Latent Graph Diffusion for Automatic Multi-Agent Communication Topology Generation

Anonymous CICAI submission

Paper ID 69

Abstract. Communication topology critically affects both task performance and token cost in LLM-based multi-agent systems, yet existing methods often rely on fixed structures or expensive inference-time search. We propose a latent graph diffusion framework for task-adaptive communication topology generation with role assignment handled as an integrated execution component. First, a graph autoencoder maps discrete communication topologies into a continuous latent space, where a conditional diffusion model generates task-specific communication graphs. Second, contrastive preference learning internalizes the correctness-cost trade-off by pulling denoised latents toward correct low-cost topologies and pushing them away from two types of negative samples: correct but redundant topologies and incorrect but overly simple topologies. Third, Triplet-CFG combines task guidance and negative guidance at inference time to steer generation without explicit topology search. Experiments on MATH-500, GSM8K, HumanEval, and MBPP show the best accuracy on three of four benchmarks and competitive performance on HumanEval; on MATH-500, our method improves over the closest topology-generation baseline GTD by 3.78 points.

Keywords: Multi-agent System · Communication Topology Generation · Graph Diffusion Model · Contrastive Learning · Large Language Model

1 Introduction

Large language models (LLMs) such as GPT-4 [13] and DeepSeek [3] have demonstrated remarkable capabilities across diverse tasks including mathematical reasoning, code generation, and natural language understanding. Building upon these capabilities, recent research has shifted toward organizing multiple LLM-based agents into collaborative systems [19, 7, 14], where agents with specialized roles communicate to solve complex tasks that challenge individual models. In such multi-agent systems (MAS), the *communication topology*—the directed graph defining which agents communicate with which others—critically determines both task performance and operational cost (measured in token consumption).

Existing frameworks predominantly employ fixed topologies: sequential chains [18], dense debate graphs [4], or star topologies with central coordinators [19]. These

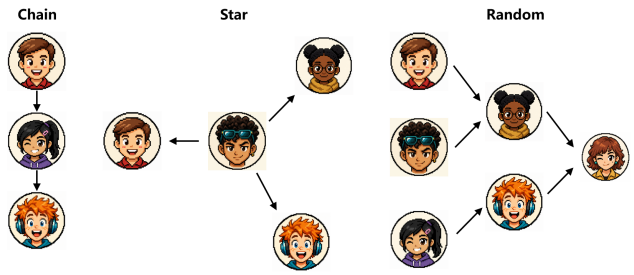

Fig. 1. Representative communication topologies. Chain and star impose fixed information-flow patterns, while random graphs add structural diversity without task conditioning.

static structures cannot adapt to varying task characteristics—a dense debate topology wastes tokens on trivial problems, while a simple chain may fail on problems requiring diverse perspectives and iterative verification. Fig. 1 illustrates representative communication patterns, including chain, star, and random graph topologies. Chain and star structures impose rigid communication rules, whereas random graphs provide structural diversity but remain task-agnostic. Recent adaptive methods address this partially: G-Designer [23] generates topologies via VGAE; AgentPrune [22] prunes edges from dense graphs but is constrained by the initial topology; GTD [10] uses discrete graph diffusion but requires expensive zero-order optimization at inference time (8–16 extra forward passes) to balance quality and cost.

We identify three key limitations of existing approaches: (1) fixed or search-based topologies cannot directly generate task-specific communication graphs in a scalable manner; (2) no internalized quality-cost trade-off mechanism exists, relying instead on costly inference-time optimization; (3) discrete diffusion approaches cannot leverage the smooth gradient landscape of continuous latent spaces for efficient generation.

To address these limitations, we propose a *latent graph diffusion* framework with the following contributions:

- We use latent graph diffusion for automatic multi-agent communication topology generation. A graph autoencoder embeds discrete communication graphs into a continuous latent space, enabling task-adaptive graph generation with efficient continuous denoising, while role assignment is incorporated as a supporting component rather than the primary contribution.
- We introduce contrastive preference learning to internalize the correctness-cost trade-off. Triplet fine-tuning pulls generated latents toward correct low-cost topologies and pushes them away from two complementary negative

types: correct but redundant topologies and incorrect but overly simple topologies.
- We propose Triplet-CFG, an inference strategy that combines task guidance with negative guidance to steer generation away from undesirable topology patterns. Experiments show the best accuracy on three of four benchmarks and the lowest measured GSM8K cost among cost-evaluated methods.

2 Related Work

LLM-based Multi-Agent Systems. Recent frameworks organize LLM agents into collaborative systems with specialized roles. AutoGen [19] provides flexible multi-agent conversations, MetaGPT [7] assigns software engineering roles with standardized procedures, and ChatDev [14] simulates a software company workflow. LLM-Debate [4] shows that multi-agent debate improves factuality via peer verification. These systems employ fixed topologies. DyLAN [12] introduces dynamic agent selection within a fixed structure, GPTSwarm [25] optimizes connections via gradient methods, and ADAS [8] uses meta-learning for architecture discovery. However, none generate topologies conditioned on specific task inputs.

Adaptive Topology Generation. G-Designer [23] trains a VGAE conditioned on task descriptions to generate adjacency matrices but handles role assignment independently through a separate classification head. AgentPrune [22] formulates topology optimization as graph pruning from an initial complete graph, which constrains the search space. AYC [11] proposes autoregressive node-by-node generation but suffers from error accumulation. AFlow [24] applies MCTS to explore workflow configurations. GTD [10] is most closely related, applying discrete graph diffusion (DiGress-style) to topology generation, but requires zero-order optimization at inference time to balance quality and cost (8–16 extra forward passes per sample), making it impractical for real-time deployment.

Graph Generation with Diffusion Models. DDPM [6] and score-based models [15] have been extended to graph-structured data. DiGress [16] operates in discrete graph space using categorical diffusion, preserving the discrete nature of adjacency matrices but sacrificing the smooth optimization landscape. An alternative approach embeds graphs into continuous latent spaces via autoencoders, benefiting from well-established continuous diffusion theory and natural handling of variable-sized graphs. Our work follows this continuous latent approach and adapts it to task-conditional multi-agent communication topology generation.

3 Method

3.1 Problem Formulation and Overview

Given a task query q, our goal is to generate a communication graph $G = (V, E, R)$ where $V = \{v_1, \ldots, v_n\}$ is the set of agent nodes, $E \subseteq V \times V$ is the

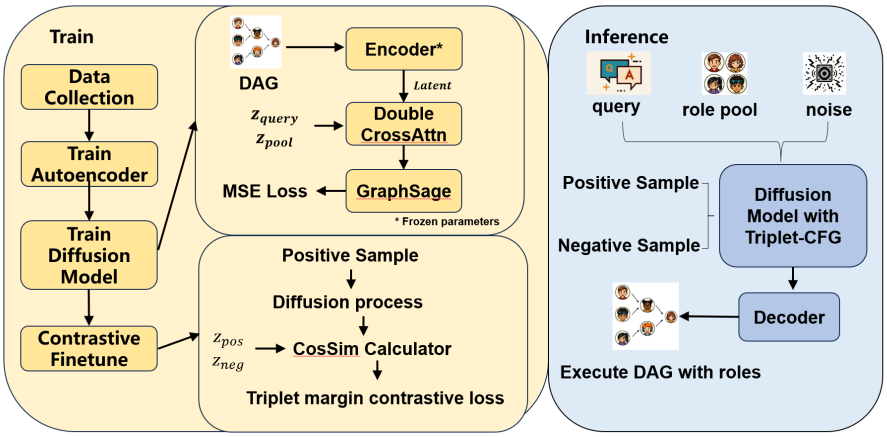

Fig. 2. Overview of the proposed training and inference pipeline. During training, the diffusion model is first trained in the latent graph space and then contrastively fine-tuned to pull denoised latents toward correct low-cost topologies and push them away from correct-but-redundant or incorrect-but-simple negative topologies. During inference, Triplet-CFG combines unconditional, task-conditional, and negative guidance to generate a task-adaptive topology with role assignments from a predefined role pool. The generated symmetric adjacency matrix is converted into an executable DAG before agent execution. Asterisks (*) denote frozen parameters.

set of communication edges, and $R : V \to \mathcal{R}$ maps each active agent to a role from a predefined role pool $\mathcal{R} = \{r_1, \ldots, r_K\}$. Following our implementation, the generative model first produces a symmetrized undirected adjacency matrix, which is then oriented into an executable DAG by a cycle-checking procedure during multi-agent execution. The main generative target is the communication topology, with role assignment used to instantiate the generated agents during execution. The objective is:

$$G^* = \arg\max_{G} \; Q(G, q) - \lambda \cdot C(G) \tag{1}$$

where $Q(G, q)$ measures the output quality when topology G is used to solve query q, and $C(G)$ represents the communication cost proportional to the number of edges and nodes.

Our framework has three central components (Fig. 2): latent diffusion for topology generation with role instantiation, contrastive preference learning for the correctness-cost trade-off, and Triplet-CFG for guided inference. Training begins with systematic multi-agent execution over communication topologies of 2–8 agents. Each (topology, query) pair is labeled as positive, negative, or neutral according to a per-query quality-cost score.

3.2 Graph Autoencoder

The graph autoencoder maps each padded topology $G = (A, m, R)$ into node-level latents $Z \in \mathbb{R}^{N \times d}$ for diffusion-based generation, where A is symmetrized and self-loops are removed. The encoder is a 3-layer GIN [20] with hidden dimension 128 and latent dimension $d = 64$. The decoder reconstructs edge existence with a bilinear adjacency head $\hat{A}_{ij} = \sigma(z_i^\top W_{\text{edge}} z_j)$, node existence with an MLP head $\hat{m}_i$, and role assignments by projecting z_i into the role-embedding space and matching it to the closest role prompt by cosine similarity. The autoencoder is trained with BCE losses for adjacency and node masks, together with a cosine role-matching loss.

3.3 Conditional Latent Diffusion Model

With the trained autoencoder providing a smooth latent space, we train a conditional diffusion model to generate task-specific topologies by producing latent vectors Z conditioned on the task query q. Following DDPM [6], the forward process gradually adds Gaussian noise over $T = 1000$ timesteps with a linear schedule ($\beta_1 = 10^{-4}$ to $\beta_T = 0.02$). Using the reparameterization with $\bar{\alpha}_t = \prod_{s=1}^{t}(1 - \beta_s)$:

$$Z_t = \sqrt{\bar{\alpha}_t} Z_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{2}$$

The denoiser $\epsilon_\theta(Z_t, t, c_q, c_\mathcal{R})$ predicts the noise component from noisy latent Z_t, timestep t, query condition c_q, and role-pool condition $c_\mathcal{R}$. We obtain c_q with a frozen sentence transformer and represent $c_\mathcal{R}$ as embeddings of the K role prompts. The SAGEConv [5] backbone operates over all padded node positions and uses cross-attention so each node can attend to both the query and the role pool, with sinusoidal timestep embeddings injected into each layer. The training loss is the standard noise prediction objective:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{Z_0, \epsilon, t}\left[\|\epsilon - \epsilon_\theta(Z_t, t, c_q, c_\mathcal{R})\|^2\right] \tag{3}$$

where $Z_0 = \text{Enc}(G)$ is the encoded latent of a training topology and $t \sim \text{Uniform}(1, T)$.

3.4 Contrastive Learning Fine-tuning

A key innovation is the contrastive fine-tuning stage that internalizes quality-cost trade-off directly into model parameters, reducing the need for explicit inference-time topology optimization [10]. From training data, we assign each (topology, query) pair a quality-cost score based on answer correctness and topology complexity. For each query, samples whose scores are close to the best candidate are treated as **positive** samples; **negative** samples include two complementary failure modes: (i) correct but redundant topologies that solve the task with unnecessary communication, and (ii) incorrect but overly simple topologies that reduce cost at the expense of answer quality. The remaining neutral samples

are retained for diffusion training but excluded from triplet construction. This per-query thresholding accounts for varying difficulty levels.

For each training query q, we sample a positive latent $Z_0^+ = \mathrm{Enc}(G^+)$ and a negative latent $Z_0^- = \mathrm{Enc}(G^-)$, compute the predicted clean latent $\hat{Z}_0$ using the x_0-prediction reparameterization from noisy Z_t, and apply a cosine-similarity triplet margin loss:

$$\mathcal{L}_{\mathrm{CL}} = \max\left(0,\ m - s_{\mathrm{pos}} + s_{\mathrm{neg}}\right), \tag{4}$$

where $s_{\mathrm{pos}} = \mathrm{mean}(\mathrm{CosSim}(\hat{Z}_0, Z_0^+))$, $s_{\mathrm{neg}} = \mathrm{mean}(\mathrm{CosSim}(\hat{Z}_0, Z_0^-))$, and $m = 0.3$ is the margin hyperparameter. This loss encourages the denoiser to predict latents with higher cosine similarity to high-quality compact topologies and lower similarity to both wasteful correct topologies and under-communicative incorrect topologies. The total fine-tuning loss is:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{diff}} + \lambda_{\mathrm{CL}} \cdot \mathcal{L}_{\mathrm{CL}} \tag{5}$$

with $\lambda_{\mathrm{CL}} = 0.1$ balancing the two objectives.

3.5 Triplet-CFG Inference

At inference time, we introduce *Triplet Classifier-Free Guidance* (Triplet-CFG), which extends standard CFG with an additional negative guidance signal. This step is separate from contrastive fine-tuning: the two negative topology types above define the training triplets, whereas Triplet-CFG uses a negative query condition selected from low-scoring training queries.

For each denoising step, we compute three noise predictions: unconditional ($\epsilon_{\mathrm{uncond}}$, with query condition replaced by a zero embedding), conditional (ϵ_{cond}, with the actual query), and negative (ϵ_{neg}, conditioned on c_q^-). The negative condition c_q^- is obtained by encoding training queries whose average composite scores fall in the bottom 25% and contain failed executions. The final guided prediction is:

$$\hat{\epsilon} = \epsilon_{\mathrm{uncond}} + w^+(\epsilon_{\mathrm{cond}} - \epsilon_{\mathrm{uncond}}) - w^-(\epsilon_{\mathrm{neg}} - \epsilon_{\mathrm{uncond}}) \tag{6}$$

with $w^+ = 1.5$ and $w^- = 0.5$. The positive weight amplifies the task-conditional signal, while the negative weight repels the denoising trajectory from low-performing query conditions. This negative condition is fixed during inference and should not be confused with the positive/negative topology pairs used in contrastive fine-tuning.

After reverse diffusion yields clean latents $\hat{Z}_0$, we decode through the autoencoder decoder: edge existence by thresholding $\hat{A}_{ij} > 0.5$, node existence by $\hat{m}_i > 0.5$, and roles by cosine similarity matching. The decoded adjacency is symmetrized, self-loops are removed, and a cycle-checking orientation procedure assigns directions to edges during execution; edges that would create cycles are flipped or removed, yielding a valid DAG execution order.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on four benchmarks spanning mathematical reasoning and code generation: MATH-500 [17] (500 competition-level math problems), GSM8K [18] (1,319 grade-school math problems), HumanEval [2] (164 Python problems with unit tests), and the 427-problem sanitized MBPP subset [1]. For each benchmark, we reserve a small training split only for topology-label collection (50 queries for MATH-500 and GSM8K, 40 for HumanEval and MBPP) and use the remaining problems exclusively for final evaluation (450, 1269, 124, and 387 problems respectively).

Baselines. We compare against methods in four categories: (1) *single-agent*: Vanilla prompting, CoT [18], Self-Consistency [17]; (2) *fixed topology*: LLM-Debate [4], LLM-Blender [9]; (3) *search/optimization*: DyLAN [12], GPTSwarm [25], ADAS [8], AFlow [24], MAAS [21]; (4) *generative*: AYC [11], GTD [10].

Implementation Details. Unless otherwise specified, all multi-agent methods use GPT-4o mini as the backbone LLM with $K = 4$ predefined roles (*MathSolver*, *ProblemDecomposer*, *CalculationChecker*, *ProgrammingExpert*) and maximum $N_{\max} = 8$ nodes. Role descriptions are encoded using SentenceTransformer (all-MiniLM-L6-v2, dim 384). For data collection, we enumerate 4 agent sizes ($n \in \{2, 4, 6, 8\}$) with 6 topology variants each (3 random graphs and 3 fixed structures: chain, star, and fully connected graph), yielding $n_{\text{train}} \times 4 \times 6$ (topology, query) pairs per benchmark (where n_{train} is the number of training queries). Each pair is executed 4 times to account for LLM stochasticity, resulting in 4,320 unique pairs and 17,280 total executions. The graph autoencoder uses 3-layer GIN (hidden 128, latent 64), trained for 200 epochs with lr 10^{-3}. The diffusion model uses 3-layer SAGEConv denoiser, $T = 1000$, trained with Adam (lr 10^{-4}). Contrastive fine-tuning uses $\lambda_{\text{CL}} = 0.1$, margin $m = 0.3$, for 5000 steps with lr 5×10^{-5}. All training completes in $\sim$30 minutes on a single RTX 4090.

4.2 Main Results

Table 1 summarizes the main evaluation across mathematical reasoning and code generation benchmarks. Overall, our method achieves the best accuracy on three of the four benchmarks (MATH-500, GSM8K, and MBPP) and remains competitive on HumanEval, where MAAS obtains the best result through meta-search over customized architectures. Among topology-generation baselines, our method consistently outperforms AYC and GTD on all four benchmarks.

The gains vary across task types. On mathematical reasoning, the largest absolute improvement appears on MATH-500 (+3.78 points over GTD), suggesting that competition-level problems benefit more from task-adaptive communication. On GSM8K, our method obtains comparable accuracy to GTD with a small 0.11-point gain, which is expected because many grade-school arithmetic

Table 1. Main results on mathematical reasoning and code generation benchmarks. Best results are in **bold**, second best are underlined. Unless otherwise specified, methods use GPT-4o mini as the backbone LLM.

Category	Method	MATH-500	GSM8K	HumanEval	MBPP
Single-agent	Vanilla	60.20	87.45	87.08	71.83
	CoT	62.67	87.10	88.13	72.00
	Self-Consistency	62.22	87.57	88.60	73.60
Fixed topology	LLM-Debate	61.11	89.47	88.68	70.29
	LLM-Blender	65.55	88.35	88.80	77.05
Search/Optim.	DyLAN	64.20	89.98	90.42	77.30
	GPTSwarm	61.55	89.14	89.32	77.43
	ADAS	58.22	86.12	84.19	68.13
	AFlow	64.20	91.16	90.93	81.67
	MAAS	65.80	92.30	**92.85**	82.17
Generative	AYC	64.00	93.90	91.74	82.80
	GTD	68.44	94.14	91.46	82.50
	Ours	**72.22**	**94.25**	91.93	**83.71**

Table 2. Results with different backbone LLMs (Acc. %).

Dataset	Method	Qwen3-32B	LLaMA-4-Scout	GPT-4o mini
GSM8K	Vanilla	82.49	94.92	87.45
	Ours	**94.63**	**95.89**	**94.25**
MBPP	Vanilla	83.84	83.13	71.83
	Ours	**85.01**	**83.71**	**83.71**

problems can already be solved by simpler communication patterns. On code generation, the margins are smaller: our method is competitive on HumanEval and achieves the best MBPP result. The poor MBPP performance of LLM-Debate further suggests that inappropriate communication topologies can be actively harmful, motivating topology generation rather than fixed communication rules.

4.3 Backbone Transfer

Table 2 reports preliminary transfer results with different backbone LLMs. Across GSM8K and MBPP, our topology generation method improves over vanilla single-agent prompting for all evaluated backbones, suggesting that the learned topology-generation strategy is not specific to GPT-4o mini.

The gain depends on the strength of the underlying backbone. When the backbone is already strong, such as LLaMA-4-Scout on GSM8K (94.92% with vanilla prompting), the marginal benefit of multi-agent collaboration is relatively small (+0.97 points). A similar pattern appears on MBPP, where LLaMA-4-Scout improves from 83.13% to 83.71% (+0.58 points). This is expected be-

Table 3. Communication cost analysis on GSM8K. Prompt and Completion tokens are in millions (M). Cost is computed using GPT-4o mini API pricing at the time of evaluation.

Method	Prompt (M)	Completion (M)	Cost ($)	Acc (%)
GPTSwarm	7.29	2.11	2.36	89.14
MAAS	3.51	2.28	1.90	92.30
AYC	2.75	1.82	1.51	93.90
GTD	2.62	1.81	1.48	94.14
Ours	**2.64**	**1.76**	**1.45**	**94.25**

cause many medium-difficulty arithmetic and programming tasks can already be solved by a single strong model, leaving less room for additional communication. In contrast, with a weaker backbone such as Qwen3-32B on GSM8K, our method improves accuracy from 82.49% to 94.63% (+12.14 points), indicating that adaptive topology generation can better compensate for weaker single-agent reasoning. These results support the adaptive nature of our method: it can exploit multi-agent communication when collaboration is useful while avoiding excessive collaboration on easier cases.

4.4 Communication Cost Analysis

Table 3 presents a detailed cost comparison on GSM8K for representative optimization and topology-generation methods. Our method achieves the lowest measured cost ($1.45) while attaining the highest accuracy in this comparison (94.25%). Compared to GPTSwarm, it uses substantially fewer tokens while improving accuracy by 5.11 points. This lower measured cost is consistent with the effect of contrastive fine-tuning, which trains the model to prefer compact topologies—on simpler GSM8K problems, it often generates sparse 2-node topologies, avoiding unnecessary agent overhead. GTD achieves comparable cost ($1.48) and comparable accuracy, while our method does not require explicit inference-time topology search or zero-order optimization.

4.5 Ablation Study

Table 4 presents ablation results on MATH-500 to evaluate the contribution of each proposed component.

Removing the diffusion model entirely (using direct AE sampling from random noise) causes the largest drop (-6.22 points), confirming that the iterative denoising process is essential for generating high-quality topologies. Removing contrastive learning leads to -2.89 points, as the model loses its internalized quality-cost trade-off and tends to generate either overly simple or overly complex topologies. Replacing Triplet-CFG with standard CFG reduces performance by 1.55 points, demonstrating the value of negative guidance. Removing role assignment and using random roles post-generation decreases accuracy by 1.33

Table 4. Ablation study on MATH-500. Each row removes one component from the full model.

Variant	Accuracy (%)	Cost ($)	Δ Acc.
Full model	**72.22**	0.71	–
w/o contrastive learning	69.33	0.78	−2.89
w/o Triplet-CFG	70.67	0.73	−1.55
w/o role assignment	70.89	0.71	−1.33
w/o diffusion (AE only)	66.00	0.81	−6.22

points, showing that role instantiation is a useful component even though the main contribution lies in topology generation.

Topology Adaptation. We analyze the generated topology statistics across benchmarks. The average topology for GSM8K (simpler arithmetic) has 1.93 nodes and 1.16 edges, while for MATH-500 (competition-level) it has 3.53 nodes and 3.20 edges, confirming that the model allocates communication resources proportionally to task difficulty. Role distribution also changes across domains: *MathSolver+CalculationChecker* appear more often in math tasks, while *ProgrammingExpert+ProblemDecomposer* are more frequent in code generation tasks. This adaptive sparsity supports the core motivation that different tasks require different communication patterns rather than a single fixed topology.

5 Limitations

Our main comparison uses GPT-4o mini as the backbone LLM; although preliminary transfer experiments with Qwen3-32B and LLaMA-4-Scout suggest that the method is not tied to a single backbone, a broader evaluation over more open and closed models remains future work. The current implementation uses a predefined role pool with $K = 4$ roles; extending the role set or learning open-ended hierarchical roles remains future work. Finally, although inference avoids explicit topology search, the data collection stage still requires repeated multi-agent executions to obtain quality-cost labels.

6 Conclusion

We proposed a latent graph diffusion framework for automatic multi-agent communication topology generation with integrated role assignment. The method combines diffusion-based topology generation, contrastive preference learning for the correctness-cost trade-off, and Triplet-CFG with both task and negative guidance. Experiments show the best accuracy on MATH-500, GSM8K, and MBPP, competitive HumanEval performance, the lowest measured GSM8K cost among cost-evaluated methods, and consistent gains in preliminary backbone transfer experiments. Future work includes dynamic topologies, broader backbone evaluation, and learnable or hierarchical role assignment.

References

1. Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al.: Program synthesis with large language models. arXiv preprint arXiv:2108.07732 (2021)
2. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)
3. DeepSeek-AI: DeepSeek-V3 technical report. arXiv preprint arXiv:2412.19437 (2024)
4. Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I.: Improving factuality and reasoning in language models through multiagent debate. In: Forty-first International Conference on Machine Learning (2024)
5. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. pp. 1025–1035 (2017)
6. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020)
7. Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S.K.S., Lin, Z., et al.: MetaGPT: Meta programming for a multi-agent collaborative framework. In: The Twelfth International Conference on Learning Representations (2024)
8. Hu, S., Lu, C., Clune, J.: Automated design of agentic systems. In: The Thirteenth International Conference on Learning Representations (2025)
9. Jiang, D., Ren, X., Lin, B.Y.: LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 14165–14178 (2023)
10. Jiang, E.H., Wan, G., Yin, S., Li, M., Wu, Y., Liang, X., Li, X., Sun, Y., Wang, W., Chang, K.W., Wu, Y.N.: Dynamic generation of multi-LLM agents communication topologies with graph diffusion models. arXiv preprint arXiv:2510.07799 (2025)
11. Li, S., Liu, Y., Wen, Q., Zhang, C., Pan, S.: Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 40, pp. 23142–23150 (2026)
12. Liu, Z., Zhang, Y., Li, P., Liu, Y., Yang, D.: Dynamic LLM-agent network: An LLM-agent collaboration framework with agent team optimization. arXiv preprint arXiv:2310.02170 (2023)
13. OpenAI: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
14. Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., et al.: ChatDev: Communicative agents for software development. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 15174–15186 (2024)
15. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: International Conference on Learning Representations (2021)
16. Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., Frossard, P.: DiGress: Discrete denoising diffusion for graph generation. In: The Eleventh International Conference on Learning Representations (2023)

17. Wang, X., Wei, J., Schuurmans, D., Le, Q.V., Chi, E.H., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. In: The Eleventh International Conference on Learning Representations (2023)
18. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)
19. Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., et al.: AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In: First Conference on Language Modeling (2024)
20. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2019)
21. Zhang, G., Niu, L., Fang, J., Wang, K., Bai, L., Wang, X.: Multi-agent architecture search via agentic supernet. In: Forty-second International Conference on Machine Learning (2025)
22. Zhang, G., Yue, Y., Li, Z., Yun, S., Wan, G., Wang, K., Cheng, D., Yu, J.X., Chen, T.: Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In: The Thirteenth International Conference on Learning Representations (2025)
23. Zhang, G., Yue, Y., Sun, X., Wan, G., Yu, M., Fang, J., Wang, K., Chen, T., Cheng, D.: G-Designer: Architecting multi-agent communication topologies via graph neural networks. In: ICLR 2025 Workshop on Foundation Models in the Wild (2025)
24. Zhang, J., Xiang, J., Yu, Z., Teng, F., Chen, X.H., Chen, J., Zhuge, M., Cheng, X., Hong, S., Wang, J., et al.: AFlow: Automating agentic workflow generation. In: The Thirteenth International Conference on Learning Representations (2025)
25. Zhuge, M., Wang, W., Kirsch, L., Faccio, F., Khizbullin, D., Schmidhuber, J.: GPTSwarm: Language agents as optimizable graphs. In: Forty-first International Conference on Machine Learning (2024)