstraction through graphical user interfaces (UI), allowing users to build and manage multi-agent workflows without coding via visual nodes and drag-and-drop configuration, suitable for non-programmers. Fourth, a more advanced research direction treats multi-agent workflows as optimizable variables [16, 17]. Analogous to supervised learning, these frameworks aim for models to automatically learn and find the best collaboration strategies and workflows, enabling dynamic adjustments based on feedback for greater efficiency and robustness. Overall, LLM-based multi-agent frameworks are progressing towards greater automation, ease of use, and intelligence, with different types catering to various needs, from finely controlled complex tasks to rapid no-code prototyping and the future of self-optimizing workflows, showing significant potential and broad applications.

Given the current technological maturity and controllability, this research chooses to adopt a multi-agent framework based on LangGraph. This framework allows for the manual writing of interaction logic between multiple agents from the bottom layer, thereby achieving highly customized and precise process control. This underlying control capability provides greater flexibility for the research, enabling fine-tuning of the interactions within the multi-agent system according to specific needs to meet the diverse requirements of complex task scenarios.

2.2 Multi-Agent-Driven Software Agents

The application of LLM-based multi-agent frameworks is rapidly emerging across various domains. Particularly in software agents, they significantly enhance software efficiency and accessibility by automating complex tasks and simplifying user interaction. For instance, AutoFLUKA [9], an AI agent application, automates Monte Carlo simulation workflows in FLUKA using the LangChain Python framework. This automation reduces manual intervention and errors, streamlining the entire process from input generation and simulation execution to results processing and visualization. In the field of Computational Fluid Dynamics (CFD), LLM agents are also demonstrating great potential. FoamPilot [14] enhances the usability of FireFOAM by providing code insight through Retrieval Augmented Generation (RAG) for efficient code navigation and summarization. It also interprets user requests in natural language to modify simulation setups and manages simulation execution in High-Performance Computing (HPC) environments, offering preliminary analysis of simulation results. MetaOpenFOAM [2] combines Chain of Thought (COT) decomposition and iterative verification, making CFD simulation and post-processing more accessible to non-expert users through natural language input. OpenFOAMGPT [10], an LLM-based agent designed for OpenFOAM, utilizes GPT-40 and a CoT-enabled model to handle complex tasks, including zero-shot case setup, boundary condition modifications, turbulence model adjustments, and code translation. These studies indicate that multi-agent-driven software agents are becoming a key technology for improving the efficiency and user experience of complex software systems.

3 Method

3.1 Overall framework

The overall framework of MooseAgent is shown in Fig.1. In the initial step, the user describes the required simulation task using natural language. The LLM parses the user's requirements, clarifies ambiguous settings, and determines the number of input cards and their specific tasks. Once the user confirms that the description meets their needs, the process moves to the simulation task architecture step. In the architecture step, the system generates retrieval content based on the simulation requirements and searches for relevant input cards in the vector knowledge base. The architect refers to these input cards to generate Moose input cards that meet the simulation requirements. After all input cards are generated, the system performs multiple rounds of error correction based on Moose's output error messages and rule-based syntax checks. If the model determines that the same error persists after multiple rounds of correction, the correction process is terminated, and error information is returned with a prompt for the architect to adopt alternative methods to avoid the error. The task ends when the input cards are successfully executed or the maximum number of iterations is reached.

3.2 Database

The database primarily comprises two components: annotated input cards and the documentation for all MOOSE functions.

We extracted over 8,000 input cards from the MOOSE repository. However, most of these input cards lack annotations, originating from sources such as test cases of each MOOSE APP and official tutorials. Due to the absence of annotations, direct retrieval of these input cards may lead to retrieval failures, or the model might misinterpret the meaning of the input cards. Therefore, we specifically designed a workflow to automatically annotate these input cards, as illustrated in Fig.2. First, unannotated input cards are randomly selected. Then, the MOOSE APPs used by these input cards are analyzed. Next, the descriptions and usage of the APPs are queried from the documentation. Finally, annotated input cards are generated using retrieval-augmented generation techniques. This annotation process is performed iteratively. All annotated input cards are stored in JSON format, including the input card name, a summary of the input card's functions, and the annotated input card content.

MOOSE's documentation system is powerful, capable of exporting the function input parameter descriptions for all APPs using the dump command. However, function input parameter descriptions alone are insufficient for a full understanding of the APP's purpose. More detailed descriptions of each APP are scattered throughout the code repository. Therefore, we developed a Python script to locate the detailed descriptions of each APP from the repository. These descriptions are also stored in JSON format, including the APP name, APP function description, and descriptions of the APP's input parameters.

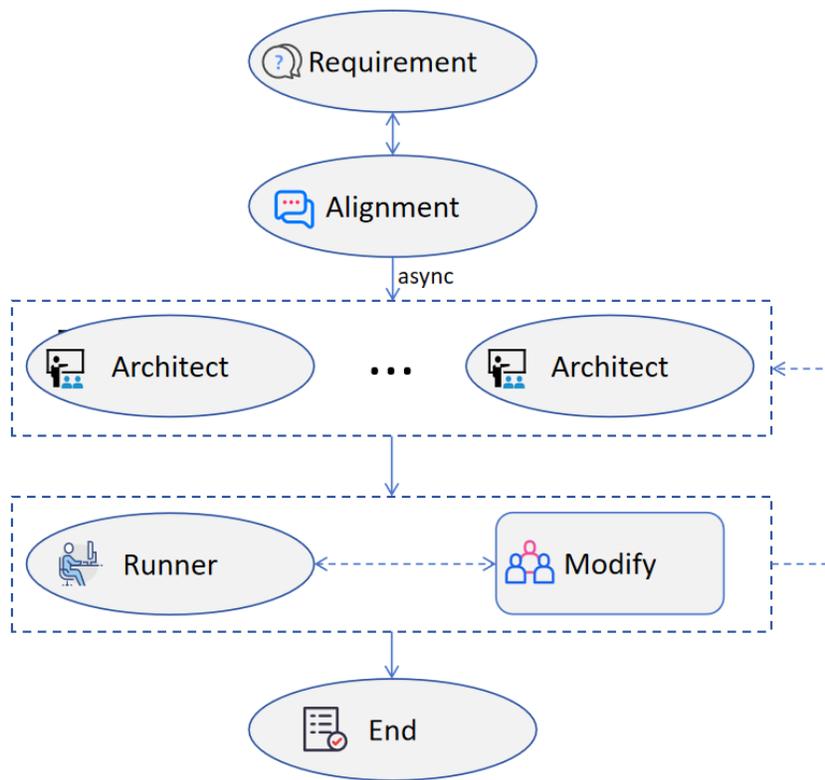


Figure 1: Overall Framework of Moose Agent

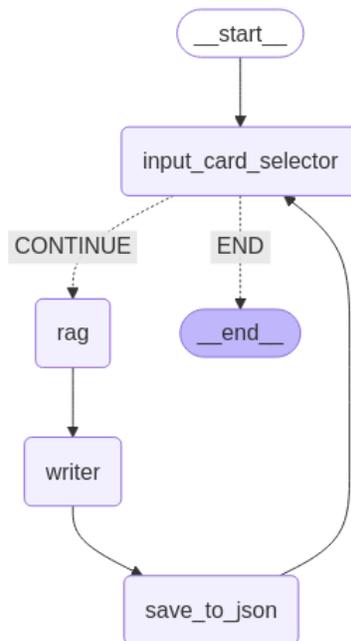


Figure 2: Automatic annotation workflow

4 Experiment

4.1 Experimental Setup

This experiment employs the langgraph framework to build a multi-agent collaborative workflow. Regarding model selection, we utilize Deepseek-R1 as the reasoning model for the overall architecture of the input card, while the remaining modules use Deepseek-V3. To reduce randomness and enhance the accuracy of the results, the temperature of all models is set to 0.01. For RAG, we have chosen the BGE-M3 embedding model and the FAISS vector database, employing similarity search for information retrieval. The maximum number of iterations in the experiment is set to 3.

To validate the effectiveness of the constructed framework, we designed the following 8 test cases:

- **Steady-State Heat Conduction (HeatSteady):** Consider a metal rod with a length of $L = 1$ m, with its two ends maintained at constant temperatures of 300 K and 350 K, respectively. The thermal conductivity k is given as 10 W/(m·K). Under this steady-state condition, the task is to calculate and output the temperature distribution along the rod, presented as a function of temperature with respect to position.
- **Transient Heat Conduction (HeatTran):** Within a $1\text{ m} \times 1\text{ m}$ square domain, the initial temperature is uniformly distributed at 300 K. The boundary conditions are as follows: the left and bottom boundaries are maintained at 300 K, the right boundary temperature is 500 K, and the top boundary is adiabatic. The material’s thermal conductivity $k = 20$ W/(m·K), specific heat capacity $c_p = 1000$ J/(kg·K), and density $\rho = 7850$ kg/m³. The experiment will simulate a transient process of 50 s, and the temperature distribution will be output every 10 s.
- **Linear Elasticity (Elasticity):** A rectangular plate with dimensions $2\text{ m} \times 1\text{ m}$ is considered, with its left side fixed and its right side subjected to a uniform tensile stress of 5 MPa. The material is linearly elastic, with an elastic modulus $E = 200$ GPa and a Poisson’s ratio $\nu = 0.3$. The objective is to calculate the stress and strain fields within the plate and output the maximum principal stress and strain values.
- **Plastic Strain (Plasticity):** Consider a $1\text{ m} \times 1\text{ m}$ square plate with its left edge fixed and a horizontal displacement of 0.01 m applied to its right edge. The material exhibits elastoplastic behavior, described by a bilinear hardening model with the following parameters: yield stress $\sigma_y = 250$ MPa, elastic modulus $E = 210$ GPa, and hardening modulus $H = 1$ GPa. The experiment requires outputting the final plastic strain distribution and identifying the regions where plastic deformation has occurred.
- **Porous Media Flow (Porous):** A cubic soil block with dimensions $1\text{ m} \times 1\text{ m} \times 1\text{ m}$ is considered, with a water head of 5 m at the bottom and 1 m at the top, and impermeable side boundaries. The permeability of the soil is $k = 10^{-5}$ m/s. The task is to calculate the steady-state flow field and pressure distribution within the soil block.
- **Phase Change Heat Conduction (PhaseChange):** A 0.1-meter-long metal rod has one end maintained at 300 K and the other end at 200 K. The melting/freezing temperature is 250 K, with a latent heat $L = 2 \times 10^5$ J/kg. The specific heat capacity c_p is the same for both the solid and liquid phases, and the thermal conductivities are $\kappa_s = 30$ (solid) and $\kappa_L = 15$ (liquid). The simulation runs for 500 seconds to track the movement of the phase change front over time.
- **Phase Field (PhaseField):** This case aims to simulate the solidification process of a pure metal within a two-dimensional rectangular region. The phase-field model is employed to simulate the solid-liquid phase transition by solving the coupled evolution equations for the phase-field variable and the temperature field. The boundary condition involves applying a low temperature below the solidification point on one side of the rectangular region to drive solidification, with the initial condition set as the metal being entirely in a liquid state. The experimental goal is to observe the formation and growth of the solid phase, as well as the evolution of the solid-liquid interface and the temperature distribution.
- **Thermal-Mechanical Coupling (ThermalMechanic):** This case utilizes the Multiapp functionality to simulate the thermal-structural coupling behavior of a two-dimensional rectangular thin plate. In the thermal analysis part, one side of the thin plate is heated, while the other side is kept at a low temperature. In the mechanical analysis part, one side of the thin plate is fixed, and the temperature distribution calculated by the main application (thermal analysis) is used as a thermal load to analyze the thermal expansion and displacement of the thin plate.

We employ the following 3 evaluation metrics to assess the experimental results:

- **Success Rate:** Defined as the ratio of the number of successfully run test cases to the total number of test cases.

- **Token Usage:** Refers to the total number of tokens consumed by the large models during the experiment.
- **Productivity:** Defined as the ratio of Token Usage to the number of code characters generated.

““latex

4.2 Experimental Results Analysis

Table 1 summarizes the performance of MooseAgent in eight different test cases. In terms of success rate, the Steady State Heat Conduction (HeatSteady), Linear Elasticity (Elasticity), and Phase Field (PhaseField) cases all achieved a 100% success rate, indicating that MooseAgent performs excellently in handling these relatively mature physical problems with many reference cases. The success rate for Transient Heat Conduction (HeatTran) is 80%, suggesting that there is still room for improvement in the framework’s robustness when dealing with time-dependent complex problems. The success rate for Plastic Strain (Plasticity) is 60%, indicating that the framework has a certain ability to handle nonlinear material behavior. The success rates for Porous Media Flow (Porous) and Phase Change Heat Transfer (PhaseChange) are both 60%, indicating that MooseAgent still has some ability to handle fluid flow and problems involving changes of state, but may be sensitive to certain specific parameters or boundary conditions. It is worth noting that the Thermal-Mechanic coupling (ThermalMechanic) case has a lower success rate of 40%, mainly because thermal-mechanic coupling problems involve the strong coupling of two sub-problems, thermal analysis and mechanical analysis, and the model configuration and parameter settings are more complex, which places higher demands on the agent’s understanding and collaborative processing capabilities for multi-physics problems.

In terms of Token usage, there are significant differences between different cases. The Phase Field (PhaseField) case has the highest Token usage, reaching 86696, which may be related to its need for longer simulation times and more complex input parameter configurations. The Token usage for Porous Media Flow (Porous) and Thermal-Mechanic coupling (ThermalMechanic) cases are also relatively high, at 79176 and 77020, respectively. Relatively speaking, the Token usage for Steady State Heat Conduction (HeatSteady) and Plastic Strain (Plasticity) cases are lower, at 24673 and 15874, respectively, which may be related to the smaller scale and lower complexity of the problems.

The productivity metric reflects the number of Tokens consumed per code character generated, with lower values typically indicating higher efficiency. As can be seen from Table 1, the Porous Media Flow (Porous) and Phase Field (PhaseField) cases have the highest productivity, at 39 and 40 respectively, which may mean that in these two cases, the agent outputted a relatively large amount of effective code when generating the input card. The productivity for the Plastic Strain (Plasticity) case is 8, and the productivity for the Thermal-Mechanic coupling (ThermalMechanic) case is 22. These values are relatively low, possibly indicating that the agent made more attempts and corrections when solving these more complex problems, resulting in higher Token consumption but a relatively small amount of effective code generated in the end.

Table 1: Performance of MooseAgent

Case	Pass	Token	Productivity
HeatSteady	1	24673	28
HeatTran	0.8	37695	24
Elasticity	1	40857	16
Plasticity	0.6	15874	8
PhaseChange	0.6	23742	10
Porous	0.6	79176	39
PhaseField	1	86696	40
ThermalMechanic	0.4	77020	22

5 Conclusion

This paper proposes an automated solution framework MooseAgent for the MOOSE platform, which adopts a multi-agent system that utilizes large language models to understand user requirements and generates executable MOOSE input cards through task decomposition and multi-round iterative verification. MooseAgent incorporates MOOSE knowledge stored in a vector database to enhance retrieval and reduce hallucinations. Experimental results show that the framework has achieved high success rates for problems such as steady-state heat conduction and linear elasticity, verifying its potential in automated finite element analysis. However, when dealing with complex multi-physics

problems such as thermal-mechanic coupling, there is still room for improvement in the success rate. The main contribution of this research is to explore the automated process from natural language to MOOSE simulation based on technologies such as LLM, multi-agent systems, task decomposition, and multi-round iterative verification. Future work will optimize knowledge retrieval and iteration strategies, and consider incorporating human feedback during the iteration process to further improve the framework's performance on complex problems.

References

- [1] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.
- [2] Yuxuan Chen, Xu Zhu, Hua Zhou, and Zhuyin Ren. Metaopenfoam 2.0: Large language model driven chain of thought for automating cfd simulation and post-processing. *arXiv preprint arXiv:2502.00498*, 2025.
- [3] Zhihua Duan and Jialin Wang. Exploration of llm multi-agent application implementation based on langgraph+crewai. *arXiv preprint arXiv:2411.18241*, 2024.
- [4] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4):6, 2023.
- [5] Casey Icenhour, Shane Keniley, Corey DeChant, Cody Permann, Alex Lindsay, Richard Martineau, Davide Curreli, and Steven Shannon. Multi-physics object oriented simulation environment (moose). Technical report, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2018.
- [6] PC Kohnke. Ansys. In *Finite element systems: a handbook*, pages 19–25. Springer, 1982.
- [7] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.
- [8] COMSOL Multiphysics. Introduction to comsol multiphysics®. *COMSOL Multiphysics, Burlington, MA, accessed Feb*, 9(2018):32, 1998.
- [9] Zavier Ndum Ndum, Jian Tao, John Ford, and Yang Liu. Autofluka: A large language model based framework for automating monte carlo simulations in fluka. *arXiv preprint arXiv:2410.15222*, 2024.
- [10] Sandeep Pandey, Ran Xu, Wenkang Wang, and Xu Chu. Openfoamgpt: a rag-augmented llm agent for openfoam-based computational fluid dynamics. *arXiv preprint arXiv:2501.06327*, 2025.
- [11] Yingheng Tang, Wenbin Xu, Jie Cao, Jianzhu Ma, Weilu Gao, Steve Farrell, Benjamin Erichson, Michael W Mahoney, Andy Nonaka, and Zhi Yao. Matterchat: A multi-modal llm for material science. *arXiv preprint arXiv:2502.13107*, 2025.
- [12] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, 2023.
- [13] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- [14] Leidong Xu, Danyal Mohaddes, and Yi Wang. Llm agent for fire dynamics simulations. *arXiv preprint arXiv:2412.17146*, 2024.
- [15] Sherry Yang, Simon Batzner, Ruiqi Gao, Muratahan Aykol, Alexander Gaunt, Brendan C McMorro, Danilo Jimenez Rezende, Dale Schuurmans, Igor Mordatch, and Ekin Dogus Cubuk. Generative hierarchical materials search. *Advances in Neural Information Processing Systems*, 37:38799–38819, 2024.
- [16] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.
- [17] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.